IR API Library

(For IR-210/IR-712A/IR-712-MTCP)

User's Manual

Version 2.0

For IR API Library v2.0.0.0

www.icpdas.com

Warranty

All products manufactured by ICP DAS are under warranty regarding defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICP DAS assumes no liability for damages resulting from the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, or for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

The names used for identification only may be registered trademarks of their respective companies.

Table of Contents

1. Introduction

ICP DAS provides the IR API library for the universal IR learning remote modules (IR-210/IR-712A/IR-712-MTCP). The library is a 32-bit MFC statically-linked regular DLL designed for Microsoft Windows desktop applications. It provides an easy way to add the IR learning functions to the applications. Users can develop their own application programs for the IR modules with the IR API library.

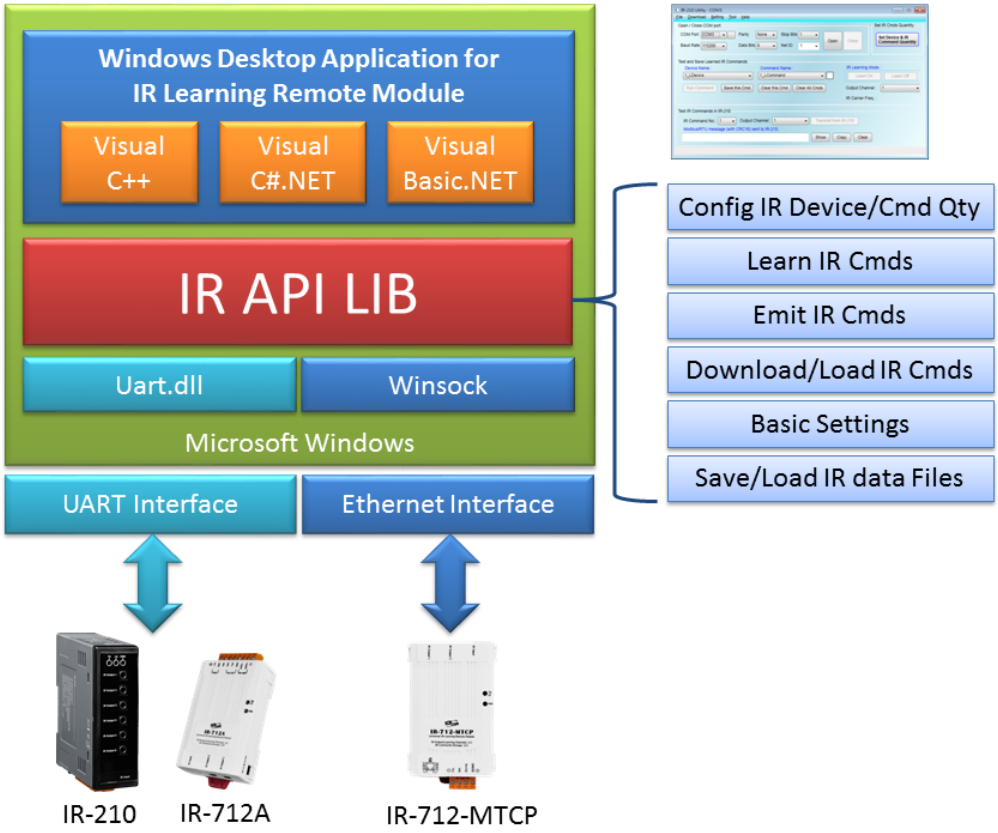


Figure 1-1: Architecture of the IR API library.

1.1 Features

- Supports Microsoft Windows XP to Windows 10.
- Supports desktop application development by Visual C++ (Win32, MFC) / C#.NET / Basic.NET.
- Supports IR Modules: IR-210 / IR-712A / IR-712-MTCP.
- Provides the planning for the quantity of IR appliances and IR commands.
- Provides IR learning and IR emitting function.
- Access IR learning data to IR module.
- Provides basic parameter settings to IR module.
- Save IR learning data to files.

1.2 Library files

Items	File Name
Header	IrLrnApi.h
Lib	irlrnapi.lib
DLL	irlrnapi.dll, Uart.dll (These twodll files should be in the same folder with the application.)
DLL for Visual C#	IrLrnCharpLib.dll, irlrnapi.dll, Uart.dll (These dll files should be in the same folder with the application.)

1.3 Flowchart of Application Development

1.3.1 Flowchart of IR learning

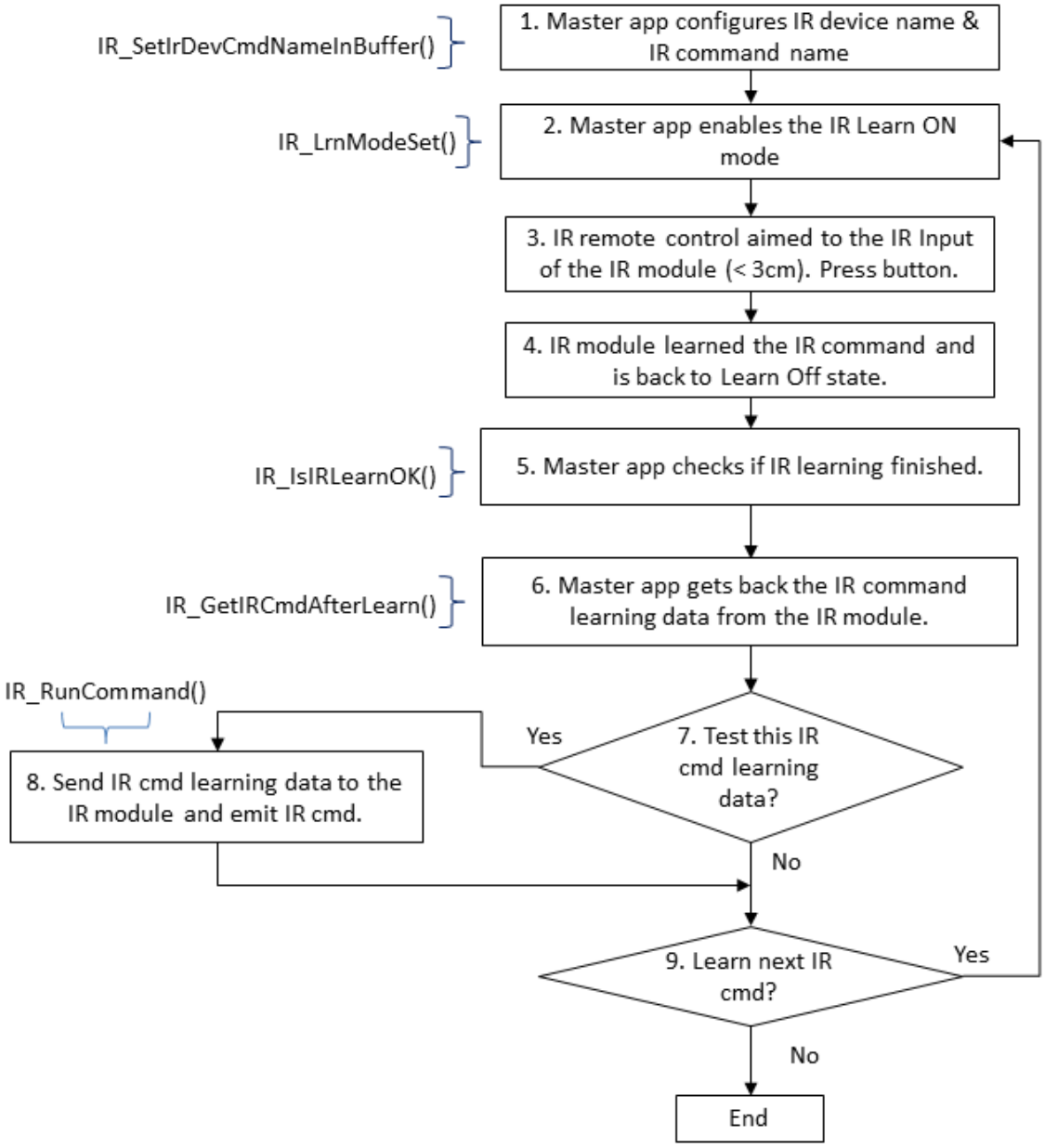


Figure 1-2: Flowchart of IR learning procedure.

1.3.2 Flowchart of downloading IR commands to IR module

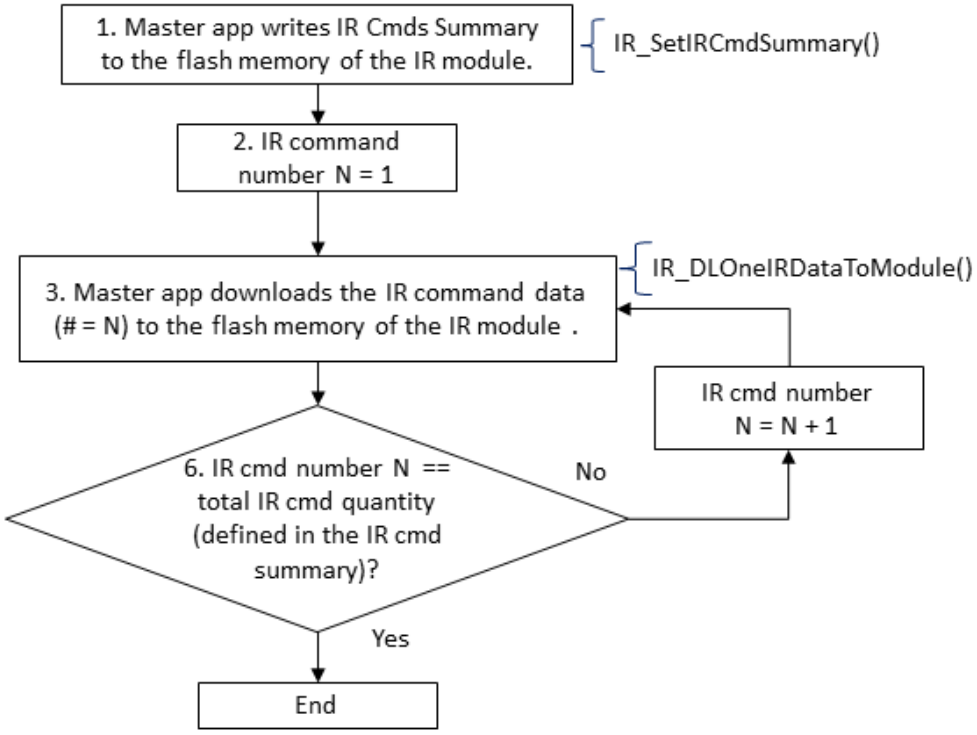


Figure 1-3: Flowchart of downloading IR commands to an IR module.

1.3.3 Flowchart of getting IR commands from IR module

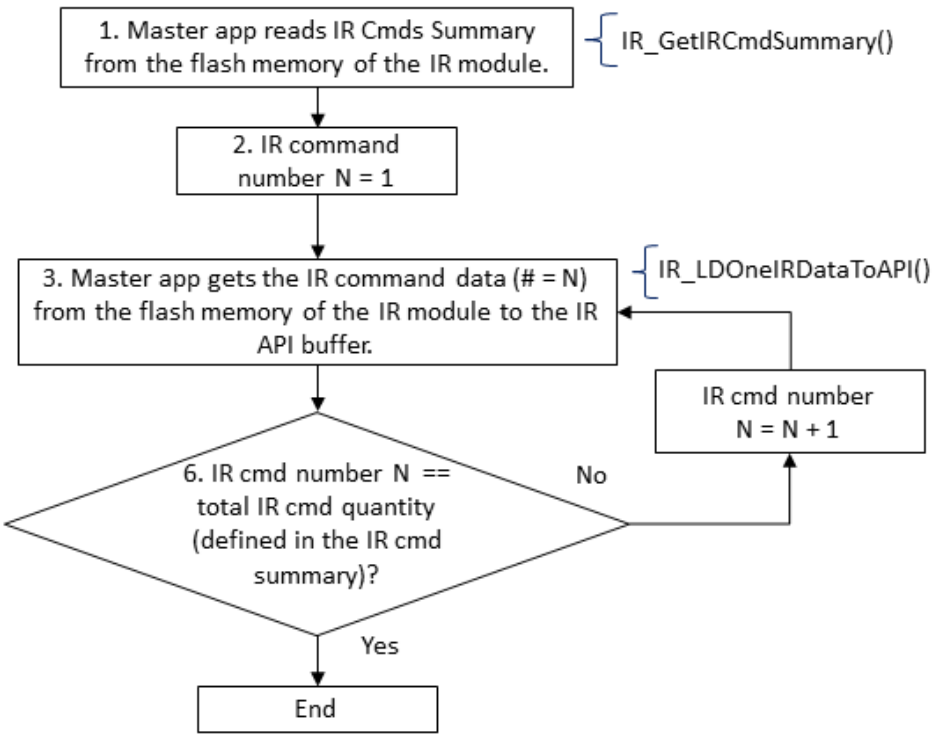


Figure 1-4: Flowchart of loading IR commands from an IR module.

2. IR API Library -- IrLrnApi.LIB

Users can develop their own Windows desktop application programs for IR learning remote modules quickly and easily by IR API library (IrLrnApi.lib). The IR API library and demos can be downloaded from the ICP DAS web site:

ftp://ftp.icpdas.com.tw/pub/cd/usbcd/napdos/ir/ir_api/windows/

2.1 API Library Overview

All the functions provided by IR API library can be separated into seven groups as shown in Figure 2-1.

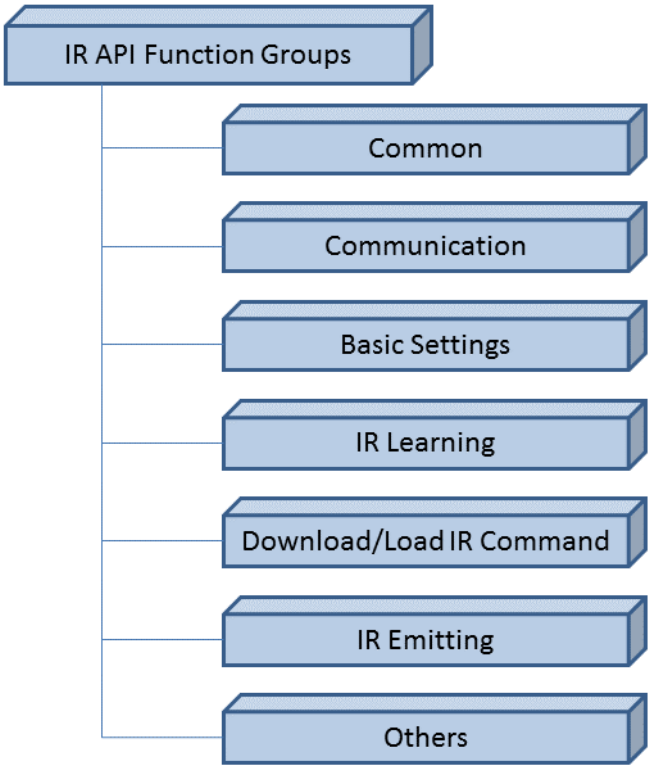


Figure 2-1: Function Groups of IR API Library

[Common Functions]
These functions can be used on the applications for the IR modules with Modbus RTU or Modbus TCP/UDP protocols, e.g. Save/Load an IR learning data file.

[Communication Functions]
Open and close the COM port or initialize and close sockets of the application program.

[Basic Settings Functions]
These functions are for the configuration of the basic settings to the IR module. The basic settings include the parameters for UART/Ethernet communication and IR Gaptime parameter.

[IR Learning Functions]
These functions are for IR learning.

[Download/Load IR Command Functions]
Download/load IR learning commands to/from the IR modules.

[IR Emitting Functions]
Request IR modules to emit the IR commands.

[Others Functions]
Get Module name

2.2 API Library Function List

All the functions provided in the IrLrnApi.lib are listed in the following tables.

2.2.1 Common Functions

The following functions can be used in both Modbus RTU and Modbus TCP/RTU applications.

Table 2-1: Common Functions for IR Modules

sections	Function Name	Description
3.1.1	IR_GetAPIVersion	Get the version number of the API library.
3.1.2	IR_GetAPIDate	Get the date of the API library.
3.1.3	IR_LoadIRCmdsFromFile	Load IR commands from an IR learning data file (*.ird) to the IR API library.
3.1.4	IR_SaveIRCmdsToFile	Save the IR commands from the IR API library to an IR learning data file(*.ird).
3.1.5	IR_ChkIrDataExistInBuffer	Check if the learning data of one IR command is available in the buffer of the library.
3.1.6	IR_ClearOneIRLrnDataInAPI	Clear an IR command learning data in the buffer of the library (excluding the IR device name & IR command name).
3.1.7	IR_ClearOneIRDeiveCmdNameInAPI	Clear an IR device name & IR command name in the buffer of the API (excluding the IR command learning data).
3.1.8	IR_GetIRCmdSummaryBuffer	Read IR command summary from the buffer of the library.
3.1.9	IR_SetIRCmdSummaryBuffer	Write IR command summary to the buffer of the library.
3.1.10	IR_GetIrDevCmdNameInBuffer	Read IR device/command name from the buffer of the library.
3.1.11	IR_SetIrDevCmdNameInBuffer	Set IR device/command name to the buffer of the library.

Table 2-2: Common Functions for IR Modules with Modbus TCP/UDP

sections	Function Name	Description
3.2.1	IR_ChkNIC	Check the number of available network interfaces of the host PC.
3.2.2	IR_GetNICInfo	Get the information of the available network interface.
3.2.3	IR_EthSearchModules	Get the number of available IR modules on the same subnet.
3.2.4	IR_EthGetSearchModuleInfo	Get the information of the searched modules.

2.2.2 Function List for Modbus RTU Communication

Table 2-3: Communication Functions

sections	Function Name	Description
4.1.1	IR_GetComPortStatus	Get COM port status of the host application.
4.1.2	IR_OpenCom	Open COM port of the host application.
4.1.3	IR_CloseCom	Close COM port of the host application.
4.1.4	IR_GetModbusRecvTimeout	Get the timeout to wait for the Modbus response for the library functions.
4.1.5	IR_SetModbusRecvTimeout	Set the timeout to wait for the Modbus response for the library functions.

Table 2-4: Basic Settings Functions

sections	Function Name	Description
4.2.1	IR_GetBasicSetting	Get basic settings from the IR module.
4.2.2	IR_SetBasicSetting	Set the basic settings to the IR module.
4.2.3	IR_ResetIRModule	Reset basic settings of the IR module to the default temporarily.

