Modbus Master API

(Modbus/TCP & Modbus/RTU & Modbus/ASCII)

Programmer's Manual

REV 1.08

2008/4/5

1. Modbus Master API

1.1 Modbus Master API For eVC++ developer

Step 1:

Create an eVC++ project

Step 2:

Include Modbus_ARM.h and call functions of Modbus_ARM.dll (Please refer to MB_eVC_Demo)

Step 3:

Build your project and copy it into WinPAC

Note: Your AP, Modbus_ARM.dll, WinPacSDK.dll, and WinConSDK.dll must be copied to the same folder in the WinPAC

Modbus/Master TCP APIs

int MBTCPInit(int iSocketNumber, char *tcpipaddr, int tcpipport, int iTimeOut);
void MBTCPClose(int iSocketNumber);
int MBTCP_R_Coils(int iSocketNumber,int iSlaveNumber, int iStartAddress, int iCount,
unsigned char *iRecv, int iFuncNumber);
int MBTCP_W_Coil(int iSocketNumber, int iSlaveNumber, int iCoilAddress, int iCoilStatus);
int MBTCP_W_Multi_Coils(int iSocketNumber , int iSlaveNumber, int iCoilAddress, int iCount, unsigned char *iSend);
int MBTCP_R_Registers(int iSocketNumber, int iSlaveNumber, int iStartAddress, int iCount, short *iRecv, int FuncNumber);
int MBTCP_W_Register(int iSocketNumber, int iSlaveNumber, int iRegAddress, short iRegStatus);
int MBTCP_W_Multi_Registers(int iSocketNumber, int iSlaveNumber, int iRegAddress, int iCount, short *iRegStauus);

Modbus/Master RTU APIs

int MBRTUInit(int iPortNumber, int iBaudrate, int iParity, int iDataBit, int iStopBit, int iTimeOut);
void MBRTUClose(int iPortNumber);
int MBRTU_R_Coils(int iPortNumber,int iSlaveNumber, int iStartAddress, int iCount,
unsigned char *iRecv, int iFuncNumber);
int MBRTU_W_Coil(int iPortNumber, int iSlaveNumber, int iCoilAddress, int iCoilStatus);
int MBRTU_W_Multi_Coils(int iPortNumber, int iSlaveNumber, int iCoilAddress, int iCount, unsigned char *iCoilStatus);
int MBRTU_R_Registers(int iPortNumber, int iSlaveNumber, int iStartAddress, int iCount, short *iRecv, int FuncNumber);
int MBRTU_W_Register(int iPortNumber, int iSlaveNumber, int iRegAddress, short iRegStatus);
int MBRTU_W_Multi_Registers(int iPortNumber, int iSlaveNumber, int iRegAddress, int iCount, short *iRegStatus);

Modbus/Master ASCII APIs

int MBASCInit(int iPortNumber, int iBaudrate,int iParity, int iDataBit, int iStopBit, int iTimeOut);
void MBASCClose(int iPortNumber);
int MBASC_R_Coils(int iPortNumber, int iSlaveNumber, int iStartAddress, int iCount,
unsigned char *iRecv, int iFuncNumber);
int MBASC_W_Coil(int iPortNumber, int iSlaveNumber, int iCoilAddress, int iCoilStatus);
int MBASC_W_Multi_Coils(int iPortNumber, int iSlaveNumber, int iCoilAddress, int iCount, unsigned char *iCoilStatus);
int MBASC_R_Registers(int iPortNumber, int iSlaveNumber, int iStartAddress, int iCount, short *iRecv, int iFuncNumber);
int MBASC_W_Register(int iPortNumber, int iSlaveNumber, int iRegAddress, short iRegStatus);
int MBASC_W_Multi_Registers(int iPortNumber, int iSlaveNumber, int iRegAddress, int iCount, short *iRegStatus);

MBTCPInit

This function initializes the socket you want to create.

int MBTCPInit(

int iSocketNumber,

char *tcpipaddr,

int tcpipport

int iTimeOut

);

Parameters

iSocketNumber

[in] The socket ID number which's range is from 1 to 255.

tcpipaddr

[in] The IP address of the target Modbus/TCP device.

tcpipport

[in] The port number of the target Modbus/TCP device.

iTimeOut

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

Return Values

0 indicates success. Non zero indicates failure. (Please refer to the Appendix 2.1)

Remarks

Before you use the following Modbus/TCP function, you have to call this function to initialize your socket.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinPAC	4.1.0.01 and later	Modbus.ARM.lib	Modbus_ARM.h	

Example

```
//Initialize the socket (IP address = 192.168.1.199; Port number=502, 5000)
//Socket ID Number = 1
//Timeout = 100 ms
int iInitSuccess;
iInitSuccess = MBTCPInit(1,"192.168.1.199",502, 100);
```

MBTCPClose

This function close the existing socket which you created using MBTCPInit.

void MBTCPClose(

int iSocketNumber

);

Parameters

iSocketNumber

[in] The socket ID number you used to create using MBTCPInit.

