ScanKernel API

Programmer's Manual

REV 1.02

2004/04/05

1. ScanKernel API

The "ScanKernel API" is a set of application interface for manipulating the "ScanKernel.exe" built in WinCon-8000. Users can easily use the "ScanKernel API" in their program to manipulate the "ScanKernel.exe" so that accessing the I/O value of I7K/I8K/I87K modules plugged in or connected with WinCon-8000. Besides, by launching into different operating mode of "ScanKernel.exe", users also can access the embedded I8K/I87K and remote modules via different communication port.

The "ScanKernel API" comprises five groups which are "System Function", "I8K/87K Function", "Remote Function", "Modbus Function", and "User Shared Function". The first group, "System Function", provides two functions for users to start and stop the "ScanKernel.exe". The second group, "I8K/87K Function", provides thirteen functions for users to access the I/O values of I8K/87K modules which plugged in the WinCon-8000. The third group, "Remote Function", provides eleven functions for users to access the I/O values of remote I7K/I87K modules. The fourth group, "Modbus Function", provides four functions for users to add their own variables into modbus server for sharing the values to modbus client. The fifth group, "User Shared Function", provides four functions for users to add their own variables into share memory block for sharing the values with different application program on the same WinCon-8000.

Users have to use the function "StartAgent()" in the "System Function" to start the "ScanKernel.exe" before using both "I8K/87K Function" and "Remote Function". And call the function "StopAgent()" while users want to terminate the "ScanKernel.exe". **If users just want to use "Modbus Function" and "User Shared Function", they do not need to call the "System Function".**

Note:

1. **ScanKernel.dll** and **eVC++** application program must be copied to the same folder in the WinCON-8000
2. **ScanKernel.dll, ScanKernelNet.dll,** and **VB.NET/VC#.NET** application program must be copied to the same folder in the WinCON-8000

1.1 ScanKernel API For eVC++ developer

Step 1:

Create a new eVC++ project with choosing “Win32[WCE ARMV4] CPU” option

Step 2:

#include "WinConAgent.h"

Step 3:

Refer to the following functions to design your own program

Step 4:

Build your project with release mode.

Note: ScanKernel.dll and eVC++ application program must be copied to the same folder in the WinCON-8000

System Function
unsigned char StartAgent(unsigned char iMode, int ComPara[][6])
unsigned char StopAgent(void)

I8K/87K Function
unsigned char GetNameOf8KModule(unsigned char iSlot, char *cName)
unsigned char GetNameOf87KModule(unsigned char iSlot, char *cName)
unsigned char GetAtriOf8KModule(unsigned char iSlot, unsigned char *iAtri)
unsigned char GetAtriOf87KModule(unsigned char iSlot, unsigned char *iAtri)
unsigned char GetDIO(unsigned char iSlot, unsigned char iChannel, unsigned char *iRecv, unsigned char iAttribute)
unsigned char GetAIO(unsigned char iSlot, unsigned char iChannel, float *iRecv, unsigned char iAttribute)
unsigned char GetMultiDIO(unsigned char iSlot, unsigned char *iRecv, unsigned char iAttribute)
unsigned char GetMultiAIO(unsigned char iSlot, float *iRecv, unsigned char iAttribute)
unsigned char SetDO(unsigned char iSlot, unsigned char iChannel, unsigned char iSend)
unsigned char SetAO(unsigned char iSlot, unsigned char iChannel, float iSend)
unsigned char SetMultiDO(unsigned char iSlot, unsigned char *iSend)
unsigned char SetMultiAO(unsigned char iSlot, float *iSend)
unsigned char Get87KCountVal(unsigned char iSlot, unsigned char iChannel, unsigned short *iRecv)

Remote Function
unsigned char GetNameOfRemoteModule(unsigned char iCOM, unsigned char iSlave, char *cName)
unsigned char GetAtriOfRemoteModule(unsigned char iCOM, unsigned char iSlave, unsigned char *iAtri)
unsigned char GetRemoteDIO(unsigned char iCOM, unsigned char iSlave, unsigned char iChannel, unsigned char *iRecv, unsigned char iAttribute)
unsigned char GetRemoteAIO(unsigned char iCOM, unsigned char iSlave, unsigned char iChannel, float *iRecv, unsigned char iAttribute)
unsigned char GetRemoteMultiDIO(unsigned char iCOM, unsigned char iSlave, unsigned char *iRecv, unsigned char iAttribute)
unsigned char GetRemoteMultiAIO(unsigned char iCOM, unsigned char iSlave, float *iRecv, unsigned char iAttribute)
unsigned char SetRemoteDO(unsigned char iCOM, unsigned char iSlave, unsigned char iChannel, unsigned char iSend)
unsigned char SetRemoteAO(unsigned char iCOM, unsigned char iSlave, unsigned char iChannel, float iSend)
unsigned char SetRemoteMultiDO(unsigned char iCOM, unsigned char iSlave, unsigned char *iSend)
unsigned char SetRemoteMultiAO(unsigned char iCOM, unsigned char iSlave, float *iSend)
unsigned char GetRemoteCountVal(unsigned char iCOM, unsigned char iSlave, unsigned char iChannel, unsigned short *iRecv)