Table 2-5: IR Learning Functions

sections	Function Name	Description
4.3.1	IR_GetIRCmdSummary	Get IR Command Summary from the IR module
4.3.2	IR_SetIRCmdSummary	Set IR Command Summary (from Lib Buffer) to the IR module.
4.3.3	IR_SetIRCmdSummary1	Set IR Command Summary to the IR module.
4.3.4	IR_GetIRCmdSummaryBuffer	Get the IR command summary from the buffer of the API.
4.3.5	IR_SetIRCmdSummaryBuffer	Set the IR command summary to the buffer of the API.
4.3.6	IR_LrnModeSet	Enable/Disable the IR learning mode.
4.3.7	IR_IsIRLearnOK	Check if the IR learning is finished.
4.3.8	IR_GetIRCmdAfterLearn	Read an IR command learning data from the IR module and store it in the library buffer.

Table 2-6: Download\Load IR Command Functions

sections	Function Name	Description
4.4.1	IR_DLOneIRDataToModule	Download one IR command from API library to the IR module.
4.4.2	IR_LDOneIRDataToAPI	Load one IR command from the IR module to the API library.

Table 2-7: IR Emitting Functions

sections	Function Name	Description
4.5.1	IR_EmitIrSignal	Request the IR module to emit the IR command stored in the flash.
4.5.2	IR_RunCommand	Send an IR learning cmd data to the IR module and emit that IR cmd.

Table 2-8: Others Functions

sections	Function Name	Description
4.6.1	IR_ChkModuleName	Get the model name of the IR module.

2.2.3 Function List for Modbus TCP communication

The function names prefixed with “IR_MTCP” are for Modbus TCP communication.

Table 2-9: Communication Functions

sections	Function Name	Description
5.1.1	IR_MTCP_TCPNew	Initialize a TCP socket for Modbus TCP communication.
5.1.2	IR_MTCP_TCPNew1	Initialize a TCP socket bound to an NIC for Modbus TCP.
5.1.3	IR_MTCP_Connect	Connect to the IR module with Modbus TCP.
5.1.4	IR_MTCP_TCPClose	Disconnect and close a TCP socket.
5.1.5	IR_MTCP_GetModbusRecvTimeout	Get the receive timeout (ms) of Modbus TCP responses from the IR module.
5.1.6	IR_MTCP_SetModbusRecvTimeout	Set the receive timeout (ms) of Modbus TCP responses from the IR module.

Table 2-10: Basic Settings Functions

sections	Function Name	Description
5.2.1	IR_MTCP_GetBasicSetting	Get the basic settings to the IR module.
5.2.2	IR_MTCP_SetBasicSetting	Set the basic settings to the IR module.
5.2.3	IR_MTCP_RebootIRModule	Reboot IR Module

Table 2-11: IR Learning Functions

sections	Function Name	Description
5.3.1	IR_MTCP_GetIRCmdSummary	Get IR Command Summary from the IR module.
5.3.2	IR_MTCP_SetIRCmdSummary	Set IR Command Summary (from Lib Buffer) to the IR module.
5.3.3	IR_MTCP_SetIRCmdSummary1	Set IR Command Summary to the IR module.
5.3.4	IR_MTCP_LrnModeSet	Enable/disable IR learning mode.
5.3.5	IR_MTCP_IsIRLearnOK	Show the status of IR learning after learning an IR command.
5.3.6	IR_MTCP_GetIRCmdAfterLearn	Get one IR learning command (without device name & IR cmd name) from the IR module to the buffer of the API library.

Table 2-12: Download/Load IR Command Functions

sections	Function Name	Description
5.4.1	IR_MTCP_DLOneIRDataToModule	Download one IR command to the IR Module.
5.4.2	IR_MTCP_LDOneIRDataToAPI	Load one IR command from the IR Module.

Table 2-13: IR Emitting Functions

sections	Function Name	Description
5.5.1	IR_MTCP_EmitIrSignal	Request the IR module to emit an IR command.
5.5.2	IR_MTCP_RunCommand	Send an IR command learning data to the IR module and emit it.

Table 2-14: Others Functions

sections	Function Name	Description
5.6.1	IR_MTCP_ChkModuleName	Get the model name of an IR remote module.

2.2.4 Function List for Modbus UDP communication

The function names prefixed with “IR_MUDP” are for Modbus UDP communication.

Table 2-15: Communication Functions

sections	Function Name	Description
6.1.1	IR_MUDP_UDPNew	Initialize a UDP socket for Modbus UDP communication.
6.1.2	IR_MUDP_UDPNew1	Initialize a UDP socket bound to an NIC for Modbus UDP.
6.1.3	IR_MUDP_UDPClose	Close a UDP socket.
6.1.4	IR_MUDP_GetModbusRecvTimeout	Get the receive timeout (ms) of Modbus UDP responses from the IR module.
6.1.5	IR_MUDP_SetModbusRecvTimeout	Set the receive timeout (ms) of Modbus UDP responses from the IR module.

Table 2-16: Basic Settings Functions

sections	Function Name	Description
6.2.1	IR_MUDP_GetBasicSetting	Set the basic settings to the IR module.
6.2.2	IR_MUDP_SetBasicSetting	Get the basic settings to the IR module.
6.2.3	IR_MUDP_RebootIRModule	Reboot IR Module

Table 2-17: IR Learning Functions

sections	Function Name	Description
6.3.1	IR_MUDP_GetIRCmdSummary	Get IR Command Summary from the IR module.
6.3.2	IR_MUDP_SetIRCmdSummary	Set IR Command Summary (from Lib Buffer) to the IR module.
6.3.3	IR_MUDP_SetIRCmdSummary1	Set IR Command Summary to the IR module.
6.3.4	IR_MUDP_LrnModeSet	Enable/disable IR learning mode
6.3.5	IR_MUDP_IsIRLearnOK	Show the status of IR learning after learning an IR command.
6.3.6	IR_MUDP_GetIRCmdAfterLearn	Get one IR learning command (without device name & IR cmd name) from the IR module to the buffer of the API library.

Table 2-18: Download/Load IR Command Functions

sections	Function Name	Description
6.4.1	IR_MUDP_DLOneIRDataToModule	Download one IR command to the IR Module.
6.4.2	IR_MUDP_LDOneIRDataToAPI	Load one IR command from the IR Module.

Table 2-19: IR Emitting Functions

sections	Function Name	Description
6.5.1	IR_MUDP_EmitIrSignal	Request the IR module to emit an IR command
6.5.2	IR_MUDP_RunCommand	Send an IR command learning data to the IR module and emit it.

Table 2-20: Others Functions

sections	Function Name	Description
6.6.1	IR_MUDP_ChkModuleName	Get the model name of the IR module.

3. Common Functions

3.1 Common Functions for IR Remote Modules

3.1.1 IR_GetAPIVersion

Description:

Get the version number of the IR API library.

Prototype:

[VC++]

```
unsigned long IR_GetAPIVersion(void);
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern UInt32 IR_GetAPIVersion();
```

Parameters:

None

Return Values:

Version number of the IR API Library. Four bytes: Major, Minor, Revision, build.
E.g. 0x01020300 => v1.2.3.0

Examples:

```
[VC++]
Unsigned long Version = 0;
Unsigned char Major, Minor, Rev, Build;

Version = IR_GetAPIVersion();

Major = (unsigned char)(version >> 24);
Minor = (unsigned char)(version >> 16);
Rev   = (unsigned char)(version >> 8);
Build = (unsigned char)version;
```

3.1.2 IR_GetAPIDate

Description:

Get the date of the IR API library.

Prototype:

[VC++]

unsigned long IR_GetAPIDate(void);

[VC#.NET]

[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]

public static extern UInt32 IR_GetAPIDate();

Parameters:

None

Return Values:

Release date of the IR API library. E.g. 0x07E00412 means 2016-Apr-12.

Examples:

```
[VC++]
unsigned long ulDate = 0;
unsigned short usYear;
unsigned char usMonth, usDay;

ulDate = IR_GetAPIDate();

usYear  = (unsigned short)(ulDate >> 16);
usMonth = (unsigned char)(ulDate >> 8);
usDay   = (unsigned short)ulDate;
```

3.1.3 IR_LoadIRCmdsFromFile

Description:

Load IR commands from an IR learning data file(*.ird) to the IR API library.

Prototype:

[VC++]

```
int IR_LoadIRCmdsFromFile (
    char* filepath
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_LoadIRCmdsFromFile (
    string filepath
    );
```

Parameters:

filepath [in]

A char array for the full path of an IR learning data file.

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_OPEN_FILE_ERROR	Open file error.
IR_E_IR_FILE_FORMAT_ERR	Format error in the IR learning data file.
IR_E_CLOSE_FILE_ERROR	Error in closing file.

Examples:

```
[VC++]
int Ret = 0;
char[] filePath = “C:\\irdata.ird”;

Ret = IR_LoadIRCmdsFromFile (filePath);
```

3.1.4 IR_SaveIRCmdsToFile

Description:

Save the IR commands buffered in the IR API library to an IR learning data file(*.ird).

Prototype:

[VC++]

```
int IR_SaveIRCmdsToFile(
    char* filepath
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_SaveIRCmdsToFile(
    string filepath
    );
```

Parameters:

filepath [in]

A char array for the full path of an IR learning data file. The existing file with the same filename will be overwritten without warning.

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_IRCMD_TOTALQTY_NOMATCH	The total quantity of IR commands do not match that in the IR command summary.
IR_E_OPEN_FILE_ERROR	Error in opening file.
IR_E_CLOSE_FILE_ERROR	Error in closing file.

Examples:

```
[VC++]
int Ret = 0;
char[] filePath = “C:\\irdata.ird”;

Ret = IR_ SaveIRCmdsToFile(filePath);
```

3.1.5 IR_ChkIrDataExistInBuffer

Description:

Check if there are IR command learning data for the specified IR command number in the buffer.

(IR cmd data := IR device name + IR cmd name + IR cmd learning data)

Prototype:

[VC++]

```
int IR_ChkIrDataExistInBuffer(
    unsigned short usIrCmdNumber
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_ChkIrDataExistInBuffer(
    UInt16 usIrCmdNumber
    );
```

Parameters:

usIrCmdNumber [in]

IR command number (1 ~ 224).

Return Values:

Return Code	Description
IR_E_IRCMD_DATA_EMPTY	No IR learning data for the specified IR command number in the buffer.
IR_E_IRCMD_DATA_EXIST	There are IR learning data for the specified IR command number in the buffer.

Examples:

[VC++]

```
int Ret = 0;
Ret = IR_ChkIrDataExistInBuffer(20);  // Check IR cmd number 20.
```

3.1.6 IR_ClearOneIRLrnDataInAPI

Description:

Clear one IR command learning data in the API buffer.
(IR device name & IR command name will not be cleared.)
(IR cmd data := IR device name + IR cmd name + IR cmd learning data)

Prototype:

[VC++]

```
int IR_ClearOneIRLrnDataInAPI(
    unsigned short usIrCmdNumber,
    unsigned short usIrModuleIndex
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_ClearOneIRLrnDataInAPI(
    UInt16 usIrCmdNumber,
    UInt16 usIrModuleIndex
    );
```

Parameters:

usIrCmdNumber [in]

IR command number (1 ~ 224).

usIrModuleIndex [in]

Index of IR learning remote module. 0x01: IR-210, 0x11: IR-712A

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_FUNC_ERR	Function error.
IR_E_ILL_IRCMD_NO	Illegal IR command number
IR_E_ILL_MODULE_INDEX	Illegal module index

Examples:

```
[VC++]
int Ret = 0;

// Clear IR cmd learning data of number 2 for IR-712A.
Ret = IR_ClearOneIRLrnDataInAPI(2, 0x11);
```

3.1.7 IR_ClearOneIRDeiveCmdNameInAPI

Description:

Clear the IR device name and IR command name corrsponding to the IR command number in the API buffer.
(IR cmd data := IR device name + IR cmd name + IR cmd learning data)

Prototype:

[VC++]

```
int IR_ClearOneIRDeiveCmdNameInAPI (
    unsigned short usIrCmdNumber,
    unsigned short usIrModuleIndex
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_ClearOneIRDeiveCmdNameInAPI(
    UInt16 usIrCmdNumber,
    UInt16 usIrModuleIndex
    );
```

Parameters:

usIrCmdNumber [in]

IR command number (1 ~ 224).

usIrModuleIndex [in]

Index of IR learning remote module. 0x01: IR-210, 0x11: IR-712A

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_FUNC_ERR	Function error.
IR_E_ILL_IRCMD_NO	Illegal IR command number
IR_E_ILL_MODULE_INDEX	Illegal module index

Examples:

```
[VC++]
int Ret = 0;

// Clear IR device & cmd name of number 3 for IR-210.
Ret = IR_ClearOneIRDeiveCmdNameInAPI(3, 0x01);
```

3.1.8 IR_GetIRCmdSummaryBuffer

Description:

Read IR command summary from the buffer of the IR API Library.

Prototype:

[VC++]

```
int IR_GetIRCmdSummaryBuffer(
    unsigned short usIrModuleIndex,
    unsigned short *dataArr,
    unsigned short usInElementCnt
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_GetIRCmdSummaryBuffer (
    UInt16 usIrModuleIndex,
    [MarshalAs(UnmanagedType.LPArray, SizeParamIndex = 2)] UInt16[] dataArr,
    UInt16 wOutElementLen
    );
```

Parameters:

usIrModuleIndex [in]

Index of IR learning remote module. 0x01: IR-210, 0x11: IR-712A

dataArr [out]

(unsigned short)data array for IR command summary. Array length = 13

where

dataArr[0]: IR device quantity (1 ~ 11)

dataArr[1]: Total quantity of IR commands (1 ~ 224)

dataArr[2]: IR command quantity of device#1

dataArr[3]: IR command quantity of device#2

....

dataArr[N+1]: IR command quantity of device#N , where N<=11

usInElementCnt [in]

Length of the dataArr array. Should be at least 13.

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_ILL_SUMMARY_LEN	Illegal length of the array (dataArr) for IR command summary.
IR_E_FUNC_ERR	Function error.
IR_E_IRCMD_TOTAL_QTY_ERR	The device quantity setting exceeds the maximum.
IR_E_IRCMD_SUM_QTY_ERR	The element "Total quantity of IR commands" in the IR command summary exceeds the maximum.
IR_E_ILL_MODULE_INDEX	Illegal module index.

Examples:

```
[VC++]
int Ret = 0;
unsigned char ucPort = 0,
              ucNetID = 0;
unsigned short usIrModuleIndex = 0,
               dataArr[13] = {0},
               usInElementCnt = 13;

Ret = IR_GetIRCmdSummary(ucPort, ucNetID, usIrModuleIndex, dataArr,
    usInElementCnt);

Ret = IR_GetIRCmdSummaryBuffer(usIrModuleIndex, dataArr,
    usInElementCnt);
```

3.1.9 IR_SetIRCmdSummary

Description:

Set IR Command Summary from the library buffer to the IR module. IR_SetIRCmdSummaryBuffer() should be called before calling this function.

Prototype:

[VC++]

```
int IR_SetIRCmdSummary(
    unsigned char ucPort,
    unsigned char usNetID,
    unsigned short usIrModuleIndex
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_SetIRCmdSummary(
    Byte ucPort,
    Byte usNetID,
    UInt16 usIrModuleIndex
    );
```

Parameters:

ucPort [in]

Serial COM port number.

usNetID [in]

Modbus Net ID of the IR module.

usIrModuleIndex [in]

Index of IR learning remote module. 0x01: IR-210, 0x11: IR-712A

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_FUNC_ERR	Function error.

IR_E_DEVICE_QTY_ERR	The device quantity exceeds the maximum.
IR_E_IRCMD_TOTAL_QTY_ERR	The device quantity setting exceeds the maximum.
IR_E_IRCMD_SUM_QTY_ERR	The element "Total quantity of IR commands" in the IR command summary exceeds the maximum.
IR_E_ILL_MODULE_INDEX	Illegal module index.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.
IR_E_SEND_MBCMD_FLASH_ERR	Error in sending/receiving Modbus command to write data to Flash.

Examples:

```
[VC++]
int Ret = 0;
unsigned char ucPort = 1,
              ucNetID = 1;
unsigned short usIrModuleIndex = 0x01, // Module index for IR-210
               dataArr[13] = {2, 18, 10, 8},
               usInElementCnt = 13;

Ret = IR_SetIRCmdSummary(ucPort, ucNetID, usIrModuleIndex, dataArr,
    usInElementCnt);
```

3.1.10 IR_GetIrDevCmdNameInBuffer

Description:

Read IR device name and IR command name from the buffer of the API.

Prototype:

[VC++]

```
int IR_GetIrDevCmdNameInBuffer(
    unsigned short usIrCmdNumber,
    unsigned char *ucIrDevNameArr,
    unsigned short usDevNameByteArrLen,
    unsigned short *usDevNameByteArrOutLen,
    unsigned char *ucIrCmdNameArr,
    unsigned short usIrCmdNameByteArrLen,
    unsigned short *usCmdNameStrByteLen
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_GetIrDevCmdNameInBuffer(
    UInt16 usIrCmdNumber,
    [MarshalAs(UnmanagedType.LPArray, SizeParamIndex = 2)] Byte[]
    cIrDevNameStr,
    UInt16 ucDevNameUcLen,
    ref UInt16 usDevNameStrByteLen,
    [MarshalAs(UnmanagedType.LPArray, SizeParamIndex = 5)] Byte[]
    cIrCmdNameStr,
    UInt16 ucIrCmdNameUcLen,
    ref UInt16 usCmdNameStrByteLen
    );
```

Parameters:

usIrCmdNumber [in]

IR command number for allocating one IR learning command data in the learning data buffer.

ucIrDevNameArr [out]

Byte array for the IR device Name string. Byte Array Max length = 100 bytes (50 words).

usDevNameByteArrLen [in]

Byte length of the IR devie name array (ucIrDevNameArr). Max length = 100 bytes (50 words).

usDevNameByteArrOutLen [out]

Byte length of the actual IR devie name string without terminator.

ucIrCmdNameArr [out]

Byte array for the IR Command Name. Byte array max length = 102 bytes (51 words).

usIrCmdNameByteArrLen [in]

Byte length of the IR command name array (ucIrCmdNameArr). Max length = 102 bytes (51 words).

usIrCmdNameByteArrOutLen [out]

Byte length of the actual IR command name string without terminator.