Return Values

No return value.

Remarks

If you don’t want to use the socket anymore, you had better call this function to close the socket.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinPAC	4.1.0.01 and later	Modbus_ARM.lib	Modbus_ARM.h	

Example

```
//Initialize the socket (IP address = 192.168.1.199; Port number=502, 5000)
//Socket ID number = 1
//Timeout = 5000 ms
int iInitSuccess;
iInitSuccess = MBTCPInit(1,”192.168.1.199”,502,5000);
//Close the socket number 1
MBTCPClose(1);
```

MBTCP_R_Coils

This function allows you to read continuous coil statuses from the Modbus/TCP device.

int MBTCP_R_Coils(

int iSocketNumber,

int iSlaveNumber,

int iStartAddress,

int iCount,

unsigned char* iRecv,

int iFuncNumber

);

Parameters

iSocketNumber

[in] The socket ID number you used to create using MBTCPInit.

iSlaveNumber

[in] The slave number of your Modbus/TCP device.

iStartAddress

[in] The decimal starting address of the coils you want to read.

iCount

[in] The count of the coils you want to read.

iRecv

[out] The array which contains coil statuses.

iFunctNumber

[in] The function number is either 1 or 2 which depends on your Modbus/TCP device.

Return Values

0 indicates success. Non zero indicates failure. (Please refer to the Appendix 2.1)

Remarks

This function use the modbus function number 1 or 2.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinPAC	4.1.0.01 and later	Modbus_ARM.lib	Modbus_ARM.h	

Example

```
//Initialize the socket (IP address = 192.168.1.199; Port number=502, 5000)
//Socket ID number = 1
//Timeout = 5000 ms
int iInitSuccess;
iInitSuccess = MBTCPInit(1,”192.168.1.199”,502,5000);
//Read coil statuses (Slave address =1; Start address =1; Count=5; Function number =1)
int iReadSuccess;
unsigned char iRecv[5]={0};
iReadSuccess =MBTCP_R_Coils(1,1,1,5,iRecv,1);
//Close the socket number 1
MBTCPClose(1);
```

MBTCP_W_Coil

This function allows you to write a coil status to the Modbus/TCP device.

int MBTCP_W_Coil(

int iSocketNumber,

int iSlaveNumber,

int iCoilAddress,

int iCoilStatus

);

Parameters

iSocketNumber

[in] The socket ID number you used to create using MBTCPInit.

iSlaveNumber

[in] The slave number of your Modbus/TCP device.

iCoilAddress

[in] The decimal address of the coil you want to write.

iCoilStatus

[in] The coil status you want to give. 1 indicates TRUE. 0 indicates FALSE.

Return Values

0 indicates success. Non zero indicates failure. (Please refer to the Appendix 2.1)

Remarks

This function use the modbus function number 5.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinPAC	4.1.0.01 and later	Modbus_ARM.lib	Modbus_ARM.h	

Example

```
//Initialize the socket (IP address = 192.168.1.199; Port number=502, 5000)
//Socket ID number = 1
//Timeout = 5000 ms
int iInitSuccess;
iInitSuccess = MBTCPInit(1,”192.168.1.199”,502,5000)
//Write a coil status (Slave address =1; Coil address =1; Coil status =TRUE)
int iWriteSuccess;
iWriteSuccess =MBTCP_W_Coil(1,1,1,1);
//Close the socket number 1
MBTCPClose(1);
```

MBTCP_W_Multi_Coils

This function allows you to write several coil statuses to the Modbus/TCP device.

int MBTCP_W_Multi_Coils(

int iSocketNumber,

int iSlaveNumber,

int iCoilAddress,

int iCount,

unsigned char* iCoilStatus

);

Parameters

iSocketNumber

[in] The socket ID number you used to create using MBTCPInit.

iSlaveNumber

[in] The slave number of your Modbus/TCP device.

iCoilAddress

[in] The decimal starting address of the coils you want to write.

iCount

[in] The count of the coils you want to write. It must be no more than 800.

iCoilStatus

[in] The array which contains coil statuses. The size of array must be no more than 800.