Modbus Function
unsigned char MBSetToCoil(unsigned short iMBAddress, unsigned char iStatus, unsigned char iAttr) unsigned char MBGetFromCoil(unsigned short iMBAddress, unsigned char *iStatus, unsigned char iAttr) unsigned char MBSetToReg(unsigned short iMBAddress, short iStatus, unsigned char iAttr) unsigned char MBGetFromReg(unsigned short iMBAddress, short *iStatus, unsigned char iAttr)

User Shared Function
unsigned char UserSetCoil(unsigned short iUserAddress, unsigned char iStatus) unsigned char UserGetCoil(unsigned short iUserAddress, unsigned char *iStatus) unsigned char UserSetReg(unsigned short iUserAddress, short iStatus) unsigned char UserGetReg(unsigned short iUserAddress, short *iStatus)

1.1.1 System Function

This group provides two functions for users to start and stop the "ScanKernel.exe" before using "I8K/87K Function" and "Remote Function".

StartAgent

This function launches the scan kernel with different mode.

unsigned char StartAgent(
unsigned char iMode,
int ComPara[][6]
)

Parameters

iMode

[in] The decimal number of kernel mode.

Bit of COM2 thread	Bit of COM1 thread	Bit of embedded I87K thread	Bit of embedded I8K thread

Example:

0	0	1	1	means decimal number **3** – Launch embedded I8K and I87K thread
1	0	1	1	means decimal number **11** – Launch COM2 , embedded I8K and I87K thread

ComPara[][6]

[in] The matrix of parameter for com port. For ComPara[**a**][**b**], the definition of **a** and **b** are as below.

a -0 *COM1*; **-1** *COM2*; **-2** *COM3*

b -0 *Baud rate*; **-1** *Parity*; **-2** *Data bit*; **-3** *Stop bit*; **-4** *Checksum*; **-5** *Timeout*;

Baud rate

[in] The baud rate of COM port which should be equal to the modules.

Parity

[in] Specifies the parity scheme to be used. It is one of the following values.

Value	Description
0	No parity
1	Odd
2	Even

Data bit

[in] Specifies the number of bits in the bytes transmitted and received.

Stop bit

[in] Specifies the number of stop bits to be used. It is one of the following values.

Value	Description
0	1 stop bit
1	1.5 stop bits
2	2 stop bits

Checksum

[in] 0 means FALSE;1 means TRUE.

Timeout

[in] Specifies the timeout (Response time) value for communication.

Return Values

0 indicates success. If the ScanKernel has been run, the function will return mode number. (Please refer to the Appendix 2.1)

Remarks

You **have to** call this function to launch the ScanKernel before using the I8K/I87K and remote functions.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Start up the ScanKernel with mode 11(COM2, embedded I8K and I87K thread)
int ComPara[2][6]={0};
ComPara[1][0] = 9600;   //Baudrate
ComPara[1][1] = 0;      //Parity
ComPara[1][2] = 8;      //Data bits
ComPara[1][3] = 0;      //Stop bits
ComPara[1][4] = 0;      //Checksum
ComPara[1][5] = 100;    //Timeout
if (StartAgent(11, ComPara) == 0){
    AfxMessageBox(_T("Start agent successfully!"));
}
else{
    AfxMessageBox(_T("Agent has been started!"));
}
```

StopAgent

This function stops the scan kernel.

unsigned char StopAgent(
void
)

Parameters

Return Values

0 indicates success. **WCA_Stop** means ScanKernel has been stopped. **WCA_NOT_MASTER** means not the main AP which calls ScanKernel (Please refer to the Appendix 2.1)

Remarks

ScanKernel only can be stopped by the AP which launched it.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Stop the ScanKernel
if(StopAgent() == 0){
	AfxMessageBox(_T("Stop agent successfully!"));
}
else if(StopAgent() == WCA_Stop){
	AfxMessageBox(_T("ScanKernel has been stopped!"));
}
else{
	AfxMessageBox(_T("Can not terminate the ScanKernel!"));
}
```

1.1.2 I8K/87K Function

This group provides thirteen functions for users to access the I/O values of I8K/87K modules which plugged in the WinCon-8000.

GetNameOf8KModule

This function can get the name of I8K module which you plugged in specific slot.

unsigned char GetNameOf8KModule(
unsigned char iSlot,
char *cName
)

Parameters

iSlot

[in] The slot number of which I8K module you want to get. The range of slot number is from 1 to 7.

cName

[out] The pointer to a char array that will hold the module name. The array size must be 80 bytes.

