1.1 FUNCTION DESCRIPTION

There are six exported functions to consist the UART.DLLs.

1.1.1 Open_Com

Description:

This DLL will initialize the COM port and This DLL must be **called firstly before** the
other DLLs are called to send/receive command.

Syntax:

WORD Open_Com(char cPort, DWORD dwBaudRate, char cData, char cParity, char cStop)

Input Parameter:

cPort: 1=COM1, 2=COM2 ,3=COM3…. , 255=COM255
dwBaudRate:
300/600/1200/1800/2400/4800/7200/9600/19200/38400/57600/115200
cData: 5/6/7/8 data bit
cParity: 0=NonParity, 1=OddParity, 2=EvenParity
cStop: 0=1-stop, 1=1.5-stp, 2=2-stop

NOTE: cData=8, cParity=0, cStop=0 for 9000 modules

Return Value:

NoError = OK
others = Error code, refer to EX9000.H

1.1.2 Close_Com

Description:

This DLL will be free for all the resources used by Open_Com. This DLL must be **called before** the program exit. The Open_Com will return error message if the program exit without calling Close_Com function.

Syntax:

WORD Close_Com(char cPort)

Input Parameter:

cPort : 1=COM1, 2=COM2 ,3=COM3….. , 255=COM255

Return Value:

NoError = OK
others = Error code, refer to EX9000.H

Example:

```
Open_Com(COM_PORT,115200L,cData,cParity,cStop); // open com first
Close_Com(COM_PORT); // close com if stop the program
```

The other DLLs are called here

1.1.3 Send_Cmd

Description:

This DLL will create a thread to send a command to 9000 and this DLL will receive the
response-result from 9000. If the wCheckSum=1, this DLL will automatically **add the two checksum bytes** to the input string. This DLL will **add the [0x0D]** to the end
of the input string, szCmd. The Send_Cmd is a multi-task and multi-thread of DLL.

Syntax:

WORD Send_Cmd(char cPort, char szCmd[], WORD wTimeOut, WORD wCheckSum)

Input Parameter:

cPort: 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate
szCmd: the starting address of the original command string (terminated with 0)
wTimeOut: constant for time-out control and unit = 1ms
wCheckSum: 1=ENABLE , 0=DISABLE

Return Value:

NoError : OK
others = Error code, refer to EX9000.H

Example :

```
Open_Com(COM_PORT,115200L,cData,cParity,cStop);// open com first
.
strcpy(cBuf,"$01M"); // check module name
ret = Send_Cmd(1,cBuf,1000,0); // create a thread
// time-out=1000 ms=1 sec
// checksum is disable
for (;;) // check the thread status
{
wRet=Read_Com_Status(1,cResult,&wT);
if (wRet >= 0x100) break;
}
sprintf(cShow,"Receive=%s wRet=%x, wT=%d",cResult,wRet,wT);
TextOut(hdc,1,yy,cShow,strlen(cShow)); yy+=16;
.
Close_Com(COM_PORT); // close com if stop the program
```

1.1.4 Read_Com_Status

Description:

The **Send_Cmd(**char cPort, char szCmd[], WORD wTimeOut, WORD wCheckSum) will create a thread to send a command to 9000 and receive the response-result from 9000. The Read_Com_Status will return the status of this
send/receive thread.**The return value will be equal to 0x105 if the send/receive operation is OK.** If the thread is finished, the status value will be larger than 0x100.
If the thread is working, the status value will be smaller than 0x100**.**
If the wCheckSum in Send_Cmd is 1, the Recom_Com_Status will check the two checksum bytes of result string. **If the checksum is incorrect, the D13 of return value will be set to 1.**

Syntax:

WORD Read_Com_Status(char cPort, char szResult[], WORD *wT)

Input Parameter:

cPort : 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate
szResult: the starting address of the result string (terminated with 0)
wT: time of send/receive interval and unit = 1 ms

Return Value:

D0-D7= thread status code, thread start=1 and stop=5
D8=1! for **send/receive finish, 0**! for **send/receive not finish**
D9= 1! for send/receive timeout
D10= reserved
D11= 1! for Com handle error
D12= 1! for send/receive overflow
D13= 1! for checksum error

Example :

```
strcpy(cBuf,"$01M"); // check module name
ret = Send_Cmd(1,cBuf,1000,0); // create a thread
for (;;) { // check the thread status
wRet = Read_Com_Status(1,cResult,&wT);
if (wRet >= 0x100) break;
}
if (wRet != 0x105) {
// this is a error condition
}
```

1.1.5 Send_Str

Description:

This DLL will create a thread to send a command to 9000 and receive the response-result
from 9000. The Send_Str is a multi-task and multi-thread of DLL. This DLL is very similar to Send_Cmd except that this DLL will **not add any char** to the input string.

Syntax:

WORD Send_Str(char cPort, char szCmd[], WORD wTimeOut, WORD wSendLen,
WORD wReceiveLen)

Input Parameter:

cPort: 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate
szCmd: the starting address of the original command string (terminated with 0)
wTimeOut: constant for time-out control and unit = 1ms
wSendLen: string length of send-string
wReceiveLen: string length of receive-string

Return Value:

NoError : OK
others = Error code, refer to EX9000.H

Example :

```
Open_Com(COM_PORT,9600L,cData,cParity,cStop); // open com first
.
.
strcpy(cBuf,"<01C00009B22>"); // counter command
wSendLen=13; // send-string=<01C00009B22>
wReceiveLen=10; // receive-string=(00009BC6)
wRet = Send_Str(cPort,cBuf,1000,wSendLen,wReceiveLen); // create a thread
for (;;) // check the thread status
{
wRet=Read_Com_Status(1,cResult,&wT);
if (wRet >= 0x100) break;
}
.
.
Close_Com(COM_PORT); // close com if stop the program
```

1.1.6 Send_Receive_Cmd

Description:

This DLL will send a command to 9000 and receive the response-result from 9000. If the wCheckSum=1, this DLL will automatically **add the two checksum bytes** to the input string and check the checksum status of the receive string. This DLL will **add the [0x0D]** to the end of the input string, szCmd. The Send_Receive_Cmd is not a multi-task and multi-thread of DLL.