Return Values

0 indicates success. Non zero indicates failure. (Please refer to the Appendix 2.1)

Remarks

This function use the modbus function number 15.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinPAC	4.1.0.01 and later	Modbus_ARM.lib	Modbus_ARM.h	

Example

```
//Initialize the socket (IP address = 192.168.1.199; Port number=502, 5000)
//Socket ID number = 1
//Timeout = 5000 ms
int iInitSuccess;
iInitSuccess = MBTCPInit(1,”192.168.1.199”,502, 5000);
//Write coil statuses (Slave address =1; Start address =10; iCount =5)
int iWriteSuccess;
unsigned char iSend[5] = {1,0,1,0,1};
iWriteSuccess =MBTCP_W_Multi_Coils(1,1,10,5,iCoilStatus);
//Close the socket number 1
MBTCPClose(1);
```

MBTCP_R_Registers

This function allows you to read continuous registry values from the Modbus/TCP device.

short MBTCP_R_Registers(

int iSocketNumber,

int iSlaveNumber,

int iStartAddress,

int iCount,

short *iRecv,

int iFuncNumber

);

Parameters

iSocketNumber

[in] The socket ID number you used to create using MBTCPInit.

iSlaveNumber

[in] The slave number of your Modbus/TCP device.

iStartAddress

[in] The decimal starting address of the registries you want to read.

iCount

[in] The count of the registries you want to read.

iRecv

[out] The array which contains registry values.

iFunctNumber

[in] The function number is either 3 or 4 which depends on your Modbus/TCP device.

Return Values

0 indicates success. Non zero indicates failure. (Please refer to the Appendix 2.1)

Remarks

This function use the modbus function number 3 or 4.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinPAC	4.1.0.01 and later	Modbus_ARM.lib	Modbus_ARM.h	

Example

```
//Initialize the socket (IP address = 192.168.1.199; Port number=502, 5000)
//Socket ID number = 1
//Timeout = 5000 ms
int iInitSuccess;
iInitSuccess = MBTCPInit(1,”192.168.1.199”,502, 5000);
//Read registry values (Slave address =1; Start address =1; Count =5; Function number =3)
int iRegSuccess;
short iRecv[5]={0};
iRegSuccess =MBTCP_R_Registers(1,1,1,5,iRecv,3);
//Close the socket number 1
MBTCPClose(1);
```

MBTCP_W_Register

This function allows you to write a registry value to the Modbus/TCP device.

int MBTCP_W_Register(

int iSocketNumber,

int iSlaveNumber,

int iRegAddress,

short iRegStatus

);

Parameters

iSocketNumber

[in] The socket ID number you used to create using MBTCPInit.

iSlaveNumber

[in] The slave number of your Modbus/TCP device.

iRegAddress

[in] The decimal address of the registry you want to write.

iRegStatus

[in] The registry value you want to give. The range is from -32768 to 32767.

Return Values

0 indicates success. Non zero indicates failure. (Please refer to the Appendix 2.1)

Remarks

This function use the modbus function number 6.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinPAC	4.1.0.01 and later	Modbus_ARM.lib	Modbus_ARM.h	

Example

```
//Initialize the socket (IP address = 192.168.1.199; Port number=502, 5000)
//Socket ID number = 1
//Timeout = 5000 ms
int iInitSuccess;
iInitSuccess = MBTCPInit(1,”192.168.1.199”,502,5000);
//Write a register value (Slave address =1; Register address =1; Register value=32767)
int iWriteSuccess;
iWriteSuccess =MBTCP_W_Register(1,1,1,32767);
//Close the socket number 1
MBTCPClose(1);
```

MBTCP_W_Multi_Registers

This function allows you to write several registry values to the Modbus/TCP device.

int MBTCP_W_Multi_Registries(

int iSocketNumber,

int iSlaveNumber,

int iRegAddress,

int iCount,

short *iRegStatus

);

Parameters

iSockettNumber

[in] The socket ID number you used to create using MBTCPInit.

iSlaveNumber

[in] The slave number of your Modbus/TCP device.

iRegAddress

[in] The decimal starting address of the registry you want to write.

iCount

[in] The count of the registries you want to write. It must be no more than 100.

iRegStatus

[in] The array which contains register values. The size of array must be no more than 100. The range is from -32768 to 32767.