Return Values

0 indicates success. **WCA_SLOTNO_OVER** means the iSlot over the range. The legal range is from number 1 to number 7.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Get the I8K module name which is plugged in slot 1
char cName[80]={0};
GetNameOf8KModule(1, cName);
```

GetNameOf87KModule

This function can get the name of I87K module which you plugged in specific slot.

unsigned char GetNameOf87KModule(
unsigned char iSlot,
char *cName
)

Parameters

iSlot

[in] The slot number of which I87K module you want to get. The range of slot number is from 1 to 7.

cName

[out] The pointer to a char array that will hold the module name. The array size must be 80 bytes.

Return Values

0 indicates success. **WCA_SLOTNO_OVER** means the iSlot over the range. The legal range is from number 1 to number 7.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Get the I87K module name which is plugged in slot 1
char cName[80]={0};
GetNameOf87KModule(1, cName);
```

GetAtriOf8KModule

This function gets the attributes of I8K module.

unsigned char GetAtriOf8KModule(
unsigned char iSlot,
unsigned char *iAtri
)

Parameters

iSlot

[in] The slot number of which I8K module you want to get. The range of slot number is from 1 to 7.

iAtri

[out] The pointer to an unsigned char array that will hold the module attibutes. The array size must be 8 bytes. The definition of each element is as bellows.

Array element	Description
iAtri[0]	The number of DI
iAtri[1]	The number of DO
iAtri[2]	The number of AI
iAtri[3]	The number of AO
iAtri[4]	The number of counter
iAtri[5]	None
iAtri[6]	None
iAtri[7]	None

Return Values

0 indicates success. **WCA_SLOTNO_OVER** means the iSlot over the range. The legal range is from number 1 to number 7.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
// Get the I8K module attributes plugged in slot 1
unsigned char Atr8K[8]={0};
GetAtriOf8KModule(1, Atr8K);
```

GetAtriOf87KModule

This function gets the attributes of I87K module.

unsigned char GetAtriOf87KModule(
unsigned char iSlot,
unsigned char *iAtri
)

Parameters

iSlot

[in] The slot number of which I87K module you want to get. The range of slot number is from 1 to 7.

iAtri

[out] The pointer to an unsigned char array that will hold the module attibutes. The array size must be 8 bytes. The definition of each element is as bellows.

Array element	Description
iAtri[0]	The number of DI
iAtri[1]	The number of DO
iAtri[2]	The number of AI
iAtri[3]	The number of AO
iAtri[4]	The number of counter
iAtri[5]	None
iAtri[6]	None
iAtri[7]	None

Return Values

0 indicates success. **WCA_SLOTNO_OVER** means the iSlot over the range. The legal range is from number 1 to number 7.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Get the I87K module attributes plugged in slot 1
unsigned char Atr87K[8]={0};
GetAtriOf87KModule(1, Atr8K);
```

GetDIO

This function can get a single digital I/O status from a specific slot and channel.

unsigned char GetDIO(
unsigned char iSlot,
unsigned char iChannel,
unsigned char *iRecv,
unsigned char iAttribute
)

Parameters

iSlot

[in] The slot number of which module you want to get. The range of slot number is from 1 to 7.

iChannel

[in] The channel number of specific module.

iRecv

[out] The digital status of specific channel. 1 means ON. 0 means OFF.

iAttribute

[in] Assign which kind of digital status you want get. 1 means digital input. 0 means digital output.

Return Values

0 indicates success. **WCA_SLOTNO_OVER** means the iSlot over the range. The legal range is from number 1 to number 7. **WCA_ATT_ERROR** means the iAttibute is neither 0 nor 1.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Get the digital I/O status
//Get the digital input status from slot1/channel1
unsigned char iRecvIn;
GetDIO(1,1,&iRecvIn,1);
//Get the digital output status from slot1/channel1
unsigned char iRecvOut;
GetDIO(1,1,&iRecvOut,0);
```

GetAIO

This function can get a single analog I/O value from a specific slot and channel.

unsigned char GetAIO(

unsigned char iSlot,

unsigned char iChannel,

float *iRecv,

unsigned char iAttribute

)

Parameters

iSlot

[in] The slot number of which module you want to get. The range of slot number is from 1 to 7.

iChannel

[in] The channel number of specific module.

iRecv

[out] The analog value of specific channel.

iAttribute

[in] Assign which kind of analog value you want get.

Return Values

0 indicates success. **WCA_SLOTNO_OVER** means the iSlot over the range. The legal range is from number 1 to number 7. **WCA_ATT_ERROR** means the iAttibute is neither 0 nor 1.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Get the analog I/O value
//Get the analog input value from slot1/channel1
float fRecvIn;
GetAIO(1,1,&fRecvIn,1);
//Get the analog output value from slot1/channel1
float fRecvOut;
GetAIO(1,1,&fRecvOut,0);
```

GetMultiDIO

This function can get multiple digital I/O status from a specific slot.

unsigned char GetMultiDIO(
unsigned char iSlot,
unsigned char *iRecv,
unsigned char iAttribute
)

Parameters

iSlot

[in] The slot number of which module you want to get. The range of slot number is from 1 to 7.

iRecv

[out] The pointer to an unsigned char array that will hold the digital statuses. The array size must be 32 bytes. iRecv[0] is the status of channel 0, iRecv[1] is the status of channel 1 and so on.

iAttribute

[in] Assign which kind of digital status you want get. 1 means digital input. 0 means digital output.