Syntax:

WORD Send_Receive_Cmd(char cPort, char szCmd[], char szResult[], WORD wTimeOut, WORD wCheckSum, WORD *wT)

Input Parameter:

cPort: 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate
szCmd: the starting address of the original command string (terminated with 0)
szResult: the starting address of the result string
wTimeOut: constant for time-out control and unit = 1ms
wCheckSum: 1=ENABLE , 0=DISABLE
wT: time of send/receive interval and unit = 1 ms

Return Value:

NoError : OK
others = Error code, refer to EX9000.H

Example:

```
Open_Com(COM_PORT,115200L,cData,cParity,cStop); // open com first
.
.
strcpy(cBuf,"$01M"); // check module name
wRet = Send_Receive_Cmd(cPort,cBuf,cResult,wTimeOut,wCheckSum,&wT);
if (wRet != NoError)
{
// something error
}
.
.
Close_Com(COM_PORT); // close com if stop the program
```

2.1 Aanolog I/O & Digital I/O Functions

2.2. Analog Input/Output Functions

2.2.1 AnalogIn

Description:
Read the analog input value from 9000.
Syntax:
AnalogIn(WORD wIPBUF[], float fOPBUF[], char szSendTo9000[], char szReceiveFrom9000[])
Input Parameter:
w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000
Return Value:
NoError: OK
others: Error code, refer to EX9000.H .

W9000: WORD Input/Output Table
wIPBUF[0] : 1 to 255 for RS-232 port number
wIPBUF[1] : 00 to FF for module address
wIPBUF[2] : module ID for 0x9011/9012/9013/9014/9017/9018/9033
wIPBUF[3] : 0=disable , 1=enable for checksum
wIPBUF[4] : normal=100 for TimeOut constant
wIPBUF[5] : channel number for multi-channel module (9017/9018/9033)
wIPBUF[6] : 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000
F9000: Float Input/Output Table
fOPBUF[0] : analog input value return
Example:

```
wIPBUF[0]=cPort; // port number number
wIPBUF[1]=wAddr; // Module Address
wIPBUF[2]=0x9033; // Module ID
wIPBUF[3]=0; // CheckSum disable
wIPBUF[4]=wTimeOut; // normal=100 for TimeOut constant
wIPBUF[5]=0; // module ID of Channel 0
wIPBUF[6]=1; //save , string debug
wRet= AnalogIn(w9000, f9000, szSend, szReceive);
if (wRet != 0 ) {
// Something Error .
}
else {
fFloatVal = fOPBUF[0]; // The analog input value
}
```

2.2.2 AnalogInHex

Description:
Read the analog input value(in Hex format) from 9000.
Syntax:
AnalogInHex(WORD wIPBUF[], float fOPBUF[], char szSendTo9000[], char szReceiveFrom9000[])
Input Parameter:
w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000
Return Value:
NoError: OK
others: Error code, refer to EX9000.H .

W9000: WORD Input/Output Table
wIPBUF[0] : 1 to 255 for RS-232 port number
wIPBUF[1] : 00 to FF for module address
wIPBUF[2] : module ID for 0x9011/9012/9013/9014/9017/9018/9033
wIPBUF[3] : 0=disable , 1=enable for checksum
wIPBUF[4] : normal=100 for TimeOut constant
wIPBUF[5] : channel number for multi-channel module (9017/9018/9033)
wIPBUF[6] : 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000
wIPBUF[7]: The analog input value in Hex format.
F9000: Float Input/Output Table
Null & void.
Example:

```
wIPBUF[0]=cPort; // port number
wIPBUF[1]=wAddr; // Module Address
wIPBUF[2]=0x9033; // Module ID
wIPBUF[3]=0; // CheckSum disable
wIPBUF[4]=wTimeOut; // normal=100 for TimeOut constant
wIPBUF[5]=0; // Module ID of Channel 0
wIPBUF[6]=1; // save , string debug
wRet= AnalogInHex(w9000, f9000, szSend, szReceive);
if (wRet != 0 ) {
// Something Error .
}
else {
wHexVal = wIPBUF[7]; // The analog input value
}
```

2.2.3 AnalogInFsr

Description:
Read the analog input value(in FSR format) from 9000.
Syntax:
AnalogInFsr(WORD wIPBUF[], float fOPBUF[], char szSendTo9000[], char szReceiveFrom9000[])
Input Parameter:
w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000
Return Value:
NoError: OK
others: Error code, refer to EX9000.H .

W9000: WORD Input/Output Table
wIPBUF[0] : 1 to 255 for RS-232 port number
wIPBUF[1] : 00 to FF for module address
wIPBUF[2] : module ID for 0x9011/9012/9013/9014/9017/9018/9033
wIPBUF[3] : 0=disable , 1=enable for checksum
wIPBUF[4] : normal=100 for TimeOut constant
wIPBUF[5] : channel number for multi-channel module(9017/9018/9033)
wIPBUF[6] : 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000
F9000: Float Input/Output Table
fOPBUF[0]: The analog input value.
Example:

```
wIPBUF[0]=cPort; // port number
wIPBUF[1]=wAddr; // Module Address
wIPBUF[2]=0x9033; // Module ID
wIPBUF[3]=0; // CheckSum disable
wIPBUF[4]=wTimeOut; // normal=100 for TimeOut constant
wIPBUF[5]=0; // Module ID of Channel 0
wIPBUF[6]=1; // save , string debug
wRet= AnalogInFsr(w9000, f9000, szSend, szReceive);
if (wRet != 0 ) {
// Something Error .
}
else {
fFsrVal = fOPBUF[0]; // The analog input value
}
```

2.2.4 AnalogIn8

Description:
Read the 8 channels of analog input values from 9017 or 9018.
Syntax:
AnalogIn8(WORD wIPBUF[], float fOPBUF[], char szSendTo9000[], char szReceiveFrom9000[])
Input Parameter:
w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000
Return Value:
NoError: OK
others: Error code, refer to EX9000.H .