Return Values

0 indicates success. Non zero indicates failure. (Please refer to the Appendix 2.1)

Remarks

This function use the modbus function number 16.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinPAC	4.1.0.01 and later	Modbus_ARM.lib	Modbus_ARM.h	

Example

```
//Initialize the socket (IP address = 192.168.1.199; Port number=502, 5000)
//Socket ID number = 1
//Timeout = 5000 ms
int iInitSuccess;
iInitSuccess = MBTCPInit(1,”192.168.1.199”,502,5000);
//Write register values (Slave address =1; Regaddress =10; iCount=5)
int iWriteSuccess;
short iSend[5] = {1234,1234,1234,1234,1234};
iWriteSuccess =MBTCP_W_Multi_Registers(1,1,10,5,iRegStatus);
//Close the socket number 1
MBTCPClose(1);
```

MBRTUInit

This function initializes the COM port you want to create.

int MBRTUInit(

int iPortNumber,

int iBaudrate,

int iParity,

int iDataBit,

int iStopBit,

int iTimeOut,

);

Parameters

iPortNumber

[in] The COM port number which's range is from 1 to 3.

iBaudrate

[in] The baud rate of COM port which should be equal to the target Modbus/RTU device.

iParity

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

Value	Description
0	No parity
1	Even
2	Mark
3	Odd
4	Space

iDataBit

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

iStopBit

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

Value	Description
1	1 stop bit
2	2 stop bits
3	1.5 stop bits

iTimeOut

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

Return Values

0 indicates success. Non zero indicates failure. (Please refer to the Appendix 2.1)

Remarks

Before you use the following Modbus/RTU function, you have to call this function to initialize your COM.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinPAC	4.1.0.01 and later	Modbus_ARM.lib	Modbus_ARM.h	

Example

```
//Initialize the COM port
// COM port number = 2
// Baud rate=19200
// Parity = No parity
// DateBits = 8
// StopBits = 1 stop bit
// TimeOut = 1000 ms
int iInitSuccess;
iInitSuccess = MBRTUInit(2, 19200, 0, 8, 1, 1000);
```

MBRTUClose

This function close the existing COM port which you created using MBRTUInit.

void MBRTUClose(

int iPortNumber

);

Parameters

iPortNumber

[in] The COM port number you used to create using MBRTUInit.

Return Values

No return value.

Remarks

If you don’t want to use the COM port anymore, you had better call this function to close the COM port.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinPAC	4.1.0.01 and later	Modbus_ARM.lib	Modbus_ARM.h	

Example

```
//Initialize the COM 2
int iInitSuccess;
iInitSuccess = MBRTUInit(2, 19200, 0, 8, 1, 1000);
//Close the COM 2
MBRTUClose(2);
```

MBRTU_R_Coils

This function allows you to read continuous coil statuses from the Modbus/RTU device.

int MBRTU_R_Coils(

int iPortNumber,

int iSlaveNumber,

int iStartAddress,

int iCount,

unsigned char* iRecv,

int iFuncNumber

);

Parameters

iPortNumber

[in] The COM port number you used to create using MBRTUInit.

iSlaveNumber

[in] The slave number of your Modbus/RTU device.

iStartAddress

[in] The decimal starting address of the coils you want to read.

iCount

[in] The count of the coils you want to read.

iRecv

[out] The array which contains coil statuses.

iFunctNumber

[in] The function number is either 1 or 2 which depends on your Modbus/RTU device.

Return Values

0 indicates success. Non zero indicates failure. (Please refer to the Appendix 2.1)

Remarks

This function use the modbus function number 1 or 2.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinPAC	4.1.0.01 and later	Modbus_ARM.lib	Modbus_ARM.h	

Example

```
//Initialize the COM 2
int iInitSuccess;
iInitSuccess = MBRTUInit(2, 19200, 0, 8, 1, 1000);
//Read coil statuses (Slave address =1; Start address =1; Count=5; Function number =1)
int iReadSuccess;
unsigned char iRecv[5]={0};
iReadSuccess =MBRTU_R_Coils(2,1,1,5,iRecv,1);
//Close the COM 2
MBRTUClose(2);
```

MBRTU_W_Coil

This function allows you to write a coil status to the Modbus/RTU device.

int MBRTU_W_Coil(

int iPortNumber,

int iSlaveNumber,

int iCoilAddress,

int iCoilStatus

);

Parameters

iPortNumber

[in] The COM port number you used to create using MBRTUInit.

iSlaveNumber

[in] The slave number of your Modbus/RTU device.

iCoilAddress

[in] The decimal address of the coil you want to write.

iCoilStatus

[in] The coil status you want to give. 1 indicates TRUE. 0 indicates FALSE.