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_DEVICE_NAME_ARR_UNDERSIZE	The size of the device name array variable is too small as an argument of a function.
IR_E_IRCMD_NAME_ARR_UNDERSIZE	The size of the IR cmd name array variable is too small as an argument of a function.

Examples:

```
[VC++]
int Ret = 0;
unsigned short usIrCmdNumber = 1;
unsigned char ucIrDevNameArr[100] = {0};
unsigned short usDevNameByteArrLen = 100;
unsigned short usDevNameByteArrOutLen = 0;
unsigned char ucIrCmdNameArr[100] = {0};
```

```
unsigned short usIrCmdNameByteArrLen = 100;
unsigned short usCmdNameStrByteLen = 0;

Ret = IR_GetIrDevCmdNameInBuffer(
        usIrCmdNumber,
        ucIrDevNameArr,
        usDevNameByteArrLen,
        &usDevNameByteArrOutLen,
        ucIrCmdNameArr,
        usIrCmdNameByteArrLen,
        &usCmdNameStrByteLen
        );
```

3.1.11 IR_SetIrDevCmdNameInBuffer

Description:

Set IR device name and IR command name to the buffer of the API.

Prototype:

[VC++]

```
int IR_SetIrDevCmdNameInBuffer(
    unsigned short usIrCmdNumber,
    unsigned char *ucIrDevNameArr,
    unsigned short usDevNameUcLen,
    unsigned char *ucIrCmdNameArr,
    unsigned short usIrCmdNameUcLen
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_SetIrDevCmdNameInBuffer(
    UInt16 usIrCmdNumber,
    [MarshalAs(UnmanagedType.LPArray, SizeParamIndex = 2)] Byte[]
    ucIrDevNameStr,
    UInt16 usDevNameUcLen,
    [MarshalAs(UnmanagedType.LPArray, SizeParamIndex = 4)] Byte[]
    ucIrCmdNameStr,
    UInt16 usIrCmdNameUcLen
    );
```

Parameters:

usIrCmdNumber [in]

IR command number for allocating one IR learning command data in the learning data buffer.

ucIrDevNameArr [in]

Byte array for the IR device Name string. Byte Array Max length = 100 bytes (50 words).

usDevNameUcLen [in]

Byte length of the IR devie name array (ucIrDevNameArr). Max length = 100 bytes (50 words).

ucIrCmdNameArr [in]

Byte array for the IR Command Name. Byte array max length = 102 bytes (51 words).

usIrCmdNameUcLen [in]

Byte length of the IR command name array (ucIrCmdNameArr). Max length = 102 bytes (51 words).

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_DEVICE_NAME_ARR_OVERSIZE	The length of the device name is too large as an argument to the function.
IR_E_IRCMD_NAME_ARR_OVERSIZE	The length of the IR command name is too large as an argument to the function.

Examples:

```
[VC++]
int Ret = 0;
unsigned short usIrCmdNumber = 1;
unsigned char ucIrDevNameArr[2] = {0x54, 0x56};  // TV
unsigned short usDevNameUcLen = 2;
unsigned char ucIrCmdNameArr[4] = {86, 79, 76, 43};  // VOL+
unsigned short usIrCmdNameUcLen = 4;

Ret = IR_SetIrDevCmdNameInBuffer(
        usIrCmdNumber,
        ucIrDevNameArr,
        usDevNameUcLen,
        ucIrCmdNameArr,
        usIrCmdNameUcLen,
        );
```

3.2 Common Functions for IR Modules with Modbus TCP/UDP

3.2.1 IR_ChkNIC

Description:

Check the number of available network interfaces of the host PC.

Prototype:

[VC++]

```
void IR_ChkNIC(
    int *iNiCnt
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern void IR_ChkNIC(
    ref int iNiCnt
    );
```

Parameters:

iNiCnt [out]

number of available network interface

Return Values:

None

Examples:

[VC++]

```
int iNiCnt = 0;

IR_ChkNIC(&iNiCnt);
printf("The number of avaliable NIC = %d", iNiCnt);
```

3.2.2 IR_GetNICInfo

Description:

Get the description and IR address of available network interfaces.

Prototype:

[VC++]

```
void IR_GetNICInfo(
        unsigned char ucIndex,
        unsigned char *szDescStr,
        unsigned short usDescStrLen,
        unsigned short *ucRtnDecrLen,
        unsigned long *ulIPAddr
        );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern void IR_GetNICInfo(
        Byte ucIndex,
       [MarshalAs(UnmanagedType.LPArray, SizeParamIndex = 2)] Byte[] szDescStr,
        int usDescStrLen,
        ref int ucRtnDecrLen,
        ref UInt32 ulIPAddr
        );
```

Parameters:

ucIndex [in]

index(0-based) of the available NICs in the API buffer for getting the NIC info

szDescStr [out]

NIC description string

usDescStrLen [in]

Length of szDescStr array. (number of unsigned char; maximum = 133)

usDescStrLen [out]

return the length of NIC description (excluding '\0')

ulIPAddr [out]

IP address of the network interface card (e.g. 0xC0A8FF0F => 192.168.255.15)

Return Values:

None

Examples:

```
[VC++]
int iNiCnt = 0;
unsigned char description[133] = {0};
unsigned short actualLen = 0;
unsigned long IPAddr = 0;
unsigned char NB0, NB1, NB2, NB3;  //Network address

IR_ChkNIC(&iNiCnt);
printf("The number of avaliable NIC = %d", iNiCnt);

for (int i = 0; i < iNiCnt; i++)
{
   IR_GetNICInfo((unsigned char)i, description, sizeof(description),
&actualLen, &IPAddr);
   NB0 = (unsigned char)(IPAddr >> 24);
   NB1 = (unsigned char)(IPAddr >> 16);
   NB2 = (unsigned char)(IPAddr >> 8);
   NB3 = (unsigned char)IPAddr;
   printf("IP Address = %d.%d.%d.%d \nDescription = %s", NB0, NB1,
NB2, NB3, description);
}
```

3.2.3 IR_EthSearchModules

Description:

Get the number of available IR modules on the same subnet.

Prototype:

[VC++]

```
int IR_EthSearchModules (
    unsigned long ulIPAddrNIC,
    unsigned char *pucModuleCnt,
    unsigned short usWaitTimeout
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern int IR_EthSearchModules(
    UInt32 ulIPAddrNIC,
    ref Byte pucModuleCnt,
    UInt16 usTimeout
    );
```

Parameters:

ulIPAddrNIC [in]

IP address of the network interface of a host PC.

pucModuleCnt [out]

count of found modules

usWaitTimeout [in]

timeout(ms) to wait for searching modules. (e.g. 2000 ms)

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_FUNC_ERR	Function error.

Examples:

```
[VC++]
unsigned char moduleCnt = 0;
unsigned long NicIPAddr = 0xC0A8007D;  // ETH NIC = 192.168.0.125

IR_EthSearchModules(NicIPAddr, &moduleCnt, 2000);
printf("Searched IR module count = %d", moduleCnt);
```

3.2.4 IR_EthGetSearchModuleInfo

Description:

Get the information of the searched modules. IR_EthSearchModules() should be called in advance.

Prototype:

[VC++]

```
void IR_EthGetSearchModuleInfo(
    unsigned short usCntIndex,
    SearchRespEth *ModuleInfo
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern void IR_EthGetSearchModuleInfo(
    UInt16 usCntIndex,
    ref SearchRespEth ModuleInfo
    );
```

Parameters:

usCntIndex [in]

zero-based index of found modules. If found total module count is 3, available usCntIndex == 0, 1, 2.

ModuleInfo [out]

output variable of struct SearchRespEth

This structure collects the information of a searched IR module with Ethernet interface.

Definition of *SearchRespEth*:

[VC++]

```
typedef struct
{
  unsigned __int64 dwlMacAddr;
  unsigned long  ulIPAddr;
  unsigned long  ulMaskAddr;
```

```
  unsigned long  ulGWAddr;
  unsigned long  ulFwVer;
  unsigned long  ulIdleTOut;
  unsigned char  ucNetID;
  unsigned char  ucAddrType;
  unsigned char  ucHwMode;
  unsigned char  ucModIndex;
} SearchRespEth;
```

Where the structure members are defined as follows:

Table 3-1: Members of the BasicSetParams structure

Members	Description
dwlMacAddr	MAC address. E.g. 1a:2b:3c:4d:5e:6f => 0x00001A2B3C4D5E6F
ulIPAddr	IP Addres. E.g. 192.168.0.1 => 0xC0A80001
ulMaskAddr	Mask Addres. E.g. 255.255.0.0 => 0xFFFF0000
ulGWAddr	Gateway address. E.g. 192.168.0.254 => 0xC0A800FE
ulFwVer	Firmware version of the module. E.g. v1.2.3 => 0x00010203
ulIdleTOut	Idle communication timeout (unit: mimnute, 0 ~ 65535)
ucNetID	Modbus Net ID of IR modules (1 ~ 247)
ucAddrType	Address type. 0: static 1: DHCP
ucHwMode	Mode: OP(=0) or FW/Init (=1) mode
ucModIndex	Module Index. 0x21 => "IR-712-MTCP"

[VC#.NET]

```
[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack = 1)]
public struct SearchRespEth
{
  public UInt64 dwlMacAddr;
  public UInt32 ulIPAddr;
  public UInt32 ulMaskAddr;
  public UInt32 ulGWAddr;
  public UInt32 ulFwVer;
  public UInt32 ulIdleTOut;
  public Byte ucNetID;
  public Byte ucAddrType;
```

```
    public Byte ucHwMode;
    public Byte ucModIndex;
  };
```

Return Values:

None

Examples:

```
[VC++]
Unsigned short mInfoIndex = 0;
SearchRespEth moduleInfoStruct;

IR_EthGetSearchModuleInfo(
    mInfoIndex,
    & moduleInfoStruct
    );

printf("Module Index = %d\n", moduleInfoStruct.ucModIndex);
```

4. Functions for Modbus RTU Communication

4.1 Communication Function

4.1.1 IR_GetComPortStatus

Description:

Check if the COM port is in use or not.

Prototype:

[VC++]

```
int IR_GetComPortStatus (
    unsigned char ucPort,
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_GetComPortStatus (
    Byte ucPort,
    );
```

Parameters:

ucPort [in]

Serial COM port number.

Return Values:

Return Code	Description
IR_E_COM_STATUS_AVAIL	COM port is available and not open.
IR_E_COM_STATUS_ERR	COM port is not available or is open.

Examples:

```
[VC++]
int Ret = 0;
unsigned char  ucComPort = 1

Ret = IR_GetComPortStatus(ucComPort);
```

4.1.2 IR_OpenCom

Description:

Open a COM port of the host device (Modbus Master).

Prototype:

[VC++]

```
int IR_OpenCom(
    unsigned char ucPort,
    unsigned char ucBaudIndex,
    unsigned char ucData,
    unsigned char ucParity,
    unsigned char ucStop
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_OpenCom(
    Byte ucPort,
    Byte ucBaudIndex,
    Byte ucData,
    Byte ucParity,
    Byte ucStop
    );
```

Parameters:

ucPort [in]

Serial COM port number.

ucBaudIndex [in]

Please refer to Table 4-1 in section 4.2.1.

ucData [in]

Please refer to Table 4-1 in section 4.2.1.

ucParity [in]

Please refer to Table 4-1 in section 4.2.1.

ucStop [in]

Please refer to Table 4-1 in section 4.2.1.

Return Values:

Return Code	Description
IR_E_OK	Open Com OK.
IR_E_ILL_BAUD_IDX	COM port is not available or is open.
IR_E_ILL_DATA_BIT	illegal data bits
IR_E_ILL_PARITY	illegal parity
IR_E_ILL_STOP_BIT	illegal stop bits
IR_E_OPEN_COM_ERR	Open COM error where COM is open or is not available.

Examples:

```
[VC++]
int Ret = 0;
Ret = IR_OpenCom(
    6,   // Com port number
    10,  // Baud rate index == 10 (115200 bps)
    8,   // Data bits is always 8.
    0,   // 0 => Parity None
    0    // 0 => stop bits 1
    );
```

4.1.3 IR_CloseCom

Description:

Close a COM port of the host device (Modbus Master).

Prototype:

[VC++]

```
int IR_CloseCom(
    unsigned char ucPort,
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_CloseCom(
    Byte ucPort,
    );
```

Parameters:

ucPort [in]

Serial COM port number.

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_FUNC_ERR	Close function errors

Examples:

```
[VC++]
int Ret = 0;
Ret = IR_CloseCom(6);
```

4.1.4 IR_GetModbusRecvTimeout

Description:

Get the timeout (ms) for waiting the Modbus response from IR module.

Prototype:

[VC++]

```
int IR_GetModbusRecvTimeout(void);
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_GetModbusRecvTimeout();
```

Parameters:

None.

Return Values:

Receive Modbus response timeout (ms). Default value =200 ms.

Examples:

```
[VC++]
int tout;
tout = IR_GetModbusRecvTimeout();
```

4.1.5 IR_SetModbusRecvTimeout

Description:

Set the timeout (ms) for waiting the Modbus response from IR module.

Prototype:

[VC++]

```
void IR_SetModbusRecvTimeout(
    unsigned short recvTout
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern void IR_SetModbusRecvTimeout(UInt16 recvTout);
```

Parameters:

recvTout [in]

Receive timeout for the Modbus response after sending a Modbus request. Default value is 50 ms.

Return Values:

None

Examples:

```
[VC++]
unsigned short recvTimeout = 100;  // 100 ms
IR_SetModbusRecvTimeout(recvTimeout);
```

4.2 Basic Settings Function

4.2.1 IR_GetBasicSetting

Description:

Get basic settings from the IR module.

Prototype:

[VC++]

```
int IR_GetBasicSetting(
    unsigned char ucPort,
    unsigned char ucNetID,
    BasicSetParams* pSBasicSetting
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_GetBasicSetting(
    Byte ucPort,
    Byte ucNetID,
    ref BasicSetParams SBasicSetting
    );
```

Parameters:

ucPort [in]

Serial COM port number.

ucNetID [In]

Modbus net ID of the IR module (1 ~ 247).

pSBasicSetting [out]

Pointer to the structure variable of the bsaic settings of the IR learning remote module.

Definition of *SBasicSetting*:

[VC++]

```
typedef struct
```

```
{
  unsigned char  ucBaudrateIndex;
  unsigned char  ucData;
  unsigned char  ucParity;
  unsigned char  ucStop;
  unsigned char  ucNetID;
  unsigned short  wMBRespDlyT;
  unsigned short  wGapTime;
}BasicSetParams;
```

Where the structure members are defined as follows:

Table 4-1: Members of the BasicSetParams structure

Members	Description
ucBaudrateIndex	Baud rate Index. For [index]: baud rate (bps) => [6]:9600 bps, [7]:19200 bps, [8]:38400 bps, [9]:57600 bps, [10]:115200 bps
ucData	Data bits. Should be always set to 8.
ucParity	Parity. 0=>None 1=>Odd 2=>Even
ucStop	Stop bits. 1=>1 stopbit, 2=>2 stopbits
ucNetID	Modbus Net ID of IR modules (1 ~ 247)
wMBRespDlyT	Modbus response delay time (ms). Default = 3 ms
wGapTime	Gap time (ms), default= 72 ms, range = 72 ~ 200 ms

```
[VC#.NET]
[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack = 1)]
public struct BasicSetParams
{
  public Byte ucBaudrateIndex;
  public Byte ucData;
  public Byte ucParity;
  public Byte ucStop;
  public Byte ucNetID;
  public UInt16 wMBRespDlyT;
  public UInt16 wGapTime;
};
```

Return Values:

Return Code	Description
IR_E_OK	No Error
IR_E_ILL_BAUD_IDX	Illegal baud rate
IR_E_ILL_DATA_BIT	Illegal data bits
IR_E_ILL_PARITY	Illegal parity
IR_E_ILL_STOP_BIT	Illegal stop bits
IR_E_SEND_MBCMD_ERR	Error in reading basic settings from the IR module.

Examples:

```
[VC++]
int Ret = 0;
unsigned char  ucComPort = 1;
unsigned char  ucMbID = 1;
BasicSetParams sBasicSet;

Ret = IR_GetBasicSetting (
     ucComPort,
     ucMbID,
     &sBasicSet
     );

printf("GapTime = %d\n", sBasicSet.wGapTime);
```

4.2.2 IR_SetBasicSetting

Description:

Set basic settings to the IR module.

Take care the times to call this function in programming because there are max 100000 times to write basic setings to the flash memory.

Prototype:

[VC++]

```
int IR_SetBasicSetting(
    unsigned char ucPort,
    unsigned char ucNetID,
    BasicSetParams SBasicSetting
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_SetBasicSetting(
    Byte ucPort,
    Byte ucNetID,
    [in]BasicSetParams SBasicSetting
    );
```

Parameters:

ucPort [in]

Serial COM port number.

ucNetID [in]

Modbus ID of the IR module (1 ~ 247).

SBasicSetting [in]

The structure variable of the basic settings of the IR learning remote module.

Definition of *SBasicSetting*:

[VC++]

Introduced in 4.2.1 IR_GetBasicSetting.

[VC#.NET]

Introduced in 4.2.1 IR_GetBasicSetting.

Return Values:

Return Code	Description
IR_E_OK	No Error
IR_E_ILL_BAUD_IDX	Illegal baud rate
IR_E_ILL_DATA_BIT	Illegal data bits
IR_E_ILL_PARITY	Illegal parity
IR_E_ILL_STOP_BIT	Illegal stop bits
IR_E_SEND_MBCMD_FLASH_ERR	Error in sending Modbus command to write data to flash memory.

Examples:

```
[VC++]
int Ret = 0;
unsigned char  ucComPort = 1
unsigned char  ucMbID = 3;
BasicSetParams sBasicSet;

sBasicSet.ucBaudrateIndex = 10;  // [10]115200 bps
sBasicSet.ucData = 8;            // Data bits is always 8
sBasicSet.ucParity = 0;          // Parity := None
sBasicSet.ucStop = 0;            // Stop bits = 0
sBasicSet. ucNetID = 1;          // MB Net ID of the IR module
sBasicSet.wMBRespDlyT = 3;       // MB response delay time
sBasicSet.wGapTime = 72;         // Gaptime for IR learning

Ret = IR_SetBasicSetting(
     ucComPort,
     ucMbID,
     sBasicSet
     );
```

4.2.3 IR_ResetIRModule

Description:

Reset Communication settings of the IR module. Wait for (block) three seconds.