W9000: WORD Input/Output Table
wIPBUF[0] : 1 to 255 for RS-232 port number
wIPBUF[1] : 00 to FF for module address
wIPBUF[2] : module ID for 0x9017/9018
wIPBUF[3] : 0=disable , 1=enable for checksum
wIPBUF[4] : normal=100 for TimeOut constant
wIPBUF[6] : 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000
F9000: Float Input/Output Table
fOPBUF[0] : analog input value of channel_0
fOPBUF[1] : analog input value of channel_1
……
fOPBUF[7] : analog input value of channel_7

2.2.5 AnalogOut

Description:
Send the analog output command to a 9000 module.
Syntax:
AnalogOut(WORD wIPBUF[], float fOPBUF[], char szSendTo9000[], char szReceiveFrom9000[])
Input Parameter:
w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000 (for debug)
szReceiveFrom9000: result string receive from 9000 (for debug)
Return Value:
NoError: OK
others: Error code, refer to EX9000.H .

W9000: WORD Input/Output Table
wIPBUF[0] : 1 to 255 for RS-232 port number
wIPBUF[1] : 00 to FF for module address
wIPBUF[2] : module ID for 0x9021/ 0x9024
wIPBUF[3] : 0=disable , 1=enable for checksum
wIPBUF[4] : normal=100 for TimeOut constant
wIPBUF[5] : Channel No. (0 to 3) if module ID is 9024 ;
// Null & void if module ID is 9021
wIPBUF[6] : 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000
F9000: Float Input/Output Table
fOPBUF[0] : analog output value
Example:
wIPBUF[0]=cPort; // port number wIPBUF[1]=wAddr; // Module Address
wIPBUF[2]=0x9021; // Module ID wIPBUF[3]=0; // CheckSum disable
wIPBUF[4]=wTimeOut; // TimeOut constant wIPBUF[6]=1; // save , string debug
fOPBUF[0]=5.432; // DA output value
wRet=AnalogOut(w9000, f9000, szSend, szReceive);

2.2.6 AnalogOutReadBack

Description:

Read back the current D/A output value of 9021/9024. There are two types of analog
output read back described as following:

1. **command read back by $AA6 command**
2. **analog output of current path read back by $AA8 command**

Syntax:

AnalogOutReadBack(WORD wIPBUF[], float fOPBUF[], char szSendTo9000[], char
szReceiveFrom9000[])

Input Parameter:

w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000 (for debug)
szReceiveFrom9000: result string receive from 9000 (for debug)

Return Value:

NoError: OK
others: Error code, refer to EX9000.H .

W9000: WORD Input/Output Table

wIPBUF[0]: 1 to 255 for RS-232 port number
wIPBUF[1]: 00 to FF for module address
wIPBUF[2]: module ID for 0x9021 / 0x9024
wIPBUF[3]: 0=disable , 1=enable for checksum
wIPBUF[4]: normal=100 for TimeOut constant
wIPBUF[5]: 0: command read back ($AA6)
1: analog output of current path read back ($AA8)
wIPBUF[6]: 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000
wIPBUF[7] : Channel No.(0 to 3), if wID is 9024
Null & void if module ID is 9021 .

F9000: Float Input/Output Table

fOPBUF[0] : analog output read back value

2.3 Digital Input/Output Functions

2.3.1 DigitalIn

Description:
Read the digital input value from a 9000 module.
Syntax:
DigitalIn(WORD wIPBUF[], float fOPBUF[], char szSendTo9000[], char szReceiveFrom9000[])
Input Parameter:
w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000
Return Value:
NoError: OK
others: Error code, refer to EX9000.H .

W9000: WORD Input/Output Table
wIPBUF[0] : 1 to 255 for RS-232 port number
wIPBUF[1] : 00 to FF for module address
wIPBUF[2] : module ID for 0x9050/9052/9053/9060/9041/9044
wIPBUF[3] : 0=disable , 1=enable for checksum
wIPBUF[4] : normal=100 for TimeOut constant
wIPBUF[5] : 16-bit digital input data
wIPBUF[6] : 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000
f9000: Float Input/Output Table
Null & void
Example :

```
wIPBUF[0]=cPort; // port number
wIPBUF[1]=wAddr; // Module Address
wIPBUF[2]=0x9053; // Module ID
wIPBUF[3]=0; // CheckSum disable
wIPBUF[4]=wTimeOut; // TimeOut constant
wIPBUF[6]=1; // save , string debug
wRet=DigitalIn(w9000, f9000, szSend, szReceive);
```

2.3.2 DigitalOut

Description:
To set the digital output value for a 9000 module.
Syntax:
DigitalOut(WORD wIPBUF[], float fOPBUF[], char szSendTo9000[], char szReceiveFrom9000[])
Input Parameter:
w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000
Return Value:
NoError: OK
others: Error code, refer to EX9000.H .

W9000: WORD Input/Output Table
wIPBUF[0] : 1 to 255 for RS-232 port number
wIPBUF[1] : 00 to FF for module address
wIPBUF[2] : module ID for 0x9050/9060/9067/9042/9043/9044/9011/9012/9014
wIPBUF[3] : 0=disable , 1=enable for checksum
wIPBUF[4] : normal=100 for TimeOut constant
wIPBUF[5] : 16-bit digital output data
wIPBUF[6] : 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000
F9000: Float Input/Output Table
Null & void
Example:

```
wIPBUF[0]=cPort; // port number
wIPBUF[1]=wAddr; // Module Address
wIPBUF[2]=0x9053; // Module ID
wIPBUF[3]=0; // CheckSum disable
wIPBUF[4]=wTimeOut; // TimeOut constant
wIPBUF[5]=wDoVal; // digital output value
wIPBUF[6]=1; // save , string debug
wRet=DigitalOut(w9000, f9000, szSend, szReceive);
```

2.3.3 DigitalOutReadBack

Description:
Read back the digital output value of a 9000 module.
Syntax:
DigitalOutReadBack(WORD wIPBUF[], float fOPBUF[], char szSendTo9000[], char
szReceiveFrom9000[])
Input Parameter:
w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000
Return Value:
NoError: OK
others: Error code, refer to 9000.H .