Return Values

0 indicates success. **WCA_SLOTNO_OVER** means the iSlot over the range. The legal range is from number 1 to number 7. **WCA_ATT_ERROR** means the iAttibute is neither 0 nor 1.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Get the multiple digital I/O statuses
//Get the digital input statuses from slot1
unsigned char iRecvIn[32];
GetMultiDIO(1,iRecvIn,1);
//Get the digital output statuses from slot1
unsigned char iRecvOut[32];
GetMultiDIO(1,iRecvOut,0);
```

GetMultiAIO

This function can get multiple analog I/O values from a specific slot.

unsigned char GetMultiAIO(
unsigned char iSlot,
float *iRecv,
unsigned char iAttribute
)

Parameters

iSlot

[in] The slot number of which module you want to get. The range of slot number is from 1 to 7.

iRecv

[out] The pointer to an unsigned char array that will hold the analog values. The array size must be 128 bytes. iRecv[0] is the value of channel 0, iRecv[1] is the value of channel 1 and so on.

iAttribute

[in] Assign which kind of analog value you want get. 1 means analog input. 0 means analog output.

Return Values

0 indicates success. **WCA_SLOTNO_OVER** means the iSlot over the range. The legal range is from number 1 to number 7. **WCA_ATT_ERROR** means the iAttibute is neither 0 nor 1.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Get the multiple analog I/O values
//Get the analog input values from slot1
float iRecvIn[32];
GetMultiAIO(1,iRecvIn,1);
//Get the analog output values from slot1
float iRecvOut[32];
GetMultiAIO(1,iRecvOut,0);
```

SetDO

This function can set a single digital output status to a specific slot and channel

unsigned char SetDO(
unsigned char iSlot,
unsigned char iChannel,
unsigned char iSend
)

Parameters

iSlot

[in] The slot number of which module you want to set. The range of slot number is from 1 to 7.

iChannel

[in] The channel number of specific module.

iSend

[in] The digital status of specific channel. 1 means ON. 0 means OFF.

Return Values

0 indicates success. **WCA_SLOTNO_OVER** means the iSlot over the range. The legal range is from number 1 to number 7.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

//Set the digital output ON to slot1/channel1
SetDO(1,1,1);

SetAO

This function can set a single analog output value to a specific slot and channel

unsigned char SetAO(
unsigned char iSlot,
unsigned char iChannel,
float iSend
)

Parameters

iSlot

[in] The slot number of which module you want to set. The range of slot number is from 1 to 7.

iChannel

[in] The channel number of specific module.

iSend

[in] The analog value of specific channel.

Return Values

0 indicates success. **WCA_SLOTNO_OVER** means the iSlot over the range. The legal range is from number 1 to number 7.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

//Set the analog output value as 5.5 to slot1/channel1

SetAO(1,1,5.5);

SetMultiDO

This function can set multiple digital output statuses to a specific slot.

unsigned char SetMultiDO(
unsigned char iSlot,
unsigned char *iSend
)

Parameters

iSlot

[in] The slot number of which module you want to set. The range of slot number is from 1 to 7.

iSend

[in] The pointer to an unsigned char array that will hold the multiple digital output statuses. The maximum elements of the array is 32. The array element 1 means ON, 0 means OFF.

Return Values

0 indicates success. **WCA_SLOTNO_OVER** means the iSlot over the range. The legal range is from number 1 to number 7.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Set the digital output channel 0 to channel 7 to slot1
unsigned char iSend[8]={1,1,1,1,1,1,0,1};
SetMultiDO(1, iSend);
```

SetMultiAO

This function can set multiple analog output values to a specific slot.

unsigned char SetMultiAO(
unsigned char iSlot,
float *iSend
)

Parameters

iSlot

[in] The slot number of which module you want to set. The range of slot number is from 1 to 7.

iSend

[in] The pointer to an float array that will hold the multiple analog output values. The maximum elements of the array is 32.

Return Values

0 indicates success. **WCA_SLOTNO_OVER** means the iSlot over the range. The legal range is from number 1 to number 7.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Set the analog output channel 0 to channel 3 to slot1
float iSend[4]={(float)3.6,(float)3,(float)-4,(float)0};
SetMultiAO(1, iSend);
```

Get87KCountVal

Available soon.

unsigned char Get87KCountVal(
unsigned char iSlot,
unsigned char iChannel,
unsigned short *iRecv
)

Parameters

Return Values

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

1.1.3 Remote Function

This group provides eleven functions for users to access the I/O values of remote I7K/I87K modules.

GetNameOfRemoteModule

This function can get the name of remote module via specific com port.

unsigned char GetNameOfRemoteModule(
unsigned char iCOM,
unsigned char iSlave,
char *cName
)

Parameters

iCOM

[in] The com port number of which remote modules you want to access through. The range of port number is from 2 to 3.

iSlave

[in] The slave number of remote module you want to get. The range of slave number is from 1 to 256.

cName

[out] The pointer to a char array that will hold the module name. The array size must be 80 bytes.