Return Values

0 indicates success. Non zero indicates failure. (Please refer to the Appendix 2.1)

Remarks

This function use the modbus function number 5.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinPAC	4.1.0.01 and later	Modbus_ARM.lib	Modbus_ARM.h	

Example

```
//Initialize the COM 2
int iInitSuccess;
iInitSuccess = MBRTUInit(2, 19200, 0, 8, 1, 1000);
//Write a coil status (Slave address =1; Coil address =1; Coil status =TRUE)
int iWriteSuccess;
iWriteSuccess =MBRTU_W_Coil(2,1,1,1);
//Close the COM 2
MBRTUClose(2);
```

MBRTU_W_Multi_Coils

This function allows you to write several coil statuses to the Modbus/RTU device.

int MBRTU_W_Multi_Coils(

int iPortNumber,

int iSlaveNumber,

int iCoilAddress,

int iCount,

unsigned char* iCoilStaus

);

Parameters

iPortNumber

[in] The COM port number you used to create using MBRTUInit.

iSlaveNumber

[in] The slave number of your Modbus/RTU device.

iCoiltAddress

[in] The decimal starting address of the coils you want to write.

iCount

[in] The count of the coils you want to write. It must be no more than 800.

iCoilStatus

[in] The array which contains coil statuses. The size of array must be no more than 800.

Return Values

0 indicates success. Non zero indicates failure. (Please refer to the Appendix 2.1)

Remarks

This function use the modbus function number 15.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinPAC	4.1.0.01 and later	Modbus_ARM.lib	Modbus_ARM.h	

Example

```
//Initialize the COM 2
int iInitSuccess;
iInitSuccess = MBRTUInit(2, 19200, 0, 8, 1, 1000);
//Write coil statuses (Slave address =1; Start address =10; iCount =5)
int iWriteSuccess;
unsigned char iSend[5] = {1,0,1,0,1};
iWriteSuccess =MBRTU_W_Multi_Coils(2,1,10,5,iCoilStatus);
//Close the COM 2
MBRTUClose(2);
```

MBRTU_R_Registers

This function allows you to read continuous registry values from the Modbus/RTU device.

short MBRTU_R_Registers(

int iPortNumber,

int iSlaveNumber,

int iStartAddress,

int iCount,

short *iRecv,

int iFuncNumber

);

Parameters

iPortNumber

[in] The COM port number you used to create using MBRTUInit.

iSlaveNumber

[in] The slave number of your Modbus/RTU device.

iStartAddress

[in] The decimal starting address of the registries you want to read.

iCount

[in] The count of the registries you want to read.

iRecv

[out] The array which contains registry values.

iFunctNumber

[in] The function number is either 3 or 4 which depends on your Modbus/RTU device.

Return Values

0 indicates success. Non zero indicates failure. (Please refer to the Appendix 2.1)

Remarks

This function use the modbus function number 3 or 4.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinPAC	4.1.0.01 and later	Modbus_ARM.lib	Modbus_ARM.h	

Example

```
//Initialize the COM 2
int iInitSuccess;
iInitSuccess = MBRTUInit(2, 19200, 0, 8, 1, 1000);
//Read register values (Slave address =1; Start address =1; Count =5; Function number =3)
int iRegSuccess;
short iRecv[5]={0};
iRegSuccess =MBRTU_R_Registers(2,1,1,5,iRecv,3);
//Close the COM 2
MBRTUClose(2);
```

MBRTU_W_Register

This function allows you to write a registry value to the Modbus/RTU device.

int MBRTU_W_Register(

int iPortNumber,

int iSlaveNumber,

int iRegAddress,

short iRegStatus

);

Parameters

iPorttNumber

[in] The socket ID number you used to create using MBRTUInit.

iSlaveNumber

[in] The slave number of your Modbus/RTU device.

iRegAddress

[in] The decimal address of the registry you want to write.

iRegStatus

[in] The registry value you want to give. The range is from -32768 to 32767.