W9000: WORD Input/Output Table
wIPBUF[0] : 1 to 255 for RS-232 port number
wIPBUF[1] : 00 to FF for module address
wIPBUF[2] : module ID for 0x9050/9060/9067/9042/9043/9044
wIPBUF[3] : 0=disable , 1=enable for checksum
wIPBUF[4] : normal=100 for TimeOut constant
wIPBUF[5] : 16-bit digital output data read back
wIPBUF[6] : 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000

f9000: Float Input/Output Table
Null & void
Example:
```
wIPBUF[0]=cPort; // port number
wIPBUF[1]=wAddr; // Module Address
wIPBUF[2]=0x9053; // Module ID
wIPBUF[3]=0; // CheckSum disable
wIPBUF[4]=wTimeOut; // TimeOut constant
wIPBUF[6]=1; // save , string debug
wRet=DigitalOutReadBack(w9000, f9000, szSend, szReceive);
// wIPBUF[5] = digital output read back
```

2.4 Misc Functions

NetworkAnalogIn

Description:

Read the multi-module analog input value from 9000 RS-485 network. The user can call AnalogIn to read analog input value one by one or call this function once for easy programming.

Syntax:

NetworkAnalogIn(WORD wPort, WORD wTotal, WORD wT, WORD wID[], WORD wConfig[], WORD wChksum[], float fOPBUF[])

Input Parameter:

wPort: 1 to 255 for RS-232 port number
wTotal: number of modules to read
wT: normal=100 for TimeOut constant
wID: wID[?]=module address of module_?, from 0x00 to 0xFF
wConfig: if wID[?]=0x9017 then wConfig[?]=08/09/0A/0B/0C/0D
if wID[?] !=0x9017 then wConfig[?] is ignored
wChksum: if wChksum[?]=1 ! the checksum of module_? is enable
fOPBUF[0]: analog value of module_0, channel_0
fOPBUF[1]: analog value of module_0, channel_1
……..
fOPBUF[7]: analog value of module_0, channel_7
fOPBUF[8]: analog value of module_1, channel_0
……..
fOPBUF[15]: analog value of module_1, channel_7
fOPBUF[n*8]: analog value of module_n, channel_0
fOPBUF[n*8+1]: analog value of module_n, channel_1
……..
fOPBUF[n*8+7]: analog value of module_n, channel_7

Return Value:

NoError: OK
others: Error code, refer to EX9000.H .

Example:

```
Open_Com(COM_PORT,9600L,cData,cParity,cStop); // open com firstly
wTotal=4; // total 4 modules
wAddr[0]=3; // module address of module_0 = 0x03
wID[0]=0x9050; // module_0 is 9050
wCheckSum[0]=0; // the checksum of module_0 is disable
wAddr[1]=0x11; // module address of module_1 = 0x11
wID[1]=0x9052; // module_1 is 9052
wCheckSum[1]=1; // the checksum of module_1 is enable
wAddr[2]=0x20; // module address of module_2 = 0x20
wID[2]=0x9053; // module_2 is 9053
wCheckSum[2]=1; // the checksum of module_2 is enable
wAddr[3]=0x21; // module address of module_3 = 0x21
```

```
wID[3]=0x9060; // module_3 is 9060
wCheckSum[3]=0; // the checksum of module_3 is disable
wRet=NetworkDigitalIn(cPort, wTotal, wTimeOut, wAddr, wID, wConfig,
wCheckSum, w9000);
if (wRet == NoError)
{
// wIPBUF[0]=16-bit digital value of module_0 (9050 is 7-bit)
// wIPBUF[1]=16-bit digital value of module_1 (9052 is 8-bit)
// wIPBUF[2]=16-bit digital value of module_2 (9053 is 16-bit)
// wIPBUF[3]=16-bit digital value of module_3 (9060 is 4-bit)
}
Close_Com(COM_PORT); // close com if stop the program
```

NetworkDigitalIn

Description:

Read the multi-module digital input value from 9000 RS-485 network. The user can call DigitalIn to read analog input value one by one or call this function once for easy programming.

Syntax:

NetworkDigitalIn(WORD wPort, WORD wTotal, WORD wT, WORD wID[], WORD wConfig[], WORD wChksum[], WORD wIPBUF[])

Input Parameter:

wPort: 1 to 255 for RS-232 port number
wTotal: number of modules to read
wT: normal=100 for TimeOut constant
wID: wID[?]=module address of module_?, from 0x00 to 0xFF
wConfig: reserved.
wChksum: if wChksum[?]=1 ! the checksum of module_? is enable
wIPBUF[0]: 16-bit digital value of module_0
wIPBUF[1]: 16-bit digital value of module_1
…….
wIPBUF[n]: 16-bit digital value of module_n

Return Value:

NoError: OK
others: Error code, refer to EX9000.H .

```
Open_Com(COM_PORT,9600L,cData,cParity,cStop); // open com first
wTotal=4; // total 4 modules
wAddr[0]=3; // module address of module_0 = 0x03
wID[0]=0x9012; // module_0 is 9012
wCheckSum[0]=0; // the checksum of module_0 is disable
wAddr[1]=0x11; // module address of module_1 = 0x11
wID[1]=0x9013; // module_1 is 9013
wCheckSum[1]=1; // the checksum of module_1 is enable
wAddr[2]=0x20; // module address of module_2 = 0x20
wID[2]=0x9018; // module_2 is 9018
```

```
wCheckSum[2]=1; // the checksum of module_2 is enable
wAddr[3]=0x21; // module address of module_3 = 0x21
wID[3]=0x9017; // module_3 is 9017
wConfig[3]=0x08; // +/-10V range
wCheckSum[3]=0; // the checksum of module_3 is disable
wRet=NetworkAnalogIn(cPort, wTotal, wTimeOut, wAddr, wID, wConfig,
wCheckSum, f9000);
if (wRet == NoError)
{
// fOPBUF[0]=analog value of module_0 (9012 is single channel)
// fOPBUF[8]=analog value of module_1 (9013 is single channel)
// fOPBUF[16+0]=analog value of module_2, channel_0
// ……..
// fOPBUF[16+7]=analog value of module_2, channel_7
// fOPBUF[24+0]=analog value of module_3, channel_0
// ……..
// fOPBUF[24+7]=analog value of module_3, channel_7
}
Close_Com(COM_PORT); // close com if stop the program
```