Prototype:

[VC++]

```
int IR_ResetIRModule (
    unsigned char ucPort
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_ResetIRModule(
    Byte ucPort
    );
```

Parameters:

ucPort [in]

Serial COM port number.

Return Values:

Return Code	Description
IR_E_RESET_OK	Reset IR module successfully.
IR_E_RESET_COM_ERR	COM error in sending reset message to the IR module. Receive timeout or COM error.
IR_E_RESET_MODULE_FAIL	Failed to reset the IR module.

Examples:

```
[VC++]
int Ret = 0;
unsigned char ucComPort = 1

Ret = IR_ ResetIRModule(ucComPort);
```

4.3 IR Learning Function

4.3.1 IR_GetIRCmdSummary

Description:

Get IR Command Summary from the IR module. The IR command summary is also saved to the buffer of the IR API Library.

Prototype:

[VC++]

```
int IR_GetIRCmdSummary(
    unsigned char ucPort,
    unsigned char usNetID,
    unsigned short usIrModuleIndex,
    unsigned short *dataArr,
    unsigned short usInElementCnt
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_GetIRCmdSummary (
    Byte ucPort,
    Byte usNetID,
    UInt16 usIrModuleIndex,
    [MarshalAs(UnmanagedType.LPArray, SizeParamIndex = 4)] UInt16[] dataArr,
    UInt16 wOutElementLen
    );
```

Parameters:

ucPort [in]

Serial COM port number.

usNetID [in]

Modbus Net ID of the IR module.

usIrModuleIndex [in]

Index of IR learning remote module. 0x01: IR-210, 0x11: IR-712A

dataArr [out]

(unsigned short)data array for IR command summary. Array length = 13 where

dataArr[0]: IR device quantity (1 ~ 11)

dataArr[1]: Total quantity of IR commands (1 ~ 224)

dataArr[2]: IR command quantity of device#1

dataArr[3]: IR command quantity of device#2

....

dataArr[N+1]: IR command quantity of device#N , where N<=11

usInElementCnt [in]

Length of the dataArr array. Should be at least 13.

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_ILL_SUMMARY_LEN	Illegal length of the array (dataArr) for IR command summary.
IR_E_FUNC_ERR	Function error.
IR_E_IRCMD_TOTAL_QTY_ERR	The device quantity setting exceeds the maximum.
IR_E_IRCMD_SUM_QTY_ERR	The element "Total quantity of IR commands" in the IR command summary exceeds the maximum.
IR_E_ILL_MODULE_INDEX	Illegal module index.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.

Examples:

```
[VC++]
int Ret = 0;
unsigned char ucPort = 0,
              ucNetID = 0;
unsigned short usIrModuleIndex = 0,
               dataArr[13] = {0},
               usInElementCnt = 13;

Ret = IR_SetIRCmdSummary(ucPort, ucNetID, usIrModuleIndex, dataArr,
usInElementCnt);
Ret = IR_GetIRCmdSummary(ucPort, ucNetID, usIrModuleIndex);
```

4.3.2 IR_SetIRCmdSummary

Description:

Set IR Command Summary to the IR module.

!Note: IR_SetIRCmdSummaryBuffer() should be called first.

Prototype:

```
[VC++]
int IR_SetIRCmdSummary(
    unsigned char ucPort,
    unsigned char usNetID,
    unsigned short usIrModuleIndex
    );

[VC#.NET]
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_SetIRCmdSummary(
    Byte ucPort,
    Byte usNetID,
    UInt16 usIrModuleIndex
    );
```

Parameters:

ucPort [in]

Serial COM port number.

usNetID [in]

Modbus Net ID of the IR module.

usIrModuleIndex [in]

Index of IR learning remote module. 0x01: IR-210, 0x11: IR-712A

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_FUNC_ERR	Function error.

IR_E_DEVICE_QTY_ERR	The device quantity exceeds the maximum.
IR_E_IRCMD_TOTAL_QTY_ERR	The device quantity setting exceeds the maximum.
IR_E_IRCMD_SUM_QTY_ERR	The element "Total quantity of IR commands" in the IR command summary exceeds the maximum.
IR_E_ILL_MODULE_INDEX	Illegal module index.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.
IR_E_SEND_MBCMD_FLASH_ERR	Error in sending/receiving Modbus command to write data to Flash.

Examples:

```
[VC++]
int Ret = 0;
unsigned char ucPort = 1,
              ucNetID = 1;
unsigned short usIrModuleIndex = 0x01, // Module index for IR-210
               dataArr[13] = {2, 18, 10, 8},
               usInElementCnt = 13;    // length of dataArr

Ret = IR_SetIRCmdSummaryBuffer(usIrModuleIndex, dataArr,
usInElementCnt);
Ret = IR_SetIRCmdSummary(ucPort, ucNetID, usIrModuleIndex);
```

4.3.3 IR_SetIRCmdSummary1

Description:

Set IR Command Summary to the IR module. The IR command summary is also written to the buffer of the API library.

Prototype:

```
[VC++]
int IR_SetIRCmdSummary1(
    unsigned char ucPort,
    unsigned char usNetID,
    unsigned short usIrModuleIndex,
    unsigned short *dataArr,
    unsigned short usInElementCnt
    );

[VC#.NET]
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_SetIRCmdSummary1(
    Byte ucPort,
    Byte usNetID,
    UInt16 usIrModuleIndex,
    [MarshalAs(UnmanagedType.LPArray, SizeParamIndex = 4)] UInt16[] dataArr,
    UInt16 wOutElementLen
    );
```

Parameters:

ucPort [in]

Serial COM port number.

usNetID [in]

Modbus Net ID of the IR module.

usIrModuleIndex [in]

Index of IR learning remote module. 0x01: IR-210, 0x11: IR-712A

dataArr [in]

(unsigned short)data array for IR command summary. Array length = 13
where
dataArr[0]: IR device quantity (1 ~ 11)
dataArr[1]: Total quantity of IR commands (1 ~ 224)
dataArr[2]: IR command quantity of device#1
dataArr[3]: IR command quantity of device#2
....
dataArr[N+1]: IR command quantity of device#N , where N<=11

usInElementCnt [in]

Length of the dataArr array. Should be at least 13.

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_ILL_SUMMARY_LEN	Illegal length of the array(dataArr) for IR command summary.
IR_E_FUNC_ERR	Function error.
IR_E_DEVICE_QTY_ERR	The device quantity exceeds the maximum.
IR_E_IRCMD_TOTAL_QTY_ERR	The device quantity setting exceeds the maximum.
IR_E_IRCMD_SUM_QTY_ERR	The element "Total quantity of IR commands" in the IR command summary exceeds the maximum.
IR_E_ILL_MODULE_INDEX	Illegal module index.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.
IR_E_SEND_MBCMD_FLASH_ERR	Error in sending/receiving Modbus command to write data to Flash.

Examples:

```
[VC++]
int Ret = 0;
unsigned char ucPort = 1,
              ucNetID = 1;
unsigned short usIrModuleIndex = 0x01, // Module index for IR-210
               dataArr[13] = {2, 18, 10, 8},
               usInElementCnt = 13;

Ret = IR_SetIRCmdSummary(ucPort, ucNetID, usIrModuleIndex, dataArr,
    usInElementCnt);
```

4.3.4 IR_GetIRCmdSummaryBuffer

Description:

Get the IR command summary from the buffer of the API library.

Prototype:

[VC++]

```
int IR_GetIRCmdSummaryBuffer (
    unsigned short usIrModuleIndex,
    unsigned short *dataArr,
    unsigned short usInElementCnt
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_GetIRCmdSummaryBuffer (
    UInt16 usIrModuleIndex,
    [MarshalAs(UnmanagedType.LPArray, SizeParamIndex = 2)] UInt16[] dataArr,
    UInt16 usInElementCnt
    );
```

Parameters:

usIrModuleIndex [in]

Index of IR learning remote module.
0x01: IR-210, 0x11: IR-712A, 0x21: IR-712-MTCP

dataArr [in]

(unsigned short)data array for IR command summary. Array length = 13
where
dataArr[0]: IR device quantity (1 ~ 11)
dataArr[1]: Total quantity of IR commands (1 ~ 224)
dataArr[2]: IR command quantity of device#1
dataArr[3]: IR command quantity of device#2
....
dataArr[N+1]: IR command quantity of device#N , where N<=11

usInElementCnt [in]

Length of the dataArr array. Should be at least 13.

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_ILL_SUMMARY_LEN	Illegal length of the array(dataArr) for IR command summary.
IR_E_FUNC_ERR	Function error.
IR_E_DEVICE_QTY_ERR	The device quantity exceeds the maximum.
IR_E_IRCMD_TOTAL_QTY_ERR	The device quantity setting exceeds the maximum.
IR_E_IRCMD_SUM_QTY_ERR	The element "Total quantity of IR commands" in the IR command summary exceeds the maximum.
IR_E_ILL_MODULE_INDEX	Illegal module index.

Examples:

```
[VC++]
int Ret = 0;
unsigned short usIrModuleIndex = 0x01, // Module index for IR-210
               dataArr[13] = {0},
               usInElementCnt = 13;    // length of dataArr

Ret = IR_GetIRCmdSummaryBuffer(usIrModuleIndex, dataArr,
usInElementCnt);
```

4.3.5 IR_SetIRCmdSummaryBuffer

Description:

Set the IR command summary to the buffer of the API library.

Prototype:

[VC++]

```
int IR_SetIRCmdSummaryBuffer (
    unsigned short usIrModuleIndex,
    unsigned short *dataArr,
    unsigned short usInElementCnt
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_SetIRCmdSummaryBuffer (
    UInt16 usIrModuleIndex,
    [MarshalAs(UnmanagedType.LPArray, SizeParamIndex = 2)] UInt16[] dataArr,
    UInt16 usInElementCnt
    );
```

Parameters:

usIrModuleIndex [in]

Index of IR learning remote module.
0x01: IR-210, 0x11: IR-712A, 0x21: IR-712-MTCP

dataArr [in]

(unsigned short)data array for IR command summary. Array length = 13
where
dataArr[0]: IR device quantity (1 ~ 11)
dataArr[1]: Total quantity of IR commands (1 ~ 224)
dataArr[2]: IR command quantity of device#1
dataArr[3]: IR command quantity of device#2
....
dataArr[N+1]: IR command quantity of device#N , where N<=11

usInElementCnt [in]

Length of the dataArr array. Should be at least 13.

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_ILL_SUMMARY_LEN	Illegal length of the array(dataArr) for IR command summary.
IR_E_FUNC_ERR	Function error.
IR_E_DEVICE_QTY_ERR	The device quantity exceeds the maximum.
IR_E_IRCMD_TOTAL_QTY_ERR	The device quantity setting exceeds the maximum.
IR_E_IRCMD_SUM_QTY_ERR	The element "Total quantity of IR commands" in the IR command summary exceeds the maximum.
IR_E_ILL_MODULE_INDEX	Illegal module index.

Examples:

```
[VC++]
int Ret = 0;
unsigned char ucPort = 1,
              ucNetID = 1;
unsigned short usIrModuleIndex = 0x01, // Module index for IR-210
               dataArr[13] = {2, 18, 10, 8},
               usInElementCnt = 13;    // length of dataArr

Ret = IR_SetIRCmdSummaryBuffer(usIrModuleIndex, dataArr,
usInElementCnt);
Ret = IR_SetIRCmdSummary(ucPort, ucNetID, usIrModuleIndex);
```

4.3.6 IR_LrnModeSet

Description:

Enable/Disable the IR learning mode.

Prototype:

[VC++]

```
int IR_LrnModeSet (
    unsigned char ucPort,
    unsigned char ucNetID,
    unsigned char ucMode
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_LrnModeSet (
    Byte ucPort,
    Byte ucNetID,
    Byte ucMode
    );
```

Parameters:

ucPort [in]

COM port number.

ucNetID [in]

Modbus net ID of the IR module (1 ~ 247).

ucMode [in]

Learning mode of the IR module. 0 => learn off, 1=> learn on.

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.

Examples:

```
[VC++]
int Ret = 0;
unsigned char ucPort = 1,
              ucNetID = 1;
unsigned char lrnMode = 1;  // Learn ON mode

Ret = IR_LrnModeSet(ucPort, ucNetID, lrnMode);
```

4.3.7 IR_IsIRLearnOK

Description:

Check if learning an IR command is finished.

Prototype:

[VC++]

```
int IR_IsIRLearnOK(
    unsigned char ucPort,
    unsigned short *pusIsOK,
    unsigned short usTimeout
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_IsIRLearnOK(
    Byte ucPort,
    ref UInt16 pusIsOK,
    UInt16 usTimeout
    );
```

Parameters:

ucPort [in]

COM port number.

pusIsOK [out]

0=>IR learning not OK, 1=>IR learning OK

usTimeout [in]

Receive timeout (ms).

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_COM_STATUS_ERR	The COM port is not available or is open.

Examples:

```
[VC++]
int Ret = 0;
unsigned short isLrnOK = 0,
               recvTout = 50;

Ret = IR_IsIRLearnOK(1, &isLrnOK, recvTout);
```

4.3.8 IR_GetIRCmdAfterLearn

Description:

Get one IR command learning data (without device name & IR cmd name) from the IR learning remote module to the buffer of the API library right after learning an IR command successfully.
(IR cmd data := IR device name + IR cmd name + IR cmd learning data)

Prototype:

[VC++]

```
int IR_GetIRCmdAfterLearn (
    unsigned char ucPort,
    unsigned char ucNetID,
    unsigned short usIrCmdNumber
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_ GetIRCmdAfterLearn(
    Byte ucPort,
    Byte ucNetID,
    UInt16 usIrCmdNumber
    );
```

Parameters:

ucPort [in]

COM port number.

ucNetID [in]

Modbus Net ID of the IR module.

usIrCmdNumber [in]

IR command number (1 ~ 224).

Return Values:

Return Code	Description
IR_E_OK	No errors.

IR_E_ILL_IRCMD_NO	Illegal IR command number.
IR_E_SEND_MBCMD_ERR	No response.

Examples:

```
[VC++]
int Ret = 0;
Ret = IR_GetIRCmdAfterLearn(1, 1, 10);  // Read IR cmd number 10.
```

4.4 Download\Load IR Command Functions

4.4.1 IR_DLOneIRDataToModule

Description:

Download one IR cmd data (:= IR device name + IR cmd name + IR cmd learning data) from the API buffer to the IR Module.
(IR cmd data := IR device name + IR cmd name + IR cmd learning data)

Prototype:

[VC++]

```
int IR_DLOneIRDataToModule(
    unsigned char ucPort,
    unsigned char ucNetID,
    unsigned short usIrCmdNumber
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_DLOneIRDataToModule (
    Byte ucPort,
    Byte ucNetID,
    UInt16 usIrCmdNumber
    );
```

Parameters:

ucPort [in]

COM port number

usNetID [in]

Modbus Net ID of the IR module.

usIrCmdNumber [in]

IR command number (1 ~ 224).

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_ILL_IRCMD_NO	Illegal IR command number
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.
IR_E_SEND_MBCMD_FLASH_ERR	Error in sending/receiving Modbus command to write data to Flash.
IR_E_LRNCMD_LEN_EXCEED_MAX	The length of the IR cmd learning data exceeds the maximum.

Examples:

```
[VC++]
int Ret = 0;

// Download IR command data #3 to the IR module.
Ret = IR_DLOneIRDataToModule (1, 1, 3);
```

4.4.2 IR_LDOneIRDataToAPI

Description:

Load one IR command data from the IR module and store it into the API buffer.
(IR cmd data := IR device name + IR cmd name + IR cmd learning data)

Prototype:

[VC++]

```
int IR_LDOneIRDataToAPI(
    unsigned char ucPort,
    unsigned char ucNetID,
    unsigned short usIrCmdNumber
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_LDOneIRDataToAPI(
    Byte ucPort,
    Byte ucNetID,
    UInt16 usIrCmdNumber
    );
```

Parameters:

ucPort [in]

COM port number

usNetID [in]

Modbus Net ID of the IR module.

usIrCmdNumber [in]

IR command number (1 ~ 224).

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_ILL_IRCMD_NO	Illegal IR command number
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.

Examples:

```
[VC++]
int Ret = 0;

// Load IR command data #3 from the IR module and store it
// in the API buffer.
Ret = IR_LDOneIRDataToAPI(1, 1, 3);
```

4.5 IR Emitting Function

4.5.1 IR_EmitIrSignal

Description:

Request the IR module to emit the IR command stored in the flash memory.

Prototype:

[VC++]

```
int IR_EmitIrSignal(
    unsigned char ucPort,
    unsigned char ucNetID,
    unsigned short usIrCmdNumber,
    unsigned char ucIrOutputCh
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_EmitIrSignal(
    Byte ucPort,
    Byte ucNetID,
    UInt16 usIrCmdNumber,
    Byte ucIrOutputCh
    );
```

Parameters:

ucPort [in]

COM port number

usNetID [in]

Modbus Net ID of the IR module.

usIrCmdNumber [in]

IR command number (1 ~ 224).

ucIrOutputCh [in]

IR output channels.

E.g., use IR output ch1 & ch2 => value = 0x03 (0000 0011(binary))

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_ILL_IRCMD_NO	Illegal IR command number
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.

Examples:

```
[VC++]
int Ret = 0;

// Request IR module to emit the IR command data #3 by the IR
// output channel 2.
Ret = IR_EmitIrSignal(1, 1, 3, 0x02);
```

4.5.2 IR_RunCommand

Description:

Send an IR learning command data to the IR module and emit that IR command.

Prototype:

[VC++]

```
int IR_RunCommand(
    unsigned char ucPort,
    unsigned char ucNetID,
    unsigned short usIrCmdNumber,
    unsigned char ucIrOutputCh
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_RunCommand(
    Byte ucPort,
    Byte ucNetID,
    UInt16 usIrCmdNumber,
    Byte ucIrOutputCh
    );
```

Parameters:

ucPort [in]

COM port number

usNetID [in]

Modbus Net ID of the IR module.

usIrCmdNumber [in]

IR command number (1 ~ 224).

ucIrOutputCh [in]

IR output channels.

E.g., use IR output ch1 & ch2 => value = 0x03 (0000 0011(binary))

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_IRCMD_DATA_EMPTY	NO IR data exists for the specified IR cmd number in the API buffer.
IR_E_ILL_IRCMD_NO	Illegal IR command number
IR_E_LRNCMD_LEN_EXCEED_MAX	The length of the IR cmd learning data exceeds the maximum.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.