Return Values

0 indicates success. Non zero indicates failure. (Please refer to the Appendix 2.1)

Remarks

This function use the modbus function number 6.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinPAC	4.1.0.01 and later	Modbus_ARM.lib	Modbus_ARM.h	

Example

```
//Initialize the COM 2
int iInitSuccess;
iInitSuccess = MBRTUInit(2, 19200, 0, 8, 1, 1000);
//Write a register value (Slave address =1; Register address =1; Register value=32767)
int iWriteSuccess;
iWriteSuccess =MBRTU_W_Register(2,1,1,32767);
//Close the COM 2
MBRTUClose(2);
```

MBRTU_W_Multi_Registers

This function allows you to write several registry values to the Modbus/RTU device.

int MBRTU_W_Multi_Registers(

int iPortNumber,

int iSlaveNumber,

int iRegAddress,

int iCount,

short *iRegStatus

);

Parameters

iPorttNumber

[in] The socket ID number you used to create using MBRTUInit.

iSlaveNumber

[in] The slave number of your Modbus/RTU device.

iRegAddress

[in] The decimal starting address of the registry you want to write.

iCount

[in] The count of the registries you want to write. It must be no more than 100.

iRegStatus

[in] The array which contains registry values. The size of array must be no more than 100. The range is from -32768 to 32767.

Return Values

0 indicates success. Non zero indicates failure. (Please refer to the Appendix 2.1)

Remarks

This function use the modbus function number 16.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinPAC	4.1.0.01 and later	Modbus_ARM.lib	Modbus_ARM.h	

Example

```
//Initialize the COM 2
int iInitSuccess;
iInitSuccess = MBRTUInit(2, 19200, 0, 8, 1, 1000);
//Write register values (Slave address =1; Regaddress =10; iCount=5)
int iWriteSuccess;
short iSend[5] = {1234,1234,1234,1234,1234};
iWriteSuccess =MBRTU_W_Registers(2,1,10,5,iRegStatus);
//Close the COM 2
MBRTUClose(2);
```

MBASCInit

This function initializes the COM port you want to create.

int MBASCInit(

int iPortNumber,

int iBaudrate,

int iParity,

int iDataBit,

int iStopBit,

int iTimeOut,

);

Parameters

iPortNumber

[in] The COM port number which's range is from 1 to 3.

iBaudrate

[in] The baud rate of COM port which should be equal to the target Modbus/ASC device.

iParity

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

Value	Description
0	No parity
1	Even
2	Mark
3	Odd
4	Space

iDataBit

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

iStopBit

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

Value	Description
1	1 stop bit
2	2 stop bits
3	1.5 stop bits

iTimeOut

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

Return Values

0 indicates success. Non zero indicates failure. (Please refer to the Appendix 2.1)

Remarks

Before you use the following Modbus/ASC function, you have to call this function to initialize your COM.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinPAC	4.1.0.01 and later	Modbus_ARM.lib	Modbus_ARM.h	

Example

```
//Initialize the COM port
// COM port number = 2
// Baud rate=19200
// Parity = No parity
// DateBits = 8
// StopBits = 1 stop bit
// TimeOut = 1000 ms
int iInitSuccess;
iInitSuccess = MBASCInit(2, 19200, 0, 8, 1, 1000);
```

MBASCClose

This function closes the existing COM port which you created using MBASCInit.

void MBASCClose(

int iPortNumber

);

Parameters

iPortNumber

[in] The COM port number you used to create using MBASCInit.

Return Values

No return value.

Remarks

If you don’t want to use the COM port anymore, you had better call this function to close the COM port.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinPAC	4.1.0.01 and later	Modbus_ARM.lib	Modbus_ARM.h	

Example

```
//Initialize the COM 2
int iInitSuccess;
iInitSuccess = MBASCInit(2, 19200, 0, 8, 1, 1000);
//Close the COM 2
MBASCClose(2);
```

MBASC_R_Coils

This function allows you to read continuous coil statuses from the Modbus/ASC device.

int MBASC_R_Coils(

int iPortNumber,

int iSlaveNumber,

int iStartAddress,

int iCount,

unsigned char* iRecv,

int iFuncNumber

);

Parameters

iPortNumber

[in] The COM port number you used to create using MBASCInit.

iSlaveNumber

[in] The slave number of your Modbus/ASC device.

iStartAddress

[in] The decimal starting address of the coils you want to read.

iCount

[in] The count of the coils you want to read.

iRecv

[out] The array which contains coil statuses.

iFunctNumber

[in] The function number is either 1 or 2 which depends on your Modbus/ASC device.