Return Values

0 indicates success. **WCA_COMNO_OVER** means the iCOM over the range. The legal range is from number 2 to number 3.**WCA_SLAVENO_OVER** means the iSlave over the range. The legal range is from number 1 to number 256.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Get the remote module name which slave number is 5 and connecting via COM2
char cName[80]={0};
GetNameOfRemoteModule(2, 5, cName);
```

GetAtriOfRemoteModule

This function gets the attributes of remote module.

unsigned char GetAtriOfRemoteModule(
unsigned char iCOM,
unsigned char iSlave,
unsigned char *iAtri
)

Parameters

iCOM

[in] The com port number of which remote modules you want to access through. The range of port number is from 2 to 3.

iSlave

[in] The slave number of remote module you want to get. The range of slave number is from 1 to 256.

iAtri

[out] The pointer to an unsigned char array that will hold the module attibutes. The array size must be 8 bytes. The definition of each element is as bellows.

Array element	Description
iAtri[0]	The number of DI
iAtri[1]	The number of DO
iAtri[2]	The number of AI
iAtri[3]	The number of AO
iAtri[4]	The number of counter
iAtri[5]	None
iAtri[6]	None
iAtri[7]	None

Return Values

0 indicates success. **WCA_COMNO_OVER** means the iCOM over the range. The legal range is from number 2 to number 3.**WCA_SLAVENO_OVER** means the iSlave over the range. The legal range is from number 1 to number 256.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Get the remote module attributes which slave number is 5 and connecting via COM2
unsigned char AtrRemote[8]={0};
GetAtriOfRemoteModule(2, 5, AtrRemote);
```

GetRemoteDIO

This function can get a single digital I/O status from a specific remote module and channel.

unsigned char GetRemoteDIO(
unsigned char iCOM,
unsigned char iSlave,
unsigned char iChannel,
unsigned char *iRecv,
unsigned char iAttribute
)

Parameters

iCOM

[in] The com port number of which remote modules you want to access through. The range of port number is from 2 to 3.

iSlave

[in] The slave number of remote module you want to get. The range of slave number is from 1 to 256.

iChannel

[in] The channel number of specific module.

iRecv

[out] The digital status of specific channel. 1 means ON. 0 means OFF.

iAttribute

[in] Assign which kind of digital status you want get. 1 means digital input. 0 means digital output.

Return Values

0 indicates success. **WCA_COMNO_OVER** means the iCOM over the range. The legal range is from number 2 to number 3.**WCA_SLAVENO_OVER** means the iSlave over the range. The legal range is from number 1 to number 256. **WCA_ATT_ERROR** means the iAttibute is neither 0 nor 1.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Get the digital I/O status
//Get the digital input status from slave5/channel1 via COM2
unsigned char iRecvIn;
GetRemoteDIO(2,5,1,&iRecvIn,1);
//Get the digital output status from slave5/channel2 via COM2
unsigned char iRecvOut;
GetRemoteDIO(2,5,2,&iRecvOut,0);
```

GetRemoteAIO

This function can get a single analog I/O value from a specific remote module and channel.

unsigned char GetRemoteAIO(
unsigned char iCOM,
unsigned char iSlave,
unsigned char iChannel,
float *iRecv,
unsigned char iAttribute
)

Parameters

iCOM

[in] The com port number of which remote modules you want to access through. The range of port number is from 2 to 3.

iSlave

[in] The slave number of remote module you want to get. The range of slave number is from 1 to 256.

iChannel

[in] The channel number of specific module.

iRecv

[out] The analog value of specific channel.

iAttribute

[in] Assign which kind of analog value you want get.

Return Values

0 indicates success. **WCA_COMNO_OVER** means the iCOM over the range. The legal range is from number 2 to number 3.**WCA_SLAVENO_OVER** means the iSlave over the range. The legal range is from number 1 to number 256. **WCA_ATT_ERROR** means the iAttibute is neither 0 nor 1.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Get the analog I/O value
//Get the analog input value from slave5/channel1 via COM2
float fRecvIn;
GetRemoteAIO(2,5,1,&fRecvIn,1);
//Get the analog output value from slave5/channel2 via COM2
float fRecvOut;
GetRemoteAIO(2,5,2,&fRecvOut,0);
```

GetRemoteMultiDIO

This function can get multiple digital I/O status from a specific remote module.

unsigned char GetRemoteMultiDIO(
unsigned char iCOM,
unsigned char iSlave,
unsigned char *iRecv,
unsigned char iAttribute
)

Parameters

iCOM

[in] The com port number of which remote modules you want to access through. The range of port number is from 2 to 3.

iSlave

[in] The slave number of remote module you want to get. The range of slave number is from 1 to 256.

iRecv

[out] The pointer to an unsigned char array that will hold the digital statuses. The array size must be 32 bytes. iRecv[0] is the status of channel 0, iRecv[1] is the status of channel 1 and so on.

iAttribute

[in] Assign which kind of digital status you want get. 1 means digital input. 0 means digital output.

Return Values

0 indicates success. **WCA_COMNO_OVER** means the iCOM over the range. The legal range is from number 2 to number 3.**WCA_SLAVENO_OVER** means the iSlave over the range. The legal range is from number 1 to number 256.**WCA_ATT_ERROR** means the iAttibute is neither 0 nor 1.