Examples:

```
[VC++]
int Ret = 0;

Ret = IR_RunCommand(1, 1, 3, 0x02);
```

4.6 Others Function

4.6.1 IR_ChkModuleName

Description:

Get model name of the IR remote module.

Prototype:

[VC++]

```
int IR_ChkModuleName(
    unsigned char ucPort,
    unsigned char ucNetID,
    unsigned short *pwModuleIndex
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_ChkModuleName(
    Byte ucPort,
    Byte ucNetID,
    ref UInt16 pwModuleIndex
    );
```

Parameters:

ucPort [in]

COM port number

usNetID [in]

Modbus Net ID of the IR module.

pwModuleIndex [out]

Module index representing for specific IR module type(name).
e.g. 0x01=>"IR210",
0x11=>"IR712A".

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.
IR_E_MODNAME_INVALID	Get invalid module index (name).

Examples:

```
[VC++]
int Ret = 0;
unsigned short ModulIndex = 0;

Ret = IR_ChkModuleName(1, 1, &ModulIndex);

switch(ModuleIndex)
{
case 0x01:  // The IR module is IR-210.
    break;
case 0x11:  // The IR module is IR-712A.
    break;
default:
    break;
}
```

5. Functions for Modbus TCP Communication

5.1 Communication Functions

5.1.1 IR_MTCP_TCPNew

Description:

Initialize a TCP socket for Modbus TCP communication.

Prototype:

[VC++]

```
tHandle IR_MTCP_TCPNew (
    void
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MTCP_TCPNew();
```

Parameters:

None

Return Values:

tHandle: TCP socket handle;

If successful, a TCP socket handle is returned.

If not successful, IR_E_SOCKET_INVALID is returned.

Examples:

```
[VC++]
tHandle socketHandle;
socketHandle = IR_MTCP_TCPNew();
```

5.1.2 IR_MTCP_TCPNew1

Description:

Initialize a TCP socket bound to an NIC for Modbus TCP.

Prototype:

[VC++]

```
tHandle IR_MTCP_TCPNew1 (
    unsigned long ulNicIPAddr
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MTCP_TCPNew1 (
    UInt32 ulNicIPAddr
    );
```

Parameters:

ulNicIPAddr [in]

IP address of a network card for binding. Network byte order. IP: 0xC0A80001 (=>192.168.0.1)

Return Values:

tHandle: TCP socket handle;

If successful, a TCP socket handle is returned.

If not successful, IR_E_SOCKET_INVALID is returned.

Examples:

```
[VC++]
tHandle sHandle;
unsigned long ulNICIpAddr = 0xC0A8000A;  // 192.168.0.10
sHandle = IR_MTCP_TCPNew1(ulNICIpAddr);
```

5.1.3 IR_MTCP_Connect

Description:

Connect to the IR module with Modbus TCP protocol.

Prototype:

[VC++]

```
int IR_MTCP_Connect (
    tHandle hSocketTCP,
    unsigned long ulIPAddr,
    unsigned char ucNetID,
    unsigned short timeoutMs
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern int IR_MTCP_Connect (
    int hSocketTCP,
    UInt32 ulIPAddr,
    Byte ucNetID,
    UInt16 timeoutMs
    );
```

Parameters:

hSocketTCP [in]

Socket handle

ulIPAddr [in]

unsigned long value of the IP address of the IR module. IP: 192.168.0.1 => 0xC0A80001

ucNetID [in]

Modbus NetID 0~247

timeoutMs [in]

connection timeout (ms)

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_SOCKET_CONN_TOUT	Connection timeout
IR_E_FUNC_ERR	Close function errors

Examples:

```
[VC++]
tHandle hSocketTCP;
unsigned long ulNICIpAddr = 0xC0A8000A;   //NIC IP: 192.168.0.10
hSocketTCP = IR_MTCP_TCPNew1(ulNICIpAddr);

int Ret = 0;
unsigned long  serverIP = 0xC0A80001;  // IP address of an IR module
uusigned char ucNetID = 1;

Ret = IR_MTCP_Connect(hSocketTCP, serverIP, ucNetID, 1000);
if (Ret == IR_E_OK)
{
    printf("Connect to IR-712-MTCP successfully!");
}
```

5.1.4 IR_MTCP_TCPClose

Description:

Disconnect and close a TCP socket.

Prototype:

[VC++]

```
int IR_MTCP_TCPClose (
    tHandle hSocketTCP
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern int IR_MTCP_ TCPClose (
    int hSocketTCP
    );
```

Parameters:

hSocketTCP [in]

Socket handle

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_FUNC_ERR	Close function errors

Examples:

```
[VC++]
int Ret = 0;
Ret = IR_MTCP_TCPClose(socketHandle);
if (Ret == IR_E_OK)
    printf("Disconnect from IR-712-MTCP successfully!");
```

5.1.5 IR_MTCP_GetModbusRecvTimeout

Description:

Get the receive timeout (ms) to wait for the Modbus TCP response from the IR module. Default value = 200 ms.

Prototype:

[VC++]

int IR_MTCP_GetModbusRecvTimeout (void);

[VC#.NET]

[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]

public static extern int IR_MTCP_GetModbusRecvTimeout ();

Parameters:

None.

Return Values:

Receive Modbus response timeout (ms). Default value = 200 ms.

Examples:

```
[VC++]
int tout;
tout = IR_MTCP_GetModbusRecvTimeout();
```

5.1.6 IR_MTCP_SetModbusRecvTimeout

Description:

Set the receive timeout (ms) of Modbus TCP responses from the IR module. Default value = 200 ms.

Prototype:

[VC++]

```
void IR_MTCP_SetModbusRecvTimeout(
    unsigned short recvTout
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern void IR_MTCP_SetModbusRecvTimeout(
    UInt16 recvTout
    );
```

Parameters:

recvTout [in]

Receive timeout for the Modbus TCP response after sending a Modbus TCP request. Default value is 200 ms.

Return Values:

None

Examples:

[VC++]

```
unsigned short recvTimeout = 100;  // 100 ms
IR_MTCP_SetModbusRecvTimeout(recvTimeout);
```

5.2 Basic Settings Function

5.2.1 IR_MTCP_GetBasicSetting

Description:

Get basic settings from the IR module.

Prototype:

[VC++]

```
int IR_GetBasicSetting(
    tHandle hSocketTCP,
    BasicSetParamsMTCP *pEBasicSettingMTCP
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MTCP_GetBasicSetting(
    int hSocketTCP,
    ref BasicSetParamsMTCP pEBasicSettingMTCP
    );
```

Parameters:

hSocketTCP [in]

TCP Socket handle.

pEBasicSettingMTCP [out]

Pointer to the BasicSetParamsMTCP structure variable of the basic settings of the IR learning remote module with Modbus TCP/UDP protocol.

Definition of *BasicSetParamsMTCP*:

[VC++]

```
typedef struct
{
  unsigned long  ulIPAddr;
  unsigned long  ulMaskAddr;
  unsigned long  ulGWAddr;
  unsigned long  ulIdleTOut;
```

```
  unsigned char  ucNetID;
  unsigned char  ucAddrType;
  unsigned short usGapTime;
} BasicSetParamsMTCP;
```

Where the structure members are defined as follows:

Table 5-1: Members of the BasicSetParamsMTCP structure

Members	Description
ulIPAddr	IP Addres. E.g. IP=192.168.0.1 = > 0xC0A80001
ulMaskAddr	Subnet Mask Addres. E.g. 255.255.0.0 = > 0xFFFF0000
ulGWAddr	Gateway address. E.g. 192.168.0.254 = > 0xC0A800FE
ulIdleTOut	Idle communication timeout (mimnute)
ucNetID	Modbus Net ID of IR modules (1 ~ 247)
ucAddrType	Address type. 0: static 1: DHCP
usGapTime	Gap time (ms), default= 72 ms, range = 72 ~ 200 ms

```
[VC#.NET]
[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack = 1)]
public struct BasicSetParamsMTCP
{
  public UInt32 ulIPAddr;
  public UInt32 ulMaskAddr;
  public UInt32 ulGWAddr;
  public UInt32 ulIdleTOut;
  public Byte ucNetID;
  public Byte ucAddrType;
  public UInt16 usGapTime;
};
```

Return Values:

Return Code	Description
IR_E_OK	No Error.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.

Examples:

[VC++]

```
int Ret = 0;
BasicSetParamsMTCP sBasicSet;

Ret = IR_MTCP_GetBasicSetting (
     hSocketTCP,
     &sBasicSet
     );

printf (“GapTime = %d\n”, sBasicSet.usGapTime);
```

5.2.2 IR_MTCP_SetBasicSetting

Description:

Set basic settings to the IR module.

Take care the times to call this function in programming because there are max 100000 times to write basic settings to the flash memory.

After calling this function, reboot the IR module to take effect the settings.

Prototype:

[VC++]

```
int IR_MTCP_SetBasicSetting (
    tHandle hSocketTCP,
    BasicSetParamsMTCP EBasicSettingMTCP,
    unsigned char ucReboot
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MTCP_SetBasicSetting(
    int hSocketTCP,
    [In]BasicSetParamsMTCP EBasicSettingMTCP,
    Byte ucReboot
    );
```

Parameters:

hSocketTCP [in]

TCP socket handle.

EBasicSetting [in]

BasicSetParamsMTCP structure variable for the bsaic settings of the IR learning remote module with Modbus TCP protocol.

Definition of *BasicSetParamsMTCP*:

[VC++]

Introduced in 5.2.1 IR_MTCP_GetBasicSetting ().

[VC#.NET]

Introduced in 5.2.1 IR_MTCP_GetBasicSetting ().

ucReboot [in]

Reset(reboot) the module after set the basic settings. 0=>nothing. 1=>write & reboot, 2=>only write, no reboot

Return Values:

Return Code	Description
IR_E_OK	No Error
IR_E_FUNC_ERR	Function Error (Parameter error in BasicSetParamsMTCP struct)
IR_E_SEND_MBCMD_FLASH_ERR	Error in sending Modbus command to write data to flash memory.

Examples:

```
[VC++]
int Ret = 0;
BasicSetParamsMTCP sBasicSet;
sBasicSet. ulIPAddr= 0xC0A8FF01;  // 192.168.255.1
sBasicSet. ulMaskAddr = 0xFFFF0000; // 255.255.0.0
sBasicSet. ulGWAddr = 0xC0A8FFFE; // 192.168.255.254
sBasicSet. ulIdleTOut = 0;        // 0 => disable
sBasicSet. ucNetID = 1;           // MB Net ID = 1
sBasicSet.ucAddrType = 0;        // 0 => static IP
sBasicSet. usGapTime; = 72;      // Gaptime = 72 ms

Ret = IR_MTCP_SetBasicSetting(
     socketHandle,
     sBasicSet,
     1                // Write & reboot module.
     );
```

5.2.3 IR_MTCP_RebootIRModule

Description:

Reboot IR Module with Modbus TCP protocol.

Prototype:

[VC++]

```
int IR_MTCP_RebootIRModule (
    tHandle hSocketTCP
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MTCP_RebootIRModule (
    int hSocketTCP
    );
```

Parameters:

hSocketTCP [in]

TCP socket handle.

Return Values:

Return Code	Description
IR_E_RESET_OK	Reset IR module successfully.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.

Examples:

```
[VC++]
int rtn = IR_MTCP_RebootIRModule(socketHandle);
```

5.3 IR Learning Function

5.3.1 IR_MTCP_GetIRCmdSummary

Description:

Get IR Command Summary from the IR module. The IR command summary is also saved to the buffer of the IR API Library.

Prototype:

[VC++]

```
int IR_MTCP_GetIRCmdSummary(
    tHandle hSocketTCP,
    unsigned short usIrModuleIndex,
    unsigned short *dataArr,
    unsigned short usInElementCnt
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MTCP_GetIRCmdSummary (
    int hSocketTCP,
    UInt16 usIrModuleIndex,
    [MarshalAs(UnmanagedType.LPArray, SizeParamIndex = 3)] UInt16[]
    dataArr,
    UInt16 usInElementCnt);
```

Parameters:

hSocketTCP [in]

TCP Socket handle.

usIrModuleIndex [in]

Index of IR learning remote module. 0x21: IR-712-MTCP

dataArr [out]

Data array for IR command summary. Array length = 13

where

dataArr[0]: IR device quantity (1 ~ 11)

dataArr[1]: Total quantity of IR commands (1 ~ 512, for IR-712-MTCP)
dataArr[2]: IR command quantity of device#1
dataArr[3]: IR command quantity of device#2
....
dataArr[N+1]: IR command quantity of device#N, where N<=11

usInElementCnt [in]

Length of the dataArr array. Should contain 13 elements exactly.

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_ILL_SUMMARY_LEN	Illegal length of the array (dataArr) for IR command summary.
IR_E_DEVICE_QTY_ERR	The device quantity exceeds the maximum.
IR_E_IRCMD_TOTAL_QTY_ERR	The device quantity setting exceeds the maximum.
IR_E_IRCMD_SUM_QTY_ERR	The element "Total quantity of IR commands" in the IR command summary exceeds the maximum.
IR_E_ILL_MODULE_INDEX	Illegal module index.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.

Examples:

```
[VC++]
int Ret = 0;
unsigned short usIrModuleIndex = 0x21, // Index for IR-712-MTCP
           dataArr[13] = {2, 10, 3, 7}, // 2 devices with 10 cmds
           usInElementCnt = 13;

Ret = IR_MTCP_GetIRCmdSummary(
       hSocketTCP,
       usIrModuleIndex,
       dataArr,
       usInElementCnt);
```

5.3.2 IR_MTCP_SetIRCmdSummary

Description:

Write the IR command summary from the library buffer to the flash memory of the IR learning module.

!Note: IR_SetIRCmdSummaryBuffer() should be called first.

Prototype:

[VC++]

```
int IR_MTCP_SetIRCmdSummary(
    tHandle hSocketTCP,
    unsigned short usIrModuleIndex
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MTCP_SetIRCmdSummary(
    int hSocketTCP,
    UInt16 usIrModuleIndex
    );
```

Parameters:

hSocketTCP [in]

TCP Socket handle.

usIrModuleIndex [in]

Index of IR learning remote module. 0x21: IR-712-MTCP

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_DEVICE_QTY_ERR	The device quantity exceeds the maximum.
IR_E_IRCMD_TOTAL_QTY_ERR	The device quantity setting exceeds the maximum.
IR_E_IRCMD_SUM_QTY_ERR	The element "Total quantity of IR commands" in the IR command summary exceeds the maximum.
IR_E_ILL_MODULE_INDEX	Illegal module index.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.
IR_E_SEND_MBCMD_FLASH_ERR	Error in sending/receiving Modbus command to write data to Flash.

Examples:

```
[VC++]
int Ret = 0;
unsigned short usIrModuleIndex = 0x21, // Index for IR-712-MTCP
               dataArr[13] = {2, 18, 10, 8},
               usInElementCnt = 13;    // length of dataArr

Ret = IR_SetIRCmdSummaryBuffer(usIrModuleIndex, dataArr,
usInElementCnt);
Ret = IR_MTCP_SetIRCmdSummary(hSocketTCP, usIrModuleIndex);
```

5.3.3 IR_MTCP_SetIRCmdSummary1

Description:

Set IR Command Summary to the flash memory of the IR module. The IR command summary is also saved to the buffer of the API library.

Prototype:

[VC++]

```
int IR_MTCP_SetIRCmdSummary1(
    tHandle hSocketTCP,
    unsigned short usIrModuleIndex,
    unsigned short *dataArr,
    unsigned short usInElementCnt
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MTCP_SetIRCmdSummary1(
    int hSocketTCP,
    UInt16 usIrModuleIndex,
    [MarshalAs(UnmanagedType.LPArray, SizeParamIndex = 3)] UInt16[] dataArr,
    UInt16 wOutElementLen
    );
```

Parameters:

hSocketTCP [in]

TCP Socket handle.

usIrModuleIndex [in]

Index of IR learning remote module. 0x21: IR-712A-MTCP

dataArr [in]

Data array for IR command summary. Array length = 13

where

dataArr[0]: IR device quantity (1 ~ 11)

dataArr[1]: Total quantity of IR commands (1 ~ 512, for IR-712-MTCP)

dataArr[2]: IR command quantity of device#1

dataArr[3]: IR command quantity of device#2

....

dataArr[N+1]: IR command quantity of device#N , where N<=11

usInElementCnt [in]

Length of the dataArr array. Should contain 13 elements exactly.

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_ILL_SUMMARY_LEN	Illegal length of the array(dataArr) for IR command summary.
IR_E_FUNC_ERR	Function error.
IR_E_DEVICE_QTY_ERR	The device quantity exceeds the maximum.
IR_E_IRCMD_TOTAL_QTY_ERR	The device quantity setting exceeds the maximum.
IR_E_IRCMD_SUM_QTY_ERR	The element "Total quantity of IR commands" in the IR command summary exceeds the maximum.
IR_E_ILL_MODULE_INDEX	Illegal module index.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.
IR_E_SEND_MBCMD_FLASH_ERR	Error in sending/receiving Modbus command to write data to Flash.

Examples:

```
[VC++]
int Ret = 0;
unsigned short usIrModuleIndex = 0x21, // Index for IR-712-MTCP
               dataArr[13] = {2, 18, 10, 8},
               usInElementCnt = 13;

Ret = IR_MTCP_SetIRCmdSummary1(
       hSocketTCP,
       usIrModuleIndex,
       dataArr,
       usInElementCnt);
```

5.3.4 IR_MTCP_LrnModeSet

Description:

Enable/Disable the IR learning mode for IR module with Modbus TCP protocol.

Prototype:

[VC++]

```
int IR_MTCP_LrnModeSet (
    tHandle hSocketTCP,
    unsigned char ucMode
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MTCP_LrnModeSet (
    int hSocketTCP,
    Byte ucMode
    );
```

Parameters:

hSocketTCP [in]

TCP Socket handle.

ucMode [in]

Learning mode of the IR module. 0 => learn off, 1=> learn on.

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.

Examples:

[VC++]

```
int Ret = 0;
unsigned char lrnMode = 1;  // Learn ON mode

Ret = IR_MTCP_LrnModeSet(hSocketTCP, lrnMode);
```

5.3.5 IR_MTCP_IsIRLearnOK

Description:

Check if learning an IR command is finished.