Return Values

0 indicates success. Non zero indicates failure. (Please refer to the Appendix 2.1)

Remarks

This function use the modbus function number 1 or 2.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinPAC	4.1.0.01 and later	Modbus_ARM.lib	Modbus_ARM.h	

Example

```
//Initialize the COM 2
int iInitSuccess;
iInitSuccess = MBASCInit(2, 19200, 0, 8, 1, 1000);
//Read coil statuses (Slave address =1; Start address =1; Count=5; Function number =1)
int iReadSuccess;
unsigned char iRecv[5]={0};
iReadSuccess =MBASC_R_Coils(2,1,1,5,iRecv,1);
//Close the COM 2
MBASCClose(2);
```

MBASC_W_Coil

This function allows you to write a coil status to the Modbus/ASC device.

int MBASC_W_Coil(

int iPortNumber,

int iSlaveNumber,

int iCoilAddress,

int iCoilStatus

);

Parameters

iPortNumber

[in] The COM port number you used to create using MBASCInit.

iSlaveNumber

[in] The slave number of your Modbus/ASC device.

iCoilAddress

[in] The decimal address of the coil you want to write.

iCoilStatus

[in] The coil status you want to give. 1 indicates TRUE. 0 indicates FALSE.

Return Values

0 indicates success. Non zero indicates failure. (Please refer to the Appendix 2.1)

Remarks

This function use the modbus function number 5.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinPAC	4.1.0.01 and later	Modbus_ARM.lib	Modbus_ARM.h	

Example

```
//Initialize the COM 2
int iInitSuccess;
iInitSuccess = MBASCInit(2, 19200, 0, 8, 1, 1000);
//Write a coil status (Slave address =1; Coil address =1; Coil status =TRUE)
int iWriteSuccess;
iWriteSuccess =MBASC_W_Coil(2,1,1,1);
//Close the COM 2
MBASCClose(2);
```

MBASC_W_Multi_Coils

This function allows you to write several coil statuses to the Modbus/RTU device.

int MBRTU_W_Multi_Coils(

int iPortNumber,

int iSlaveNumber,

int iCoilAddress,

int iCount,

unsigned char* iCoilStaus

);

Parameters

iPortNumber

[in] The COM port number you used to create using MBASCInit.

iSlaveNumber

[in] The slave number of your Modbus/ASC device.

iCoiltAddress

[in] The decimal starting address of the coils you want to write.

iCount

[in] The count of the coils you want to write. It must be no more than 800.

iCoilStatus

[in] The array which contains coil statuses. The size of array must be no more than 800.

Return Values

0 indicates success. Non zero indicates failure. (Please refer to the Appendix 2.1)

Remarks

This function use the modbus function number 15.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinPAC	4.1.0.01 and later	Modbus_ARM.lib	Modbus_ARM.h	

Example

```
//Initialize the COM 2
int iInitSuccess;
iInitSuccess = MBASCInit(2, 19200, 0, 8, 1, 1000);
//Write coil statuses (Slave address =1; Start address =10; iCount =5)
int iWriteSuccess;
unsigned char iSend[5] = {1,0,1,0,1};
iWriteSuccess =MBASC_W_Multi_Coils(2,1,10,5,iCoilStatus);
//Close the COM 2
MBASCClose(2);
```

MBASC_R_Registers

This function allows you to read continuous registry values from the Modbus/ASC device.

short MBASC_R_Registers(

int iPortNumber,

int iSlaveNumber,

int iStartAddress,

int iCount,

short *iRecv,

int iFuncNumber

);

Parameters

iPortNumber

[in] The COM port number you used to create using MBASCInit.

iSlaveNumber

[in] The slave number of your Modbus/ASC device.

iStartAddress

[in] The decimal starting address of the registries you want to read.

iCount

[in] The count of the registries you want to read.

iRecv

[out] The array which contains registry values.

iFunctNumber

[in] The function number is either 3 or 4 which depends on your Modbus/ASC device.

Return Values

0 indicates success. Non zero indicates failure. (Please refer to the Appendix 2.1)

Remarks

This function use the modbus function number 3 or 4.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinPAC	4.1.0.01 and later	Modbus_ARM.lib	Modbus_ARM.h	

Example

```
//Initialize the COM 2
int iInitSuccess;
iInitSuccess = MBASCInit(2, 19200, 0, 8, 1, 1000);
//Read register values (Slave address =1; Start address =1; Count =5; Function number =3)
int iRegSuccess;
short iRecv[5]={0};
iRegSuccess =MBASC_R_Registers(2,1,1,5,iRecv,3);
//Close the COM 2
MBASCClose(2);
```

MBASC_W_Register

This function allows you to write a registry value to the Modbus/RTU device.

int MBASC_W_Register(

int iPortNumber,

int iSlaveNumber,

int iRegAddress,

short iRegStatus

);

Parameters

iPorttNumber

[in] The socket ID number you used to create using MBASCInit.

iSlaveNumber

[in] The slave number of your Modbus/ASC device.

iRegAddress

[in] The decimal address of the registry you want to write.

iRegStatus

[in] The registry value you want to give. The range is from -32768 to 32767.