Remarks

Requirements

Runs on	**Versions**	**Defined in**	**Include**	**Link to**
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Get the multiple digital I/O statuses
//Get the digital input statuses from slave1 via COM2
unsigned char iRecvIn[32];
GetRemoteMultiDIO(2,1,iRecvIn,1);
//Get the digital output statuses from slave1 via COM2
unsigned char iRecvOut[32];
GetRemoteMultiDIO(2,1,iRecvOut,0);
```

GetRemoteMultiAIO

This function can get multiple analog I/O values from a specific remote module.

unsigned char GetRemoteMultiAIO(
unsigned char iCOM,
unsigned char iSlave,
float *iRecv,
unsigned char iAttribute
)

Parameters

iCOM

[in] The com port number of which remote modules you want to access through. The range of port number is from 2 to 3.

iSlave

[in] The slave number of remote module you want to get. The range of slave number is from 1 to 256.

iRecv

[out] The pointer to an unsigned char array that will hold the analog values. The array size must be 128 bytes. iRecv[0] is the value of channel 0, iRecv[1] is the value of channel 1 and so on.

iAttribute

[in] Assign which kind of analog value you want get. 1 means analog input. 0 means analog output

Return Values

0 indicates success. **WCA_COMNO_OVER** means the iCOM over the range. The legal range is from number 2 to number 3.**WCA_SLAVENO_OVER** means the iSlave over the range. The legal range is from number 1 to number 256.**WCA_ATT_ERROR** means the iAttibute is neither 0 nor 1.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Get the multiple analog I/O values
//Get the analog input values from slave1 via COM2
float iRecvIn[32];
GetRemoteMultiAIO(2,1,iRecvIn,1);
//Get the analog output values from slave1 via COM2
float iRecvOut[32];
GetRemoteMultiAIO(2,1,iRecvOut,0);
```

SetRemoteDO

This function can set a single digital output status to a specific remote module and channel.

unsigned char SetRemoteDO(
unsigned char iCOM,
unsigned char iSlave,
unsigned char iChannel,
unsigned char iSend
)

Parameters

iCOM

[in] The com port number of which remote modules you want to access through. The range of port number is from 2 to 3.

iSlave

[in] The slave number of remote module you want to get. The range of slave number is from 1 to 256.

iChannel

[in] The channel number of specific module.

iSend

[in] The digital status of specific channel. 1 means ON. 0 means OFF.

Return Values

0 indicates success. **WCA_COMNO_OVER** means the iCOM over the range. The legal range is from number 2 to number 3.**WCA_SLAVENO_OVER** means the iSlave over the range. The legal range is from number 1 to number 256.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Set the digital output ON to slave1/channel1 via COM2
SetRemoteDO(2,1,1,1);
```

SetRemoteAO

This function can set a single analog output value to a specific remote module and channel.

unsigned char SetRemoteAO(
unsigned char iCOM,
unsigned char iSlave,
unsigned char iChannel,
float iSend
)

Parameters

iCOM

[in] The com port number of which remote modules you want to access through. The range of port number is from 2 to 3.

iSlave

[in] The slave number of remote module you want to get. The range of slave number is from 1 to 256.

iChannel

[in] The channel number of specific module.

iSend

[in] The analog value of specific channel.

Return Values

0 indicates success. **WCA_COMNO_OVER** means the iCOM over the range. The legal range is from number 2 to number 3.**WCA_SLAVENO_OVER** means the iSlave over the range. The legal range is from number 1 to number 256.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Set the analog output value as 5.5 to slave1/channel1 via COM2
SetRemoteAO(2,1,1,5.5);
```

SetRemoteMultiDO

This function can set multiple digital output statuses to a specific remote module.

```
unsigned char SetRemoteMultiDO(
unsigned char iCOM,
unsigned char iSlave,
unsigned char *iSend
)
```

Parameters

iCOM

[in] The com port number of which remote modules you want to access through. The range of port number is from 2 to 3.

iSlave

[in] The slave number of remote module you want to get. The range of slave number is from 1 to 256.

iSend

[in] The pointer to an unsigned char array that will hold the multiple digital output statuses. The maximum elements of the array is 32. The array element 1 means ON, 0 means OFF.

Return Values

0 indicates success. **WCA_COMNO_OVER** means the iCOM over the range. The legal range is from number 2 to number 3.**WCA_SLAVENO_OVER** means the iSlave over the range. The legal range is from number 1 to number 256.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Set the digital output channel 0 to channel 7 to slave1 via COM2
unsigned char iSend[8]={1,1,1,1,1,1,0,1};
SetRemoteMultiDO(2, 1, iSend);
```

SetRemoteMultiAO

This function can set multiple analog output values to a specific remote module.

unsigned char SetRemoteMultiAO(
unsigned char iCOM,
unsigned char iSlave,
float *iSend
)

Parameters

iCOM

[in] The com port number of which remote modules you want to access through. The range of port number is from 2 to 3.

iSlave

[in] The slave number of remote module you want to get. The range of slave number is from 1 to 256.

iSend

[in] The pointer to an float array that will hold the multiple analog output values. The maximum elements of the array is 32.

Return Values

0 indicates success. **WCA_COMNO_OVER** means the iCOM over the range. The legal range is from number 2 to number 3.**WCA_SLAVENO_OVER** means the iSlave over the range. The legal range is from number 1 to number 256.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Set the analog output channel 0 to channel 3 to slave1 via COM2
float iSend[4]={(float)3.6,(float)3,(float)-4,(float)0};
SetRemoteMultiAO(2, 1, iSend);
```

GetRemoteCountVal

Available soon.

unsigned char GetRemoteCountVal(
unsigned char iCOM,
unsigned char iSlave,
unsigned char iChannel,
unsigned short *iRecv
)

Parameters

Return Values

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

1.1.4 Modbus Function

These functions allow users to add their own variables into modbus server for sharing the values to modbus client.