Prototype:

[VC++]

```
int IR_MTCP_IsIRLearnOK(
    tHandle hSocketTCP,
    unsigned short *pusIsOK,
    int sTimeout
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MTCP_IsIRLearnOK(
    int hSocketTCP,
    ref UInt16 pusIsOK,
    int sTimeout
    );
```

Parameters:

hSocketTCP [in]

TCP Socket handle.

pusIsOK [out]

0=>IR learning not OK, 1=>IR learning OK

sTimeout [in]

Receive timeout (ms).

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_FUNC_ERR	Function error.
IR_E_SOCKET_RECV_TOUT	receive timeout

Examples:

```
[VC++]
int Ret = 0;
int recvTout = 50;
unsigned short isLrnOK = 0;

Ret = IR_MTCP_IsIRLearnOK(hSocketTCP, &isLrnOK, recvTout);
```

5.3.6 IR_MTCP_GetIRCmdAfterLearn

Description:

Get one IR command learning data (without device name & IR cmd name) from the IR learning remote module to the buffer of the API library after learning an IR command right away.

(IR cmd data := IR device name + IR cmd name + IR cmd learning data)

Prototype:

[VC++]

```
int IR_MTCP_GetIRCmdAfterLearn (
    tHandle hSocketTCP,
    unsigned short usIrCmdNumber
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MTCP_GetIRCmdAfterLearn(
    int hSocketTCP,
    UInt16 usIrCmdNumber
    );
```

Parameters:

hSocketTCP [in]

TCP Socket handle.

usIrCmdNumber [in]

IR command number (1 ~ 512, for IR-712-MTCP).

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_ILL_IRCMD_NO	Illegal IR command number.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.

Examples:

```
[VC++]
int Ret = 0;
// Read IR cmd number 10.
Ret = IR_MTCP_GetIRCmdAfterLearn(hSocketTCP, 10);
```

5.4 Download/Load IR Command Functions

5.4.1 IR_MTCP_DLOneIRDataToModule

Description:

Download one IR cmd data (:= IR device name + IR cmd name + IR cmd learning data) from the API buffer to the IR Module.
(IR cmd data := IR device name + IR cmd name + IR cmd learning data)

Prototype:

```
[VC++]
int IR_MTCP_DLOneIRDataToModule(
    tHandle hSocketTCP,
    unsigned short usIrCmdNumber
    );

[VC#.NET]
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MTCP_DLOneIRDataToModule (
    int hSocketTCP,
    UInt16 usIrCmdNumber
    );
```

Parameters:

hSocketTCP [in]

TCP Socket handle.

usIrCmdNumber [in]

IR command number (1 ~ 512, for IR-712-MTCP).

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_ILL_IRCMD_NO	Illegal IR command number
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.
IR_E_SEND_MBCMD_FLASH_ERR	Error in sending/receiving Modbus command to write data to Flash.
IR_E_LRNCMD_LEN_EXCEED_MAX	The length of the IR cmd learning data exceeds the maximum.

Examples:

```
[VC++]
int Ret = 0;

// Download IR command data #3 to the IR module.
Ret = IR_MTCP_DLOneIRDataToModule (hSocketTCP, 3);
```

5.4.2 IR_MTCP_LDOneIRDataToAPI

Description:

Load one IR command data from the IR module and store it into the API buffer.
(IR cmd data := IR device name + IR cmd name + IR cmd learning data)

Prototype:

[VC++]

```
int IR_MTCP_LDOneIRDataToAPI(
    tHandle hSocketTCP,
    unsigned short usIrCmdNumber
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MTCP_LDOneIRDataToAPI(
    int hSocketTCP,
    UInt16 usIrCmdNumber
    );
```

Parameters:

hSocketTCP [in]

TCP Socket handle.

usIrCmdNumber [in]

IR command number (1 ~ 512, for IR-712-MTCP).

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_ILL_IRCMD_NO	Illegal IR command number
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.

Examples:

[VC++]

```
int Ret = 0;
```

```
// Load IR command data #3 from the IR module and store it
// in the API buffer.
Ret = IR_MTCP_LDOneIRDataToAPI(hSocketTCP, 3);
```

5.5 IR Emitting Function

5.5.1 IR_MTCP_EmitIrSignal

Description:

Request the IR module to emit the IR command stored in the flash memory. Emit IR signal by selected IR command number and IR output channels.

Prototype:

[VC++]

```
int IR_MTCP_EmitIrSignal(
    tHandle hSocketTCP,
    unsigned short usIrCmdNumber,
    unsigned char ucIrOutputCh
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MTCP_EmitIrSignal(
    int hSocketTCP,
    UInt16 usIrCmdNumber,
    Byte ucIrOutputCh
    );
```

Parameters:

hSocketTCP [in]

TCP Socket handle.

usIrCmdNumber [in]

IR command number (1 ~ 512, for IR-712-MTCP).

ucIrOutputCh [in]

IR output channels.

E.g., use IR output ch1 & ch2 => value = 0x03 (0000 0011(binary))

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_ILL_IRCMD_NO	Illegal IR command number
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.

Examples:

```
[VC++]
int Ret = 0;

// Request IR module to emit the IR command data #3 by the IR
// output channel 2.
Ret = IR_MTCP_EmitIrSignal(hSocketTCP, 3, 0x02);
```

5.5.2 IR_MTCP_RunCommand

Description:

Send an IR learning command data to the Modbus memory of the IR module and request IR module to emit that IR command.

Prototype:

[VC++]

```
int IR_MTCP_RunCommand(
    tHandle hSocketTCP,
    unsigned short usIrCmdNumber,
    unsigned char ucIrOutputCh
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MTCP_RunCommand(
    int hSocketTCP,
    UInt16 usIrCmdNumber,
    Byte ucIrOutputCh
    );
```

Parameters:

hSocketTCP [in]

TCP Socket handle.

usIrCmdNumber [in]

IR command number (1 ~ 512, for IR-712-MTCP).

ucIrOutputCh [in]

IR output channels.

E.g., use IR output ch1 & ch2 => value = 0x03 (0000 0011(binary))

Return Values:

Return Code	Description
IR_E_OK	No errors.

IR_E_IRCMD_DATA_EMPTY	NO IR data exists for the specified IR cmd number in the API buffer.
IR_E_ILL_IRCMD_NO	Illegal IR command number
IR_E_LRNCMD_LEN_EXCEED_MAX	The length of the IR cmd learning data exceeds the maximum.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.

Examples:

```
[VC++]
int Ret = 0;

Ret = IR_MTCP_RunCommand(hSocketTCP, 3, 0x02);
```

5.6 Others Function

5.6.1 IR_MTCP_ChkModuleName

Description:

Get model name of an IR remote module.

Prototype:

[VC++]

```
int IR_MTCP_ChkModuleName(
    tHandle hSocketTCP,
    unsigned short *pwModuleIndex
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MTCP_ChkModuleName(
    int hSocketTCP,
    ref UInt16 pwModuleIndex
    );
```

Parameters:

hSocketTCP [in]

TCP Socket handle.

pwModuleIndex [out]

Module index representing for specific IR module type(name).
E.g. 0x21=>"IR-712-MTCP".

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.
IR_E_MODNAME_INVALID	Get invalid module index (name).

Examples:

[VC++]

```
int Ret = 0;
unsigned short ModulIndex = 0;

Ret = IR_MTCP_ChkModuleName(hSocketTCP, &ModulIndex);

switch(ModuleIndex)
{
case 0x21:  // The IR module is IR-712-MTCP.
    break;
default:
    break;
}
```

6. Functions for Modbus UDP Communication

6.1 Communication Functions

6.1.1 IR_MUDP_UDPNew

Description:

Initialize a UDP socket for Modbus UDP communication.

Prototype:

[VC++]

```
tHandle IR_MUDP_UDPNew (
    unsigned long ulIPAddr,
    unsigned char ucNetID
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MUDP_UDPNew(
    UInt32 ulIPAddr,
    Byte ucNetID
    );
```

Parameters:

ulIPAddr [in]

IP addresss of the IR module (UDP server)

IP = 192.168.0.1 => 0xC0A80001

ucNetID [in]

Modbus Net ID of the IR module: 1 ~247.

Return Values:

tHandle: UDP socket handle;

If successful, a UDP socket handle is returned.

If not successful, IR_E_SOCKET_INVALID is returned.

Examples:

[VC++]

```
tHandle hSocketUDP;
Uint32 serverIP = 0xC0A8FF01;  // IP: 192.168.255.1 => 0xC0A8FF01
unsigned char ucNetID = 1;
hSocketUDP = IR_MUDP_UDPNew(serverIP, ucNetID);
```

6.1.2 IR_MUDP_UDPNew1

Description:

Initailize a UDP socket which binds to a specific network interface (e.g. Ethernet card, Wi-Fi).

Prototype:

[VC++]

```
tHandle IR_MUDP_UDPNew1 (
    unsigned long ulNicIPAddr,
    unsigned long ulIPAddr,
    unsigned char ucNetID
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MUDP_UDPNew1 (
    UInt32 ulNicIPAddr,
    UInt32 ulIPAddr,
    Byte ucNetID
    );
```

Parameters:

ulNicIPAddr [in]

IP address of a network card for binding.
IP: 0xC0A80001 (=>192.168.0.1)

ulIPAddr [in]

IP addresss of the IR Module (UDP server).
IP: 0xC0A80002 (=>192.168.0.2)

ucNetID [in]

Modbus Net ID of the IR module: 1 ~247.

Return Values:

tHandle: UDP socket handle;

If successful, a UDP socket handle is returned.

If not successful, IR_E_SOCKET_INVALID is returned.

Examples:

```
[VC++]
tHandle hSocketUDP;
unsigned long ulNICIpAddr = 0xC0A8000A;   // IP: 192.168.0.10
unsigned long ulIPAddr = 0xC0A80014;      // IP: 192.168.0.20
unsigned char ucNetID = 1;
hSocketUDP = IR_MTCP_TCPNew1(ulNICIpAddr, ulIPAddr, ucNetID);
```

6.1.3 IR_MUDP_UDPClose

Description:

Close and release a UDP socket.

Prototype:

[VC++]

```
int IR_MTCP_UDPClose (
    tHandle hSocketUDP
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern int IR_MUDP_ UDPClose (
    int hSocketUDP
    );
```

Parameters:

hSocketUDP [in]

Modbus UDP socket handle.

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_FUNC_ERR	Close function errors

Examples:

```
[VC++]
int Ret = 0;
Ret = IR_MUDP_UDPClose(hSocketUDP);
if (Ret == IR_E_OK)
    printf("Close Modbus UDP socket successfully!");
```

6.1.4 IR_MUDP_GetModbusRecvTimeout

Description:

Get the receive timeout (ms) to wait for the Modbus UDP response from the IR module.

Prototype:

[VC++]

```
int IR_MUDP_GetModbusRecvTimeout (void);
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern int IR_MUDP_GetModbusRecvTimeout ();
```

Parameters:

None.

Return Values:

Timeout (ms) for receiving Modbus response. Default value = 200 ms.

Examples:

```
[VC++]
int tout;
tout = IR_MUdP_GetModbusRecvTimeout();
```

6.1.5 IR_MUDP_SetModbusRecvTimeout

Description:

Set the receive timeout (ms) of Modbus UDP responses from the IR module.

Prototype:

[VC++]

```
void IR_MUDP_SetModbusRecvTimeout(
    unsigned short recvTout
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern void IR_MUDP_SetModbusRecvTimeout(
    UInt16 recvTout
    );
```

Parameters:

recvTout [in]

Timeout for receiving the Modbus UDP response after sending a Modbus UDP request to the IR module by the API function.

Default value is 200 ms.

Return Values:

None

Examples:

[VC++]

```
unsigned short recvTimeout = 100;  // 100 ms
IR_MUDP_SetModbusRecvTimeout(recvTimeout);
```

6.2 Basic Settings Functions

6.2.1 IR_MUDP_GetBasicSetting

Description:

Get basic settings from the IR module.

Prototype:

[VC++]

```
int IR_MUDP_GetBasicSetting(
    tHandle hSocketUDP,
    BasicSetParamsMUDP *pEBasicSettingMTCsP
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MUDP_GetBasicSetting(
    int hSocketUDP,
    ref BasicSetParamsMTCP pEBasicSettingMUDP
    );
```

Parameters:

hSocketUDP [in]

UDP Socket handle.

pEBasicSettingMTCP [out]

Pointer to the BasicSetParamsMTCP structure variable of the basic settings of the IR learning remote module with Modbus TCP/UDP protocol.

Definition of *BasicSetParamsMTCP*:

[VC++]

Refer to section 5.2.1 IR_MTCP_GetBasicSetting ().

[VC#.NET]

Refer to section 5.2.1 IR_MTCP_GetBasicSetting ().

Return Values:

Return Code	Description
IR_E_OK	No Error.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.

Examples:

```
[VC++]
int Ret = 0;
BasicSetParamsMTCP sBasicSet;

Ret = IR_MUDP_GetBasicSetting (
     hSocketUDP,
     &sBasicSet
     );

printf (“GapTime = %d\n”, sBasicSet.usGapTime);
```

6.2.2 IR_MUDP_SetBasicSetting

Description:

Set basic settings to the IR module.

Take care the times to call this function in programming because there are max 100000 times to write basic settings to the flash memory.

After calling this function, reboot the IR module to take effect the settings.

Prototype:

[VC++]

```
int IR_MUDP_SetBasicSetting (
    tHandle hSocketUDP,
    BasicSetParamsMTCP EBasicSettingMTCP,
    unsigned char ucReboot
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MUDP_SetBasicSetting(
    int hSocketUDP,
    [In]BasicSetParamsMTCP EBasicSettingMTCP,
    Byte ucReboot
    );
```

Parameters:

hSocketUDP [in]

Modbus UDP socket handle.

EBasicSetting [in]

BasicSetParamsMTCP structure variable for the bsaic settings of the IR learning remote module with Modbus TCP/UDP protocol.

Definition of *BasicSetParamsMTCP*:

[VC++]

Refer to section 5.2.1 IR_MTCP_GetBasicSetting ().

[VC#.NET]

Refer to section 5.2.1 IR_MTCP_GetBasicSetting ().

ucReboot [in]

Reset(reboot) the module after set the basic settings. 0=>nothing. 1=>write & reboot, 2=>only write, no reboot

Return Values:

Return Code	Description
IR_E_OK	No Error
IR_E_FUNC_ERR	Function Error (Parameter error in BasicSetParamsMTCP struct)
IR_E_SEND_MBCMD_FLASH_ERR	Error in sending Modbus command to write data to flash memory.

Examples:

```
[VC++]
int Ret = 0;
BasicSetParamsMTCP sBasicSet;
sBasicSet. ulIPAddr= 0xC0A8FF01;  // 192.168.255.1
sBasicSet. ulMaskAddr = 0xFFFF0000; // 255.255.0.0
sBasicSet. ulGWAddr = 0xC0A8FFFE; // 192.168.255.254
sBasicSet. ulIdleTOut = 0;        // 0 => disable
sBasicSet. ucNetID = 1;           // MB Net ID = 1
sBasicSet.ucAddrType = 0;        // 0 => static IP
sBasicSet. usGapTime; = 72;      // Gaptime = 72 ms

Ret = IR_MUDP_SetBasicSetting(
     hSocketUDP,
     sBasicSet,
     1                // Write & reboot module.
     );
```

6.2.3 IR_MUDP_RebootIRModule

Description:

Reboot IR Module with Modbus UDP protocol.

Prototype:

[VC++]

```
int IR_MUDP_RebootIRModule (
    tHandle hSocketUDP
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MUDP_RebootIRModule (
    int hSocketUDP
    );
```

Parameters:

hSocketUDP [in]

Modbus UDP socket handle.

Return Values:

Return Code	Description
IR_E_RESET_OK	Reset IR module successfully.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.

Examples:

```
[VC++]
int rtn = IR_MUDP_RebootIRModule(hSocketUDP);
```

6.3 IR Learning Functions

6.3.1 IR_MUDP_GetIRCmdSummary

Description:

Get IR Command Summary from the IR module. The IR command summary is also saved to the buffer of the IR API Library.

Prototype:

[VC++]

```
int IR_MUDP_GetIRCmdSummary(
      tHandle hSocketUDP,
      unsigned short usIrModuleIndex,
      unsigned short *dataArr,
      unsigned short usInElementCnt
      );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MUDP_GetIRCmdSummary (
      int hSocketUDP,
      UInt16 usIrModuleIndex,
      [MarshalAs(UnmanagedType.LPArray, SizeParamIndex = 3)] UInt16[]
      dataArr,
      UInt16 usInElementCnt);
```

Parameters:

hSocketUDP [in]

Modbus UDP Socket handle.

usIrModuleIndex [in]

Index of IR learning remote module. 0x21: IR-712-MTCP

dataArr [out]

Data array for IR command summary. Array length = 13

where

dataArr[0]: IR device quantity (1 ~ 11)

dataArr[1]: Total quantity of IR commands (1 ~ 512, for IR-712-MTCP)
dataArr[2]: IR command quantity of device#1
dataArr[3]: IR command quantity of device#2
....
dataArr[N+1]: IR command quantity of device#N, where N<=11

usInElementCnt [in]

Length of the dataArr array. Should contain 13 elements exactly.

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_ILL_SUMMARY_LEN	Illegal length of the array (dataArr) for IR command summary.
IR_E_DEVICE_QTY_ERR	The device quantity exceeds the maximum.
IR_E_IRCMD_TOTAL_QTY_ERR	The device quantity setting exceeds the maximum.
IR_E_IRCMD_SUM_QTY_ERR	The element "Total quantity of IR commands" in the IR command summary exceeds the maximum.
IR_E_ILL_MODULE_INDEX	Illegal module index.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.

Examples:

```
[VC++]
int Ret = 0;
unsigned short usIrModuleIndex = 0x21, // Index for IR-712-MTCP
           dataArr[13] = {2, 10, 3, 7}, // 2 devices with 10 cmds
           usInElementCnt = 13;

Ret = IR_MUDP_GetIRCmdSummary(
        hSocketUDP,
        usIrModuleIndex,
        dataArr,
        usInElementCnt);
```

6.3.2 IR_MUDP_SetIRCmdSummary

Description:

Write the IR command summary from the library buffer to the flash memory of the IR learning module.

!Note: IR_SetIRCmdSummaryBuffer() should be called first.