2.4.1 ThermocoupleOpen_9011 [9011 only]

Description:
To detect if the thermocouple is open for a 9011 module.
Syntax:
ThermocoupleOpen_9011(WORD wIPBUF[], float fOPBUF[], char szSendTo9000[],
char szReceiveFrom9000[])
Input Parameter:
w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000
Return Value:
NoError: OK
others: Error code, refer to EX9000.H .
W9000: WORD Input/Output Table
wIPBUF[0] : 1 to 255 for RS-232 port number
wIPBUF[1] : 00 to FF for module address
wIPBUF[2] : module ID for 0x9011
wIPBUF[3] : 0=disable , 1=enable for checksum
wIPBUF[4] : normal=100 for TimeOut constant
wIPBUF[5] : 0 ! the thermocouple is close
1 ! the thermocouple is open
wIPBUF[6] : 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000

F9000: Float Input/Output Table
Null & void
Example:
Code Fragment

```
wIPBUF[0]=cPort; // port number
wIPBUF[1]=wAddr; // Module Address
wIPBUF[2]=0x9011; // Module ID
wIPBUF[3]=0; // CheckSum disable
wIPBUF[4]=wTimeOut; // TimeOut constant, normal=100
wIPBUF[6]=1; // save , string debug
wRet=Thermocouple_9011(w9000, f9000, szSend, szReceive);
if( wIPBUF[5]==0 )
{
// the thermocouple is close to this 9011 module
}
else
{
// the thermocouple is open to this 9011 module
}
```

2.4.2 GetLedDisplay_9033

Description:

The module 9033 has three channels of Analog Input. The LED can display the Analog Input value of specify channel. The function GetLedDisplay_9033() can get the
active channel number for 9033 module.

Syntax:

GetLedDisplay_9033(WORD wIPBUF[], float fOPBUF[], char szSendTo9000[], char
szReceiveFrom9000[])

Input Parameter:

w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000

Return Value:

NoError: OK
others: Error code, refer to EX9000.H .

W9000: WORD Input/Output Table

wIPBUF[0] : 1 to 255 for RS-232 port number
wIPBUF[1] : 00 to FF for module address
wIPBUF[2] : module ID for 0x9033
wIPBUF[3] : 0=disable , 1=enable for checksum
wIPBUF[4] : normal=100 for TimeOut constant
wIPBUF[5] : Return the channel number that is actived (Displayed)
wIPBUF[6] : 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000

F9000: Float Input/Output Table

Null & void

Example:

Code Fragment

```
wIPBUF[0]=cPort; // port number
wIPBUF[1]=wAddr; // Module Address
wIPBUF[2]=0x9033; // Module ID
wIPBUF[3]=0; // CheckSum disable
wIPBUF[4]=wTimeOut; // normal=100 for TimeOut constant
wIPBUF[6]=1; // save , string debug
wRet=GetLedDisplay_9033(w9000, f9000, szSend, szReceive);
if( wRet != 0) {
// Something Error
}
else {
wChannelNo = wIPBUF[5];
}
```

2.4.3 SetLedDisplay_9033

Description:

The module 9033 has three channels of Analog Input. The LED can display the Analog Input value of specify channel. The function SetLedDisplay_9033() is used to set
the active channel number for 9033 module.

Syntax:

SetLedDisplay_9033(WORD wIPBUF[], float fOPBUF[], char szSendTo9000[], char
szReceiveFrom9000[])

Input Parameter:

w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000

Return Value:

NoError: OK
others: Error code, refer to EX9000.H .

W9000: WORD Input/Output Table

wIPBUF[0] : 1 to 255 for RS-232 port number
wIPBUF[1] : 00 to FF for module address
wIPBUF[2] : module ID for 0x9033
wIPBUF[3] : 0=disable , 1=enable for checksum
wIPBUF[4] : normal=100 for TimeOut constant
wIPBUF[5] : The channel number that is setted to actived (Displayed)
wIPBUF[6] : 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000

F9000: Float Input/Output Table

Null & void

Example:

Code Fragment

```
wIPBUF[0]=cPort; // port number
wIPBUF[1]=wAddr; // Module Address
wIPBUF[2]=0x9033; // Module ID
wIPBUF[3]=0; // CheckSum disable
wIPBUF[4]=wTimeOut; // normal=100 for TimeOut constant
wIPBUF[5]=0; // Setting the channel 0 to be dispalyed.
wIPBUF[6]=1; // save , string debug
wRet=SetLedDisplay_9033(w9000, f9000, szSend, szReceive);
if( wRet != 0) {
// Something Error
}
```

2.5 Alarm Functions

2.5.1 EnableAlarm [9011/9012/9014 only]

Description:

To let 9000 module enter *momentary alarm mode* or *latch alarm mode*.

Syntax:

EnabledAlarm(WORD wIPBUF[], float fOPBUF[], char szSendTo9000[], char szReceiveFrom9000[])

Input Parameter:

w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000

Return Value:

NoError: OK
others: Error code, refer to EX9000.H .

W9000: WORD Input/Output Table

wIPBUF[0] : 1 to 255 for RS-232 port number
wIPBUF[1] : 00 to FF for module address
wIPBUF[2] : module ID for 0x9011/9012/9014
wIPBUF[3] : 0=disable , 1=enable for checksum
wIPBUF[4] : normal=100 for TimeOut constant
wIPBUF[5] : 0 ! momentary alarm mode
1 ! latch alarm mode
wIPBUF[6] : 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000

F9000: Float Input/Output Table

Null & void

2.5.2 DisableAlarm [9011/9012/9014 only]

Description:
To disable alarm function for a 9000 module.
Syntax:
DisableAlarm(WORD wIPBUF[], float fOPBUF[], char szSendTo9000[], char szReceiveFrom9000[])
Input Parameter:
w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000
Return Value:
NoError: OK
others: Error code, refer to EX9000.H .
W9000: WORD Input/Output Table
wIPBUF[0] : 1 to 255 for RS-232 port number
wIPBUF[1] : 00 to FF for module address
wIPBUF[2] : module ID for 0x9011/9012/9014
wIPBUF[3] : 0=disable , 1=enable for checksum
wIPBUF[4] : normal=100 for TimeOut constant
wIPBUF[5] : Null & void
wIPBUF[6] : 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000
F9000: Float Input/Output Table
Null & void