MBSetToCoil

The function can set a coil value into modbus server.

unsigned char MBSetToCoil(
unsigned short iMBAddress,
unsigned char iStatus,
unsigned char iAttr
)

Parameters

iMBAddress

[in] The modbus address which you want to set into. The range of modbus address is from 499 to 2048.

iStatus

[in] The coil status of specific modbus address. 1 means ON. 0 means OFF.

iAttr

[in] Assign which kind of coil you want set. 1 means input coil which will be requested by modbus function number 2. 0 means output coil which will be requested by modbus function number 1/5/15.

Return Values

0 indicates success. **WCA_MBADDR_OVER** means the iMBAddress over the range. The legal range is from number 499 to number 2048. **WCA_MBATTR_ERROR** means the iAttr is neither 1 nor 0.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

//Set input coil status ON at address 1
MBSetToCoil(1,1,1);

MBGetFromCoil

The function can get a coil value from a specific modbus address.

unsigned char MBGetFromCoil(
unsigned short iMBAddress,
unsigned char *iStatus,
unsigned char iAttr
)

Parameters

iMBAddress

[in] The modbus address which you want to get from. The range of modbus address is from 499 to 2048.

iStatus

[out] The coil status of specific modbus address. 1 means ON. 0 means OFF.

iAttr

[in] Assign which kind of coil you want get. 1 means input coil which will be requested by modbus function number 2. 0 means output coil which will be requested by modbus function number 1/5/15.

Return Values

0 indicates success. **WCA_MBADDR_OVER** means the iMBAddress over the range. The legal range is from number 499 to number 2048. **WCA_MBATTR_ERROR** means the iAttr is neither 1 nor 0.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Get input coil status from address 1
unsigned char iStatus;
MBGetFromCoil(1,&iSatus,1);
```

MBSetToReg

The function can set a register value into modbus server.

unsigned char MBSetToReg(
unsigned short iMBAddress,
short iStatus,
unsigned char iAttr
)

Parameters

iMBAddress

[in] The modbus address which you want to set into. The range of modbus address is from 255 to 2048.

iStatus

[in] The register value of specific modbus address.

iAttr

[in] Assign which kind of register you want set. 1 means input register which will be requested by modbus function number 4. 0 means output register which will be requested by modbus function number 3/6/16.

Return Values

0 indicates success. **WCA_MBADDR_OVER** means the iMBAddress over the range. The legal range is from number 255 to number 2048. **WCA_MBATTR_ERROR** means the iAttr is neither 1 nor 0.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Set input register value 123 at address 1
MBSetToReg(1,123,1);
```

MBGetFromReg

The function can get a register value from a specific modbus address.

unsigned char MBGetFromReg(
unsigned short iMBAddress,
short *iStatus,
unsigned char iAttr
)

Parameters

iMBAddress

[in] The modbus address which you want to get from. The range of modbus address is from 255 to 2048.

iStatus

[out] The register value of specific modbus address.

iAttr

[in] Assign which kind of register you want get. 1 means input register which will be requested by modbus function number 4. 0 means output register which will be requested by modbus function number 3/6/16.

Return Values

0 indicates success. **WCA_MBADDR_OVER** means the iMBAddress over the range. The legal range is from number 255 to number 2048. **WCA_MBATTR_ERROR** means the iAttr is neither 1 nor 0.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Get input register value from address 1
short iSataus;
MBGetFromReg(1,&iSatus,1);
```

1.1.5 User Shared Function

These functions allow users to add their own variables into share memory block for sharing the values with different application program.

UserSetCoil

The function can set an unsigned char variable into share memory block.

unsigned char UserSetCoil(
unsigned short iUserAddress,
unsigned char iStatus
)

Parameters

iUserAddress

[in] The address which you want to set into. The range of address is from 1 to 19999.

iStatus

[in] unsigned char variable.

Return Values

0 indicates success. **WCA_USERADDR_OVER** means the iUserAddress over the range. The legal range is from number 1 to number 19999.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

//Set coil value into address 1
UserSetCoil(1,1);

UserGetCoil

The function can get an unsigned char variable from share memory block.

unsigned char UserGetCoil(
unsigned short iUserAddress,
unsigned char *iStatus
)

Parameters

iUserAddress

[in] The address which you want to get from. The range of address is from 1 to 19999.

iStatus

[out] The pointer to an unsigned char variable.

Return Values

0 indicates success. **WCA_USERADDR_OVER** means the iUserAddress over the range. The legal range is from number 1 to number 19999.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Get coil value from address 1
unsigned char iStatus;
UserGetCoil(1,&iSatus);
```

UserSetReg

The function can set a short variable into share memory block.

unsigned char UserSetReg(
unsigned short iUserAddress,
short iStatus
)

Parameters

iUserAddress

[in] The address which you want to set into. The range of address is from 1 to 19999.

iStatus

[in] short variable.