Prototype:

[VC++]

```
int IR_MUDP_SetIRCmdSummary(
    tHandle hSocketUDP,
    unsigned short usIrModuleIndex
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MUDP_SetIRCmdSummary(
    int hSocketUDP,
    UInt16 usIrModuleIndex
    );
```

Parameters:

hSocketUDP [in]

Modbus TCP Socket handle.

usIrModuleIndex [in]

Index of IR learning remote module. 0x21: IR-712-MTCP

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_DEVICE_QTY_ERR	The device quantity exceeds the maximum.
IR_E_IRCMD_TOTAL_QTY_ERR	The device quantity setting exceeds the maximum.
IR_E_IRCMD_SUM_QTY_ERR	The element "Total quantity of IR commands" in the IR command summary exceeds the maximum.
IR_E_ILL_MODULE_INDEX	Illegal module index.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.
IR_E_SEND_MBCMD_FLASH_ERR	Error in sending/receiving Modbus command to write data to Flash.

Examples:

```
[VC++]
int Ret = 0;
unsigned short usIrModuleIndex = 0x21, // Index for IR-712-MTCP
               dataArr[13] = {2, 18, 10, 8},
               usInElementCnt = 13;    // length of dataArr

Ret = IR_SetIRCmdSummaryBuffer(usIrModuleIndex, dataArr,
usInElementCnt);
Ret = IR_UDP_SetIRCmdSummary(hSocketUDP, usIrModuleIndex);
```

6.3.3 IR_MUDP_SetIRCmdSummary1

Description:

Set IR Command Summary to the flash memory of the IR module. The IR command summary is also saved to the buffer of the API library.

Prototype:

[VC++]

```
int IR_MUDP_SetIRCmdSummary1(
    tHandle hSocketUDP,
    unsigned short usIrModuleIndex,
    unsigned short *dataArr,
    unsigned short usInElementCnt
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MUDP_SetIRCmdSummary1(
    int hSocketUDP,
    UInt16 usIrModuleIndex,
    [MarshalAs(UnmanagedType.LPArray, SizeParamIndex = 3)] UInt16[] dataArr,
    UInt16 wOutElementLen
    );
```

Parameters:

hSocketUDP [in]

UDP Socket handle.

usIrModuleIndex [in]

Index of IR learning remote module. 0x21: IR-712A-MTCP

dataArr [in]

Data array for IR command summary. Array length = 13

where

dataArr[0]: IR device quantity (1 ~ 11)

dataArr[1]: Total quantity of IR commands (1 ~ 512, for IR-712-MTCP)

dataArr[2]: IR command quantity of device#1

dataArr[3]: IR command quantity of device#2

....

dataArr[N+1]: IR command quantity of device#N , where N<=11

usInElementCnt [in]

Length of the dataArr array. Should contain 13 elements exactly.

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_ILL_SUMMARY_LEN	Illegal length of the array(dataArr) for IR command summary.
IR_E_FUNC_ERR	Function error.
IR_E_DEVICE_QTY_ERR	The device quantity exceeds the maximum.
IR_E_IRCMD_TOTAL_QTY_ERR	The device quantity setting exceeds the maximum.
IR_E_IRCMD_SUM_QTY_ERR	The element "Total quantity of IR commands" in the IR command summary exceeds the maximum.
IR_E_ILL_MODULE_INDEX	Illegal module index.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.
IR_E_SEND_MBCMD_FLASH_ERR	Error in sending/receiving Modbus command to write data to Flash.

Examples:

```
[VC++]
int Ret = 0;
unsigned short usIrModuleIndex = 0x21, // Index for IR-712-MTCP
               dataArr[13] = {2, 18, 10, 8},
               usInElementCnt = 13;

Ret = IR_MUDP_SetIRCmdSummary1(
       hSocketUDP,
       usIrModuleIndex,
       dataArr,
       usInElementCnt);
```

6.3.4 IR_MUDP_LrnModeSet

Description:

Enable/Disable the IR learning mode for IR module with Modbus UDP protocol.

Prototype:

[VC++]

```
int IR_MUDP_LrnModeSet (
    tHandle hSocketUDP,
    unsigned char ucMode
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MUDP_LrnModeSet (
    int hSocketUDP,
    Byte ucMode
    );
```

Parameters:

hSocketUDP [in]

UDP Socket handle.

ucMode [in]

Learning mode of the IR module. 0 => learn off, 1=> learn on.

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.

Examples:

[VC++]

```
int Ret = 0;
unsigned char lrnMode = 1;  // Learn ON mode

Ret = IR_MUDP_LrnModeSet(hSocketUDP, lrnMode);
```

6.3.5 IR_MUDP_IsIRLearnOK

Description:

Check if learning an IR command is finished.

Prototype:

[VC++]

```
int IR_MUDP_IsIRLearnOK(
    tHandle hSocketUDP,
    unsigned short *pusIsOK,
    int sTimeout
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MUDP_IsIRLearnOK(
    int hSocketUDP,
    ref UInt16 pusIsOK,
    int sTimeout
    );
```

Parameters:

hSocketUDP [in]

UDP Socket handle.

pusIsOK [out]

0=>IR learning not OK, 1=>IR learning OK

sTimeout [in]

Receive timeout (ms).

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_FUNC_ERR	Function error.
IR_E_SOCKET_RECV_TOUT	receive timeout

Examples:

```
[VC++]
int Ret = 0;
int recvTout = 50;
unsigned short isLrnOK = 0;

Ret = IR_MUDP_IsIRLearnOK(hSocketUDP, &isLrnOK, recvTout);
```

6.3.6 IR_MUDP_GetIRCmdAfterLearn

Description:

Get one IR command learning data (without device name & IR cmd name) from the IR learning remote module to the buffer of the API library after learning an IR command right away.
(IR cmd data := IR device name + IR cmd name + IR cmd learning data)

Prototype:

[VC++]

```
int IR_MUDP_GetIRCmdAfterLearn (
    tHandle hSocketUDP,
    unsigned short usIrCmdNumber
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MUDP_GetIRCmdAfterLearn(
    int hSocketUDP,
    UInt16 usIrCmdNumber
    );
```

Parameters:

hSocketUDP [in]

UDP Socket handle.

usIrCmdNumber [in]

IR command number (1 ~ 512, for IR-712-MTCP).

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_ILL_IRCMD_NO	Illegal IR command number.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.

Examples:

```
[VC++]
int Ret = 0;
// Read IR cmd number 10.
Ret = IR_MUDP_GetIRCmdAfterLearn(hSocketUDP, 10);
```

6.4 Download/Load IR Command Functions

6.4.1 IR_MUDP_DLOneIRDataToModule

Description:

Download one IR cmd data (:= IR device name + IR cmd name + IR cmd learning data) from the API buffer to the IR Module.
(IR cmd data := IR device name + IR cmd name + IR cmd learning data)

Prototype:

```
[VC++]
int IR_MUDP_DLOneIRDataToModule(
    tHandle hSocketUDP,
    unsigned short usIrCmdNumber
    );

[VC#.NET]
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MUDP_DLOneIRDataToModule (
    int hSocketUDP,
    UInt16 usIrCmdNumber
    );
```

Parameters:

hSocketUDP [in]

UDP Socket handle.

usIrCmdNumber [in]

IR command number (1 ~ 512, for IR-712-MTCP).

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_ILL_IRCMD_NO	Illegal IR command number
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.
IR_E_SEND_MBCMD_FLASH_ERR	Error in sending/receiving Modbus command to write data to Flash.
IR_E_LRNCMD_LEN_EXCEED_MAX	The length of the IR cmd learning data exceeds the maximum.

Examples:

```
[VC++]
int Ret = 0;

// Download IR command data #3 to the IR module.
Ret = IR_MUDP_DLOneIRDataToModule (hSocketUDP, 3);
```

6.4.2 IR_MUDP_LDOneIRDataToAPI

Description:

Load one IR command data from the IR module and store it into the API buffer.
(IR cmd data := IR device name + IR cmd name + IR cmd learning data)

Prototype:

[VC++]

```
int IR_MUDP_LDOneIRDataToAPI(
    tHandle hSocketUDP,
    unsigned short usIrCmdNumber
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MUDP_LDOneIRDataToAPI(
    int hSocketUDP,
    UInt16 usIrCmdNumber
    );
```

Parameters:

hSocketUDP [in]

UDP Socket handle.

usIrCmdNumber [in]

IR command number (1 ~ 512, for IR-712-MTCP).

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_ILL_IRCMD_NO	Illegal IR command number
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.

Examples:

[VC++]

```
int Ret = 0;
```

```
// Load IR command data #3 from the IR module and store it
// in the API buffer.
Ret = IR_MUDP_LDOneIRDataToAPI(hSocketUDP, 3);
```

6.5 IR Emitting Functions

6.5.1 IR_MUDP_EmitIrSignal

Description:

Request the IR module to emit the IR command stored in the flash memory. Emit IR signal by selected IR command number and IR output channels.

Prototype:

[VC++]

```
int IR_MUDP_EmitIrSignal(
    tHandle hSocketUDP,
    unsigned short usIrCmdNumber,
    unsigned char ucIrOutputCh
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MUDP_EmitIrSignal(
    int hSocketUDP,
    UInt16 usIrCmdNumber,
    Byte ucIrOutputCh
    );
```

Parameters:

hSocketUDP [in]

UDP Socket handle.

usIrCmdNumber [in]

IR command number (1 ~ 512, for IR-712-MTCP).

ucIrOutputCh [in]

IR output channels.

E.g., use IR output ch1 & ch2 => value = 0x03 (0000 0011(binary))

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_ILL_IRCMD_NO	Illegal IR command number
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.

Examples:

```
[VC++]
int Ret = 0;

// Request IR module to emit the IR command data #3 by the IR
// output channel 2.
Ret = IR_MUDP_EmitIrSignal(hSocketTCP, 3, 0x02);
```

6.5.2 IR_MUDP_RunCommand

Description:

Send an IR learning command data to the Modbus memory of the IR module and request IR module to emit that IR command.

Prototype:

[VC++]

```
int IR_MUDP_RunCommand(
    tHandle hSocketUDP,
    unsigned short usIrCmdNumber,
    unsigned char ucIrOutputCh
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MUDP_RunCommand(
    int hSocketUDP,
    UInt16 usIrCmdNumber,
    Byte ucIrOutputCh
    );
```

Parameters:

hSocketUDP [in]

UDP Socket handle.

usIrCmdNumber [in]

IR command number (1 ~ 512, for IR-712-MTCP).

ucIrOutputCh [in]

IR output channels.

E.g., use IR output ch1 & ch2 => value = 0x03 (0000 0011(binary))

Return Values:

Return Code	Description
IR_E_OK	No errors.

IR_E_IRCMD_DATA_EMPTY	NO IR data exists for the specified IR cmd number in the API buffer.
IR_E_ILL_IRCMD_NO	Illegal IR command number
IR_E_LRNCMD_LEN_EXCEED_MAX	The length of the IR cmd learning data exceeds the maximum.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.

Examples:

```
[VC++]
int Ret = 0;

Ret = IR_MUDP_RunCommand(hSocketUDP, 3, 0x02);
```

6.6 Others Functions

6.6.1 IR_MUDP_ChkModuleName

Description:

Get model name of an IR remote module.

Prototype:

[VC++]

```
int IR_MUDP_ChkModuleName(
    tHandle hSocketUDP,
    unsigned short *pwModuleIndex
    );
```

[VC#.NET]

```
[DllImport("irlrnapi.dll", CallingConvention = CallingConvention.StdCall)]
public static extern Int32 IR_MUDP_ChkModuleName(
    int hSocketUDP,
    ref UInt16 pwModuleIndex
    );
```

Parameters:

hSocketUDP [in]

UDP Socket handle.

pwModuleIndex [out]

Module index representing for specific IR module type(name).
E.g. 0x21=>"IR-712-MTCP".

Return Values:

Return Code	Description
IR_E_OK	No errors.
IR_E_SEND_MBCMD_ERR	Error in sending/receiving Modbus command.
IR_E_MODNAME_INVALID	Get invalid module index (name).

Examples:

[VC++]

```
int Ret = 0;
unsigned short ModulIndex = 0;

Ret = IR_MUDP_ChkModuleName(hSocketUDP, &ModulIndex);

switch(ModuleIndex)
{
case 0x21:  // The IR module is IR-712-MTCP.
    break;
default:
    break;
}
```

7. Return Code

Table 7-1: Return Codes and Values

Return Code	Value	Description
IR_E_OK	0x00	No errors.
IR_E_COM_STATUS_AVAIL	0x01	COM port is available and not open.
IR_E_IRCMD_DATA_EMPTY	0x00	No IR learning data for the specified IR command number in the buffer.
IR_E_IRCMD_DATA_EXIST	0x01	There are IR learning data for the specified IR command number in the buffer.
IR_E_RESET_OK	0x02	Reset IR module successfully.
IR_E_FUNC_ERR	0x11	Function errors
IR_E_COM_STATUS_ERR	0x12	COM port is not available or is open.
IR_E_OPEN_COM_ERR	0x13	Open COM error where COM is open or is not available.
IR_E_ILL_BAUD_IDX	0x14	Illegal baud rate
IR_E_ILL_DATA_BIT	0x15	Illegal data bits
IR_E_ILL_STOP_BIT	0x16	Illegal stop bits
IR_E_ILL_PARITY	0x17	Illegal parity
IR_E_MODNAME_INVALID	0x18	Get invalid module index (name).
IR_E_SEND_MBCMD_ERR	0x19	Error in sending/receiving Modbus command.
IR_E_SEND_MBCMD_FLASH_ERR	0x1A	Error in sending Modbus command to write data to flash memory.
IR_E_ILL_SUMMARY_LEN	0x1B	Illegal length of the array (dataArr) for IR command summary.
IR_E_DEVICE_QTY_ERR	0x1C	The device quantity exceeds the maximum.
IR_E_IRCMD_TOTAL_QTY_ERR	0x1D	The device quantity setting exceeds the maximum.
IR_E_IRCMD_SUM_QTY_ERR	0x1E	The element "Total quantity of IR commands" in the IR command summary exceeds the maximum.
IR_E_ILL_MODULE_INDEX	0x1F	Illegal module index.
IR_E_ILL_IRCMD_NO	0x20	Illegal IR command number.
IR_E_DEVICE_NAME_ARR_UNDERSIZE	0x21	The size of the device name array variable is too small as an argument of a function.
IR_E_IRCMD_NAME_ARR_UNDERSIZE	0x22	The size of the IR cmd name array variable is too small as an argument of a function.
IR_E_DEVICE_NAME_ARR_OVERSIZE	0x23	The length of the device name is too large as an argument to the function.
IR_E_IRCMD_NAME_ARR_OVERSIZE	0x24	The length of the IR command name is too large as an argument to the function.
IR_E_LRNCMD_LEN_EXCEED_MAX	0x25	The length of the IR cmd learning data exceeds the maximum.
IR_E_RESET_COM_ERR	0x26	COM error in sending reset message to the IR module. Receive timeout or COM error.
IR_E_RESET_MODULE_FAIL	0x27	Failed to reset the IR module.

Table 7-1: Return Codes and Values (cont.)

Return Code	Value	Description
IR_E_IRCMD_TOTALQTY_NOMATCH	0x28	The totl quantity of IR cmds do not match the sum of IR cmds in all devices in the array of IR cmd summary
IR_E_OPEN_FILE_ERROR	0x29	Open file error.
IR_E_CLOSE_FILE_ERROR	0x2A	Close file error.
IR_E_IR_FILE_FORMAT_ERR	0x2B	Format error in IR learning data file
IR_E_SOCKET_BIND_ERR	0x2C	Error in binding socket with the network address of NIC.
IR_E_SOCKET_CONN_TOUT	0x2D	Connection timeout by Modbus TCP.
IR_E_SOCKET_RECV_TOUT	0x2E	Socket receive timeout.
IR_E_SOCKET_INVALID	(-1)	SOCKET Error

Appendix A Build a VC++ MFC Project for IR API Library

The following steps indicates how to establish an MFC desktop application invoking the IR API library. The demonstration uses Visual Studio C++ 2013.

1. New a MFC Project

a. After the Visual Studio 2013 IDE is launched, click the Menu [File] -> [New]->[Project…] to open the “New Project” Dialog.

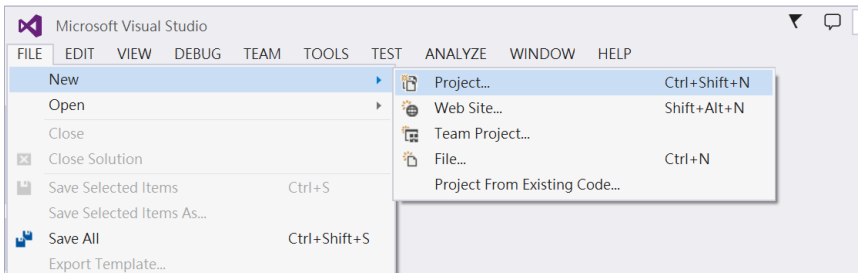


Figure A-1

b. In the left pane of the New Project Dialog, select the [installed]->[Temlates]->[Visual C++]->[MFC]. In the middle pane, select [MFC Application]. Fill in the project “Name”, “Location” and “Solution Name” at the buttom of the dialog. Click “OK” button. The Project Name here is “IrAPI_MFC_Demo1”.