2.5.3 ClearLatchAlarm [9011/9012/9014 only]

Description:
To clear the latch alarm for a 9000 mode.
Syntax:
ClearLatchAlarm(WORD wIPBUF[], float fOPBUF[], char szSendTo9000[], char szReceiveFrom9000[])
Input Parameter:
w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000
Return Value:
NoError: OK
others: Error code, refer to EX9000.H .
W9000: WORD Input/Output Table
wIPBUF[0] : 1 to 255 for RS-232 port number
wIPBUF[1] : 00 to FF for module address
wIPBUF[2] : module ID for 0x9011/9012/9014
wIPBUF[3] : 0=disable , 1=enable for checksum
wIPBUF[4] : normal=100 for TimeOut constant
wIPBUF[5] : Null & void
wIPBUF[6] : 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000
F9000: Float Input/Output Table
Null & void

2.5.4 SetAlarmLimitValue [9011/9012/9014 only]

Description:
To set a high or low alarm limit value for a 9000 module.
Syntax:
SetAlarmLimitValue(WORD wIPBUF[], float fOPBUF[], char szSendTo9000[], char
szReceiveFrom9000[])
Input Parameter:
w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000

Return Value:

NoError: OK
others: Error code, refer to EX9000.H .
W9000: WORD Input/Output Table
wIPBUF[0]: 1 to 255 for RS-232 port number
wIPBUF[1]: 00 to FF for module address
wIPBUF[2]: module ID for 0x9011/0x9012/0x9014

wIPBUF[3]: 0=disable , 1=enable for checksum
wIPBUF[4]: normal=100 for TimeOut constant
wIPBUF[5]: 0 ! low alarm value setting
1 ! high alarm value setting
wIPBUF[6]: 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000

F9000: Float Input/Output Table
fOPBUF[0] : alarm value

Example:
Code Fragment

```
wIPBUF[0]=cPort; // port number
wIPBUF[1]=wAddr; // Module Address
wIPBUF[2]=0x9011; // Module ID
wIPBUF[3]=0; // CheckSum disable
wIPBUF[4]=wTimeOut; // normal=100 for TimeOut constant
wIPBUF[5]=0; // to set low alarm value
wIPBUF[6]=1; // save , string debug
fOPBUF[0]=300.0; // the low alarm value is 300
wRet=SetAlarmLimitValue(w9000, f9000, szSend, szReceive);
if( wRet>0 )
{
// the setting alarm value process is error
}
else
{
// the setting alarm value process is successful
}
```

2.5.5 ReadAlarmLimitValue [9011/9012/9014 only]

Description:
To get the high or low alarm limit value for a 9000 module.
Syntax:
GetAlarmLimitValue(WORD wIPBUF[], float fOPBUF[], char szSendTo9000[], char
szReceiveFrom9000[])
Input Parameter:
w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000
Return Value:
NoError: OK
others: Error code, refer to EX9000.H .
W9000: WORD Input/Output Table
wIPBUF[0]: 1 to 255 for RS-232 port number
wIPBUF[1]: 00 to FF for module address
wIPBUF[2]: module ID for 0x9011/0x9012/0x9014
wIPBUF[3]: 0=disable , 1=enable for checksum
wIPBUF[4]: normal=100 for TimeOut constant
wIPBUF[5]: 0 ! low alarm value setting
1 ! high alarm value setting
wIPBUF[6]: 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000
F9000: Float Input/Output Table
fOPBUF[0] : alarm value

Example:
Code Fragment

```
wIPBUF[0]=cPort; // port number
wIPBUF[1]=wAddr; // Module Address
wIPBUF[2]=0x9011; // Module ID
wIPBUF[3]=0; // CheckSum disable
wIPBUF[4]=wTimeOut; // normal=100 for TimeOut constant
wIPBUF[5]=0; // to get the low alarm value
wIPBUF[6]=1; // save , string debug
wRet=GetAlarmLimitValue(w9000, f9000, szSend, szReceive);
if( wRet>0 )
{
// the setting alarm value process is error
}
else
{
// the get alarm limit value process is successful
```

// and the alarm limit value stored in the fOPBUF[0]
}

2.5.6 ReadOutputAlarmState [9011/9012/9014 only]

Description:
Reading the alarm mode and digital output value for a 9000 module.
Syntax:
ReadOutputAlarmState(WORD wIPBUF[], float fOPBUF[], char szSendTo9000[], char
szReceiveFrom9000[])
Input Parameter:
w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000
Return Value:
NoError: OK
others: Error code, refer to EX9000.H .
W9000: WORD Input/Output Table
wIPBUF[0]: 1 to 255 for RS-232 port number
wIPBUF[1]: 00 to FF for module address
wIPBUF[2]: module ID for 0x9011/0x9012/0x9014
wIPBUF[3]: 0=disable , 1=enable for checksum
wIPBUF[4]: normal=100 for TimeOut constant
wIPBUF[5]: no used
wIPBUF[6]: 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000
wIPBUF[7]: 0 ! alarm disable
1 ! momentary alarm
2 ! latch alarm

wIPBUF[8]: 0 ! DO:0 off DO:1 off
1 ! DO:0 on DO:1 off
2 ! DO:0 off DO:1 on
3 ! DO:0 on DO:1 lying on
F9000: Float Input/Output Table
Null & void
Example:

```
wIPBUF[0]=cPort; // port number
wIPBUF[1]=wAddr; // Module Address
wIPBUF[2]=0x9012; // Module ID
wIPBUF[3]=0; // CheckSum disable
wIPBUF[4]=wTimeOut; // normal=100 for TimeOut constant
wIPBUF[6]=1; // save , string debug
wRet=ReadOutputAlarmState(w9000, f9000, szSend, szReceive);
if( wRet>0 ) {
```

```
// some error occur !!!
}
else {
switch( wIPBUF[7] ) // to detect the alarm status
{
case 0: // the alarm is disable
break;
case 1: // the momentary alarm
break;
case 2: // the latch alarm
break;
}
switch( wIPBUF[8] ) // to detect the digital output status
{
case 0: // the DO:0 is off, DO:1 is off
break;
case 1: // the DO:0 is on, DO:1 is off
break;
case 2: // the DO:0 is off, DO:1 is on
break;
case 3: // the DO:0 is on, DO:1 is on
break;
}
}
```

2.6 Event Counter Functions

2.6.1 ReadEventCounter [9011/9012/9014 only]

Description:

To read the value of event counter for a 9000 module.