Return Values

0 indicates success. **WCA_USERADDR_OVER** means the iUserAddress over the range. The legal range is from number 1 to number 19999.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Set register value 123 into address 1
UserSetReg(1,123);
```

UserGetReg

The function can get an short variable from share memory block.

unsigned char UserGetReg(
unsigned short iUserAddress,
short *iStatus
)

Parameters

iUserAddress

[in] The address which you want to get from. The range of address is from 1 to 19999.

iStatus

[out] The pointer to a short variable.

Return Values

0 indicates success. **WCA_USERADDR_OVER** means the iUserAddress over the range. The legal range is from number 1 to number 19999.

Remarks

Requirements

Runs on	Versions	Defined in	Include	Link to
WinCon 8000	4.1.0.01 and later	ScanKernel.lib	WinConAgent.h	

Example

```
//Get register value from address 1
short iStatus;
UserGetReg(1,&iSatus);
```

1.2 ScanKernel API For VB.NET/VC#.NET developer

Step 1:

Create a smart device project

Step 2:

[Add Reference] ->ScanKernelNet.dll

Step 3:

Refer to the function prototype of ScanKernelNet.dll by Object Browser

Step 4:

Call the functions in the ScanKernelNet.dll (Please refer to the SKernel_VB.NET_Demo / SKernel_VC#.NET_Demo)

Step 5:

Build your project and copy it and relative library into WinCON 8000

Note: ScanKernel.dll, ScanKernelNet.dll, and VB.NET/VC#.NET application program must be copied to the same folder in the WinCON-8000

1.3 Supported Modbus Commands

The Modbus protocol establishes the format for the master's query by placing into the device (or broadcast) address, a function code defining the requested action, any data to be sent, and an error checking field. The slave's response message is also constructed using the Modbus protocol. It contains fields confirming the action taken, any data to be returned, and an error-checking field. If an error occurred in receipt of the message, or if the slave is unable to perform the requested action, the slave will construct an error message and send it as its response.

Code Description I/O Unit Min Max					
Code	**Description**	**I/O**	**Unit**	**Min**	**Max**
01(0x01)	Read Coil	Status In	Bit	1	2000(0x7D0)
02(0x02)	Read Discrete Inputs	Status In	Bit	1	2000(0x7D0)
03(0x03)	Read Holding Registers	Registers In	Word	1	125(0x7D)
04(0x04)	Read Input Registers	Registers In	Word	1	125(0x7D)
05(0x05)	Write Single Coil	Coil Out	Bit	1	1
06(0x06)	Write Single Register	Register Out	Word	1	1
15(0x0F)	Write Multiple Coils	Coils Out Bit	Bit	1	800
16(0x10)	Write Multiple registers	Registers Out Word	Word	1	100

2. Appendix

2.1 Appendix A - Error list and description

Code Description I/O Unit Min Max		
Code	**Define**	**Description**
0	WCA_OK	OK
102	WCA_Stop	ScanKernel has been stopped
103	WCA_SLOTNO_OVER	Slot number must be 1 - 8
104	WCA_ATT_ERROR	Attribute number error. It should be 1 or 0
105	WCA_COMNO_OVER	COM port No. must be 2 or 3
106	WCA_SLAVENO_OVER	Slave number must be 1 - 256
107	WCA_NOT_MASTER	Not the main AP which calls ScanKernel
108	WCA_MBADDR_OVER	Modbus DIO address must be 449 – 2048, AIO address must be 225 - 2048
109	WCA_MBATTR_ERROR	Modbus attribute must be 1 or 0
110	WCA_USERADDR_OVER	User defined address must be 1 - 8192
111	WCA_USERRATTR_ERROR	User defined register value must be -32768 to 32767

2.2 Appendix B – Module list

Type	Analog Input/Output Modules	Digital I/O, Relay and Counter Modules	Analog Output Modules
7K	7011/ 7011D/ 7011P/ 7011PD 7012/ 7012D/ 7012F/ 7012FD 7013/ 7013D 7014D 7016/ 7016D/ 7016P/ 7016PD 7017/ 7017F/ 7017C/ 7017R 7018/ 7018P/7018BL 7033/ 7033D	7041/ 7041D 7042/ 7042D 7043/ 7043D 7044/ 7044D 7050/ 7050D/ 7050A /7050AD 7052/ 7052D 7053/ 7053D 7060/ 7060D 7063/ 7063A/ 7063B 7063D/ 7063AD/ 7063BD 7065/ 7065D/ 7065A/ 7065B 7065AD/ 7065BD 7066/ 7066D 7067/ 7067D 7080/ 7080D	7021/ 7021P 7022/ 7024
8K	8017H	8037/ 8040/ 8041/ 8042/ 8050/ 8051/ 8052/ 8053/ 8054/ 8055/ 8056/ 8057/ 8058/ 8060/ 8063/ 8064/ 8065/ 8066/ 8068/ 8069/ 8077	8024
87K	87013, 87017, 87018	87051/ 87052/ 87053 87054/ 87055/ 87057 87058/ 87063/ 87064 87065/ 87066/ 87068/ 87069	87022,87024,87026