Return Values

0 indicates success. Non zero indicates failure. (Please refer to the Appendix 2.1)

Remarks

This function use the modbus function number 6.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinPAC	4.1.0.01 and later	Modbus_ARM.lib	Modbus_ARM.h	

Example

```
//Initialize the COM 2
int iInitSuccess;
iInitSuccess = MBASCInit(2, 19200, 0, 8, 1, 1000);
//Write a register value (Slave address =1; Register address =1; Register value=32767)
int iWriteSuccess;
iWriteSuccess =MBASC_W_Register(2,1,1,32767);
//Close the COM 2
MBASCClose(2);
```

MBASC_W_Multi_Registers

This function allows you to write several registry values to the Modbus/RTU device.

int MBASC_W_Multi_Registers(

int iPortNumber,

int iSlaveNumber,

int iRegAddress,

int iCount,

short *iRegStatus

);

Parameters

iPorttNumber

[in] The socket ID number you used to create using MBASCInit.

iSlaveNumber

[in] The slave number of your Modbus/ASC device.

iRegAddress

[in] The decimal starting address of the registry you want to write.

iCount

[in] The count of the registries you want to write. It must be no more than 100.

iRegStatus

[in] The array which contains registry values. The size of array must be no more than 100. The range is from -32768 to 32767.

Return Values

0 indicates success. Non zero indicates failure. (Please refer to the Appendix 2.1)

Remarks

This function use the modbus function number 16.

Requirements

Runs on	Versions	Defined in	Include	Link to
WinPAC	4.1.0.01 and later	Modbus_ARM.lib	Modbus_ARM.h	

Example

```
//Initialize the COM 2
int iInitSuccess;
iInitSuccess = MBASCInit(2, 19200, 0, 8, 1, 1000);
//Write register values (Slave address =1; Regaddress =10; iCount=5)
int iWriteSuccess;
short iSend[5] = {1234,1234,1234,1234,1234};
iWriteSuccess =MBASC_W_Registers(2,1,10,5,iRegStatus);
//Close the COM 2
MBASCClose(2);
```

1.2 Modbus Master API For VB.NET/VC#.NET developer

Step 1:

Create a smart device project

Step 2:

[Add Reference] ->Modbus.dll

Step 3:

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

Step 4:

Call the functions in the Modbus.dll (Please refer to the Modbus_VB.NET_Demo / Modbus_VC#.NET_Demo)

Step 5:

Build your project and copy it and relative library into WinPAC

Note: Your AP, Modbus_ARM.dll, Modbus.dll, WinPacSDK.dll and WinConSDK.dll must be copied to the same folder in the WinPAC

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**
101	MB_OPEN_PORT_ERROR	Open COM/TCP Port error
102	MB_PORTNO_OVER	COM Port is 1 - 8
103	MB_PORT_NOT_OPEN	COM/TCP Port does not open yet
104	MB_FUN_ERROR	Modbus Fun. No. error
105	MB_READ_COUNT_OVER	reading Count of Register or Bits is over range RTU: 120 register, 1920 coils ASCII: 60 register, 960 coils TCP: 120 register, 1920 coils
106	MB_SLAVENO_OVER	Modbus Slave No. must be 1 - 247
107	MB_ADDRESS_OVER	Register or Coil Address must count from 1
108	MB_COMM_TIMEOUT	Comm. timeout
109	MB_CRC_ERROR	RTU CRC Check error
110	MB_LRC_ERROR	ASCII LRC Check error
111	MB_INVALID_SOCKET	Initial Socket error
112	MB_TCP_CONNECT_ERROR	Connect Remote Modbus Server error
113	MB_TCP_SEND_ERROR	Send TCP Data error
114	MB_TCP_TIMEOUT	Waiting Modbus Response Timeout
115	MB_WSA_INIT_ERROR	WSA Startup error
116	MB_TCP_SOCKET_ERROR	Create Socket error
117	MB_TCP_BIND_ERROR	TCP Server Bind error
118	MB_TCP_LISTEN_ERROR	TCP Server Listen error
119	MB_TCP_HAS_DATA	it has data from remote Modbus Master
120	MB_WRITE_COUNT_OVER	reading Count of Register or Bits is over range RTU: 120 register, 1920 coils ASCII: 60 register, 960 coils TCP: 120 register, 1920 coils