Syntax:

ReadEventCounter(WORD wIPBUF[], float fOPBUF[], char szSendTo9000[], char szReceiveFrom9000[])

Input Parameter:

w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000

Return Value:

NoError: OK
others: Error code, refer to EX9000.H .

W9000: WORD Input/Output Table

wIPBUF[0]: 1 to 255 for RS-232 port number
wIPBUF[1]: 00 to FF for module address
wIPBUF[2]: module ID for 0x9011/0x9012/0x9014
wIPBUF[3]: 0=disable , 1=enable for checksum
wIPBUF[4]: normal=100 for TimeOut constant
wIPBUF[5]: Null & void
wIPBUF[6]: 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000
wIPBUF[7]: the reading event counter value

F9000: Float Input/Output Table

Null & void

2.6.2 ClearEventCounter [9011/9012/9014 only]

Description:

To clear the value of event counter for a 9000 module.

Syntax:

ClearEventCounter(WORD wIPBUF[], float fOPBUF[], char szSendTo9000[], char szReceiveFrom9000[])

Input Parameter:

w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000

Return Value:

NoError: OK
others: Error code, refer to EX9000.H .

W9000: WORD Input/Output Table

wIPBUF[0]: 1 to 255 for RS-232 port number
wIPBUF[1]: 00 to FF for module address
wIPBUF[2]: module ID for 0x9011/0x9012/0x9014
wIPBUF[3]: 0=disable , 1=enable for checksum
wIPBUF[4]: normal=100 for TimeOut constant
wIPBUF[5]: Null & void
wIPBUF[6]: 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000

F9000: Float Input/Output Table

Null & void

2.8 Dual Watchdog Functions

2.8.1 HostIsOK

Description:

To tell all modules “Host is OK” by send this command “~**”.

Syntax:

WORD HostIsOK(WORD wIPBUF[], float fOPBUF[],
char szSendTo9000 [], char szReceiveFrom9000 [])

Input Parameter:

w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000

Return Value:

NoError: OK
others: Error code, refer to EX9000.H .

W9000: WORD Input/Output Table

wIPBUF[0]: 1 to 255 for RS-232 port number
wIPBUF[1]: Null & void
wIPBUF[2]: Null & void
wIPBUF[3]: 0=disable , 1=enable for checksum
wIPBUF[4]: normal=100 for TimeOut constant
wIPBUF[5]: Null & void
wIPBUF[6]: 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000

F9000: Float Input/Output Table

Null & void

2.8.2 ReadModuleResetStatus

Description:
To read the module reset status.
Syntax:
WORD ReadModuleResetStatus(WORD wIPBUF[], float fOPBUF[],
char szSendTo9000[], char szReceiveFrom9000[])
Input Parameter:
w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000
Return Value:
NoError: OK
others: Error code, refer to EX9000.H .
W9000: WORD Input/Output Table
wIPBUF[0]: 1 to 255 for RS-232 port number
wIPBUF[1]: 00 to FF for module address
wIPBUF[2]: Module ID for 0x9011, 0x9012 …….
wIPBUF[3]: 0=disable , 1=enable for checksum
wIPBUF[4]: normal=100 for TimeOut constant
wIPBUF[6]: 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000
------- output -----
wIPBUF[5]: 0: module has not been reset
1: module has been reset
F9000: Float Input/Output Table
Null & Null & void

2.8.3 ToSetupHostWatchdog

Description:
To setup the module's Host Watchdog.
Syntax:
WORD ToSetupHostWatchdog(WORD wIPBUF[], float fOPBUF[],
char szSendTo9000[], char szReceiveFrom9000[])
Input Parameter:
w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000
Return Value:
NoError: OK
others: Error code, refer to EX9000.H .
W9000: WORD Input/Output Table
wIPBUF[0]: 1 to 255 for RS-232 port number
wIPBUF[1]: 00 to FF for module address
wIPBUF[2]: Null & void
wIPBUF[3]: 0=disable , 1=enable for checksum
wIPBUF[4]: normal=100 for TimeOut constant
wIPBUF[5]: 0: Disable host watchdog
1: Enable host watchdog
wIPBUF[6]: 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000
wIPBUF[7]: a time interval for watchdog, unit is 0.1 second,
e.x when wIPBUF[7]=45, the time interval is 4.5 second
F9000: Float Input/Output Table

2.8.4 ToReadHostWatchdog

Description:
To read the module's Host Watchdog setup values.
Syntax:
WORD ToReadHostWatchdog(WORD wIPBUF[], float fOPBUF[],
char szSendTo9000[], char szReceiveFrom9000[])
Input Parameter:
w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000
Return Value:
NoError: OK
others: Error code, refer to EX9000.H .
W9000: WORD Input/Output Table
wIPBUF[0]: 1 to 255 for RS-232 port number
wIPBUF[1]: 00 to FF for module address
wIPBUF[2]: Null & void
wIPBUF[3]: 0=disable , 1=enable for checksum
wIPBUF[4]: normal=100 for TimeOut constant
wIPBUF[6]: 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000
---- output ----
wIPBUF[5]: 0: Host watchdog is disabled.
1: Host watchdog is enabled.
wIPBUF[7]: a time interval for watchdog, unit is 0.1 second,
e.x when wIPBUF[7]=45, the time interval is 4.5 second
F9000: Float Input/Output Table

2.8.5 ReadModuleHostWatchdogStatus

Description:
To read the module's Host Watchdog status.
Syntax:
WORD ReadModuleHostWatchdogStatus(WORD wIPBUF[], float fOPBUF[], char szSendTo9000[], char szReceiveFrom9000[])
Input Parameter:
w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000
Return Value:
NoError: OK
others: Error code, refer to EX9000.H .
W9000: WORD Input/Output Table
wIPBUF[0]: 1 to 255 for RS-232 port number
wIPBUF[1]: 00 to FF for module address
wIPBUF[2]: Null & void
wIPBUF[3]: 0=disable , 1=enable for checksum
wIPBUF[4]: normal=100 for TimeOut constant
wIPBUF[6]: 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000
---- output ----
wIPBUF[5]: 0: Module's Host watchdog is in NORMAL mode.
4: Module's Host watchdog is in HOST FAILURE mode.
F9000: Float Input/Output Table

2.8.6 ResetModuleHostWatchdogStatus

Description:
To reset the module's Host Watchdog status.
Syntax:
WORD ReadModuleHostWatchdogStatus(WORD wIPBUF[], float fOPBUF[], char szSendTo9000[], char szReceiveFrom9000[])
Input Parameter:
w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000
Return Value:
NoError: OK
others: Error code, refer to EX9000.H .
W9000: WORD Input/Output Table
wIPBUF[0]: 1 to 255 for RS-232 port number
wIPBUF[1]: 00 to FF for module address
wIPBUF[2]: Null & void
wIPBUF[3]: 0=disable , 1=enable for checksum
wIPBUF[4]: normal=100 for TimeOut constant
wIPBUF[5]: Null & void
wIPBUF[6]: 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000
F9000: Float Input/Output Table

2.8.7 SetSafeValueForDo

Description:
To setup the safe value for DO modules.
Syntax:
WORD SetSafeValueForDo(WORD wIPBUF[], float fOPBUF[],
char szSendTo9000[], char szReceiveFrom9000[])
Input Parameter:
w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000
Return Value:
NoError: OK
others: Error code, refer to EX9000.H .
W9000: WORD Input/Output Table
wIPBUF[0]: 1 to 255 for RS-232 port number
wIPBUF[1]: 00 to FF for module address
wIPBUF[2]: module ID for 0x9050/60/67/42/43/44
wIPBUF[3]: 0=disable , 1=enable for checksum
wIPBUF[4]: normal=100 for TimeOut constant
wIPBUF[5]: safe value
wIPBUF[6]: 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000
F9000: Float Input/Output Table

2.8.8 SetPowerOnValueForDo

Description:
To setup the power on value for DO modules.
Syntax:
WORD SetPowerOnValueForDo(WORD wIPBUF[], float fOPBUF[],
char szSendTo9000[], char szReceiveFrom9000[])
Input Parameter:
w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000
Return Value:
NoError: OK
others: Error code, refer to EX9000.H .
W9000: WORD Input/Output Table
wIPBUF[0]: 1 to 255 for RS-232 port number
wIPBUF[1]: 00 to FF for module address
wIPBUF[2]: module ID for 0x9050/60/67/42/43/44
wIPBUF[3]: 0=disable , 1=enable for checksum
wIPBUF[4]: normal=100 for TimeOut constant
wIPBUF[5]: PowerOn value
wIPBUF[6]: 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000
F9000: Float Input/Output Table

2.8.9 SetSafeValueForAo

Description:
To setup the safe value for AO modules.
Syntax:
WORD SetSafeValueForAo(WORD wIPBUF[], float fOPBUF[],
char szSendTo9000[], char szReceiveFrom9000[])
Input Parameter:
w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000
Return Value:
NoError: OK
others: Error code, refer to EX9000.H .
W9000: WORD Input/Output Table
wIPBUF[0]: 1 to 255 for RS-232 port number
wIPBUF[1]: 00 to FF for module address
wIPBUF[2]: module ID for 0x9021/0x9024
wIPBUF[3]: 0=disable , 1=enable for checksum
wIPBUF[4]: normal=100 for TimeOut constant
wIPBUF[5]: Channel No
wIPBUF[6]: 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000
F9000: Float Input/Output Table
fOPBUF[0]: safe value

2.8.10 SetPowerOnValueForAo

Description:
To setup the power on value for AO modules.
Syntax:
WORD SetPowerOnValueForAo (WORD wIPBUF[], float fOPBUF[],
char szSendTo9000[], char szReceiveFrom9000[])
Input Parameter:
w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000
Return Value:
NoError: OK
others: Error code, refer to EX9000.H .
W9000: WORD Input/Output Table
wIPBUF[0]: 1 to 255 for RS-232 port number
wIPBUF[1]: 00 to FF for module address
wIPBUF[2]: module ID for 0x9021/0x9024
wIPBUF[3]: 0=disable , 1=enable for checksum
wIPBUF[4]: normal=100 for TimeOut constant
wIPBUF[5]: Channel No
wIPBUF[6]: 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000
F9000: Float Input/Output Table
fOPBUF[0]: Power On value

2.8.11 SetPowerOnSafeValue

Description:
To setup the power on and safe value for modules.
Syntax:
WORD SetPowerOnSafeValue(WORD wIPBUF[], float fOPBUF[],
char szSendTo9000[], char szReceiveFrom9000[])
Input Parameter:
w9000: WORD Input/Output argument table
f9000: float Input/Output argument table
szSendTo9000: command string send to 9000
szReceiveFrom9000: result string receive from 9000
Return Value:
NoError: OK
others: Error code, refer to EX9000.H .
W9000: WORD Input/Output Table
wIPBUF[0]: 1 to 255 for RS-232 port number
wIPBUF[1]: 00 to FF for module address
wIPBUF[2]: module ID for 0x9011/0x9012/0x9014
wIPBUF[3]: 0=disable , 1=enable for checksum
wIPBUF[4]: normal=100 for TimeOut constant.

wIPBUF[5]: PowerOn value
wIPBUF[6]: 0 ! no save to szSendTo9000&szReceiveFrom9000
1 ! szSendTo9000=command string send to 9000
! szReceiveFrom9000=result string receive from 9000
wIPBUF[7]: safe value
F9000: Float Input/Output Table