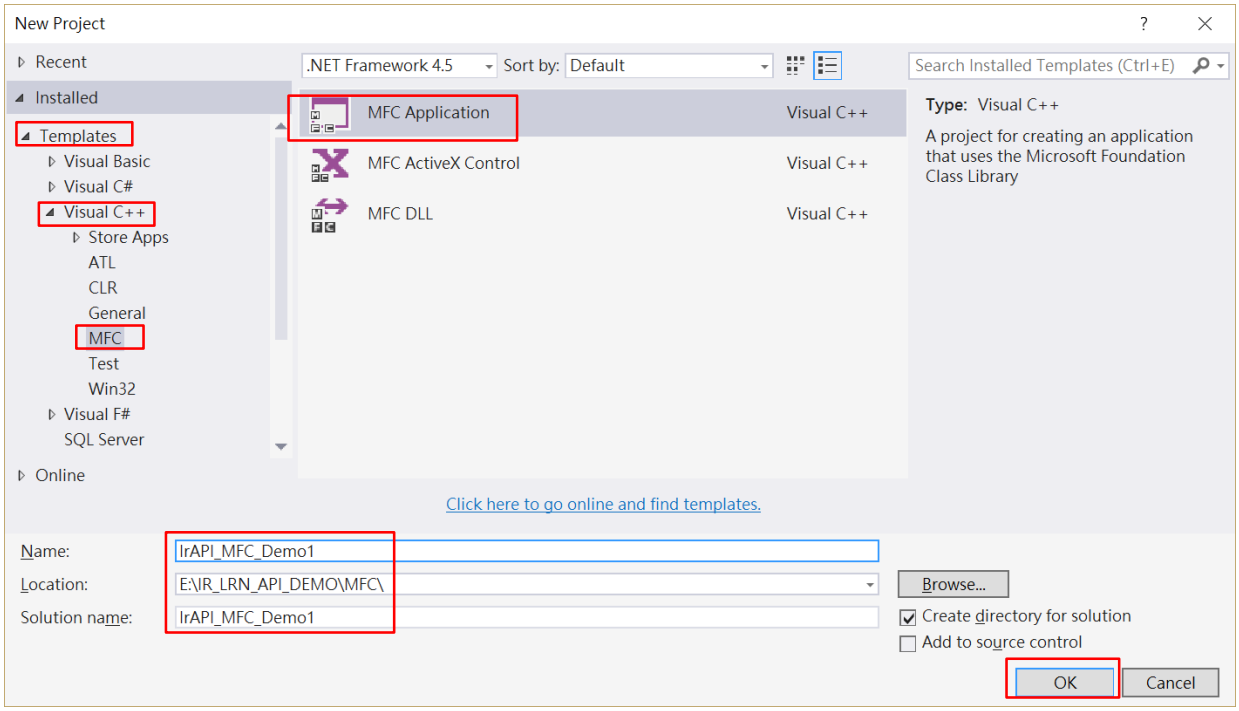


Figure A-2

c. Cick “Next” button in the Figure A-3.

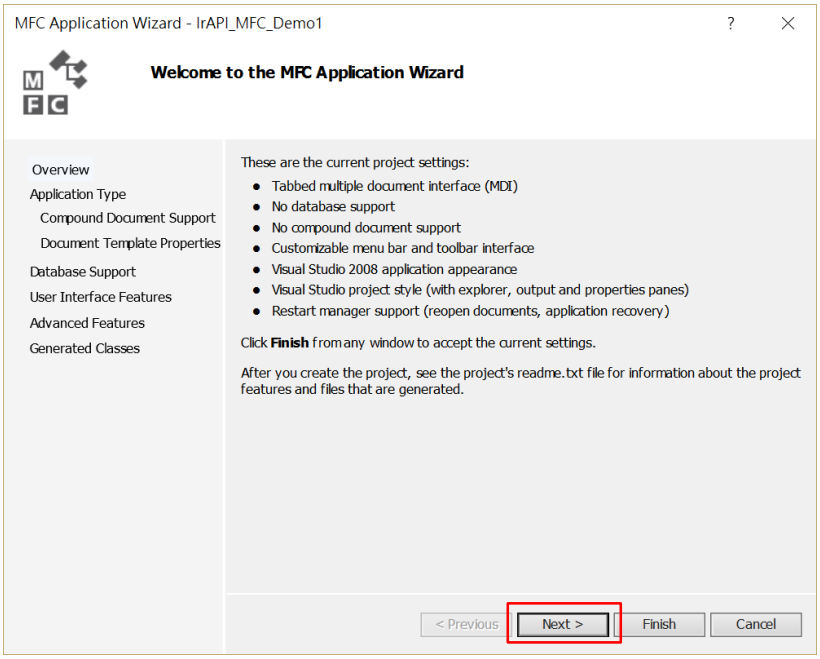


Figure A-3

d. Select “Dialog based” radio button in the Application type. Than click “Fnisth” button.

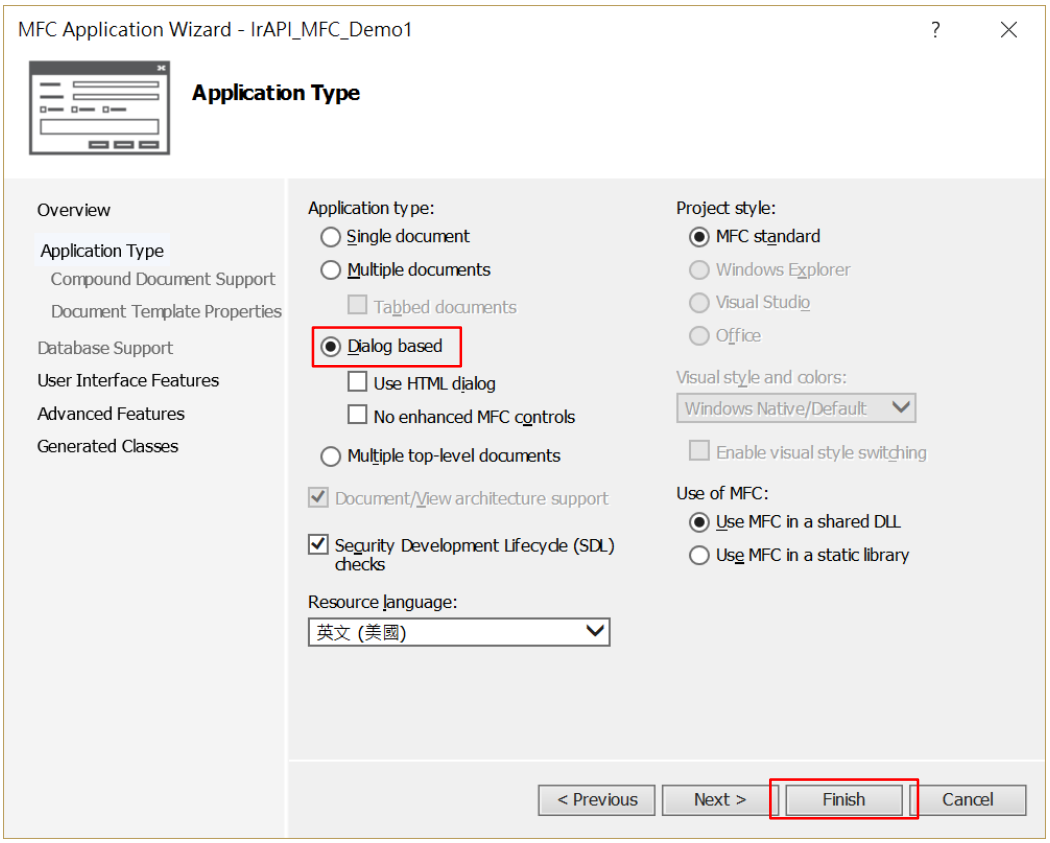


Figure A-4

2. Configure the Project Properties

a. Click the Project Item (IrAPI_MFC_Demo1) in the Solution Explorer pane. Click Menu [PROJECT]=>[Properties] to open the Properties Pages.

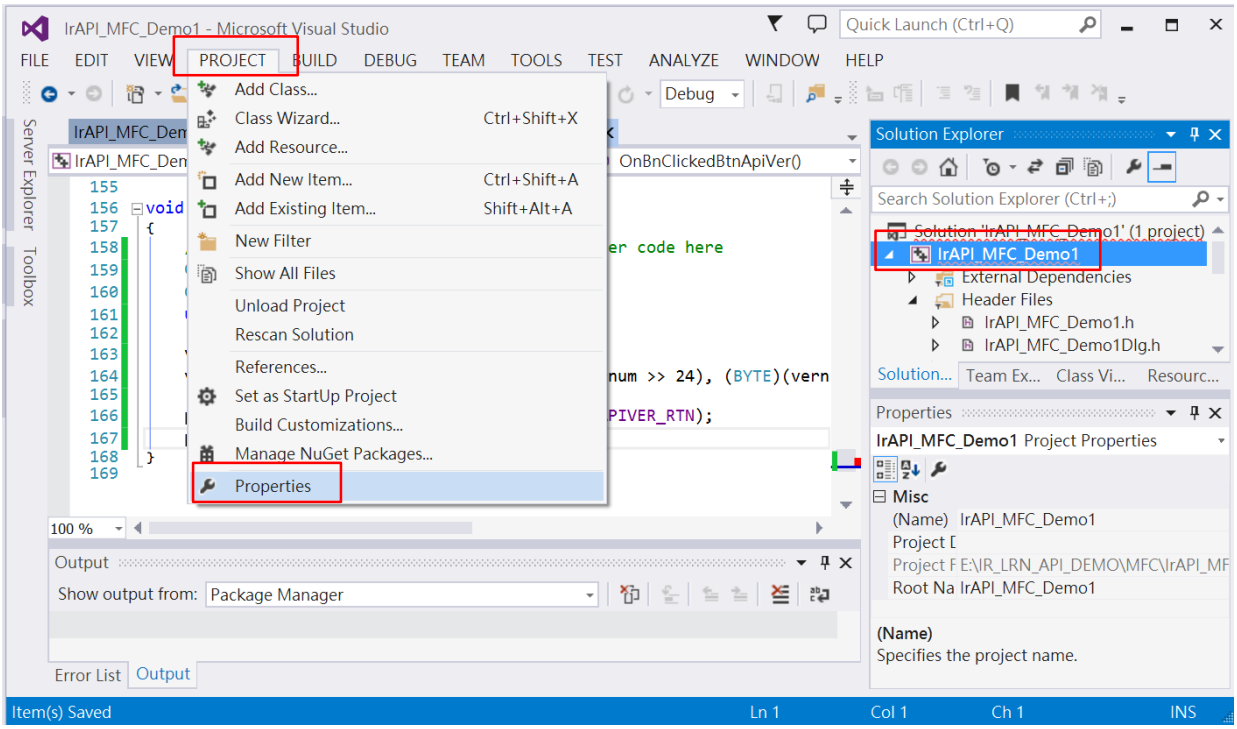


Figure A-5

b. In the left pane of Properties Pages, select [Configuration Properties] => [VC++ Directories]. Then in the right pane, fill in the directory path where the “IrLrnApi.h” locates in the [Include Directories] item. Fill in the directory path where the “irlrnapi.lib” locates in the [Library Directories] item.

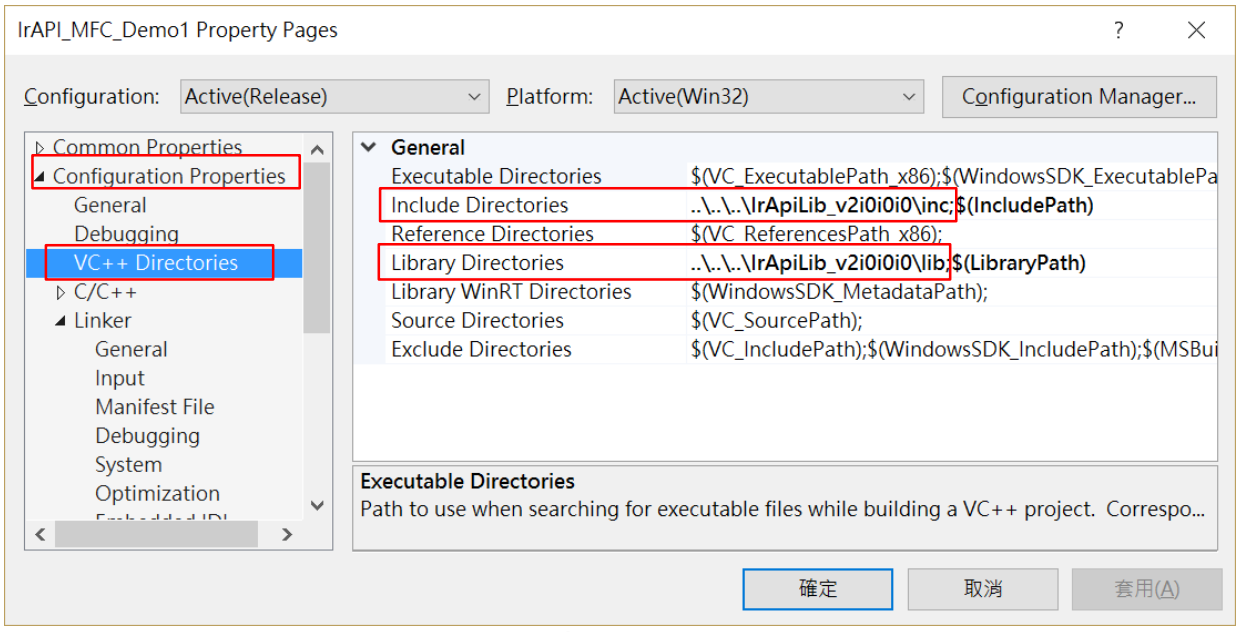


Figure A-6

c. In the left pane of Properties Pages, select [Configuration Properties] => [Linker] => [Input]. In the right pane, fill in the “irlrnapi.lib” in the “Additional Dependencies” item. The click the “OK” button.

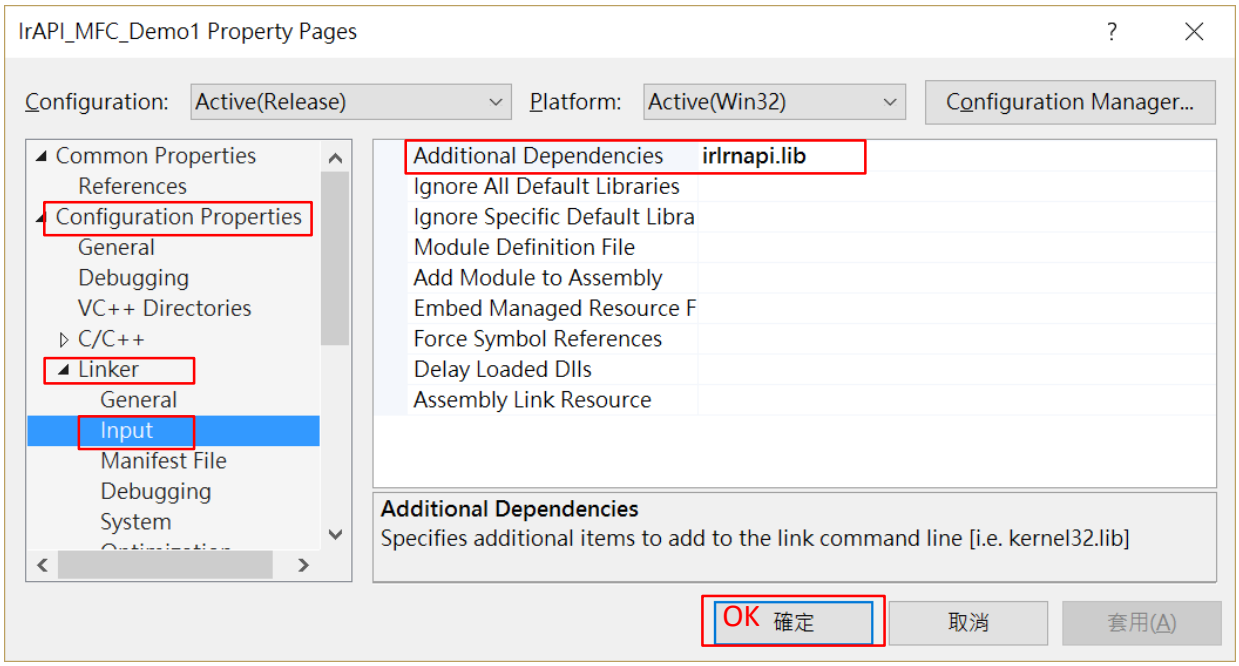


Figure A-7

3. Buil the Application

a. Open the “IrAPI_MFC_Demo1Dlg.cpp” in the Source Files folder in the Solution Explorer. Include the header file (IrLrnApi.h) of the IR API library (Figure A-8).

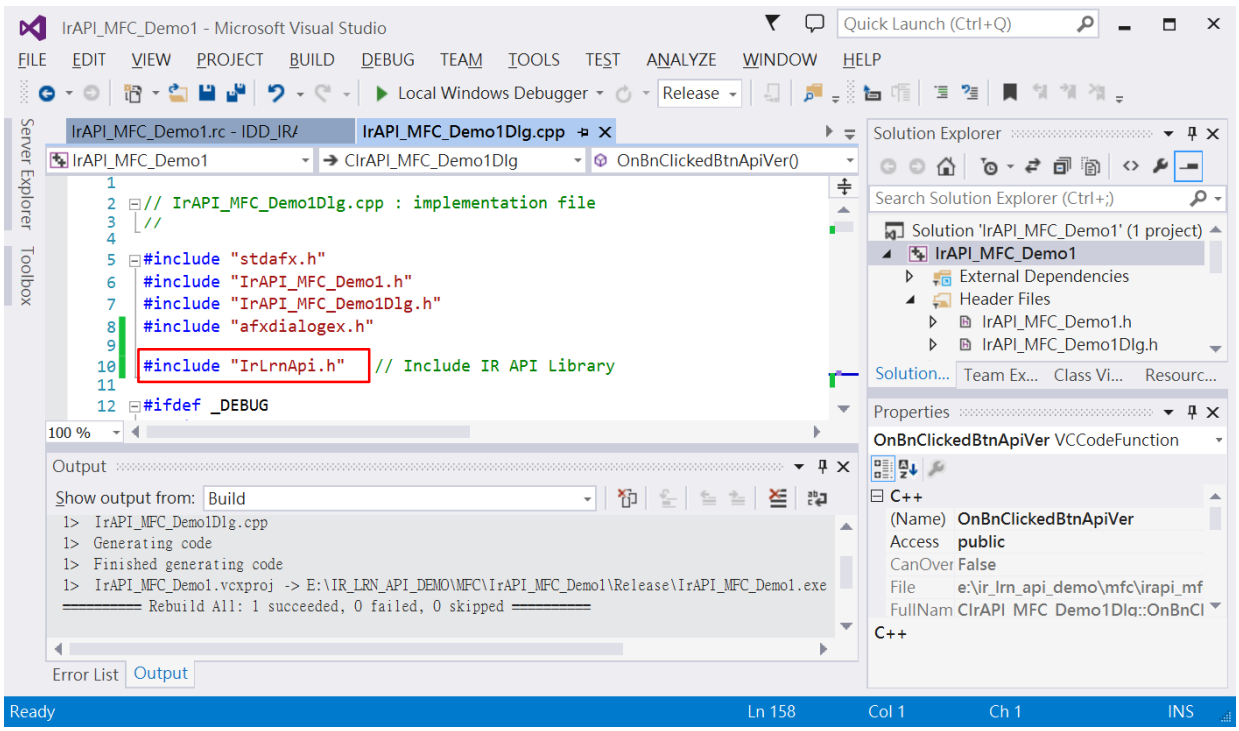


Figure A-8

b. Double-click the “DD_IRAPI_MFC_DEMO1_DIALOG” in the Dialog folder in the right pane of the Resource View as shown in Figure A-9. The dialog window will appear.

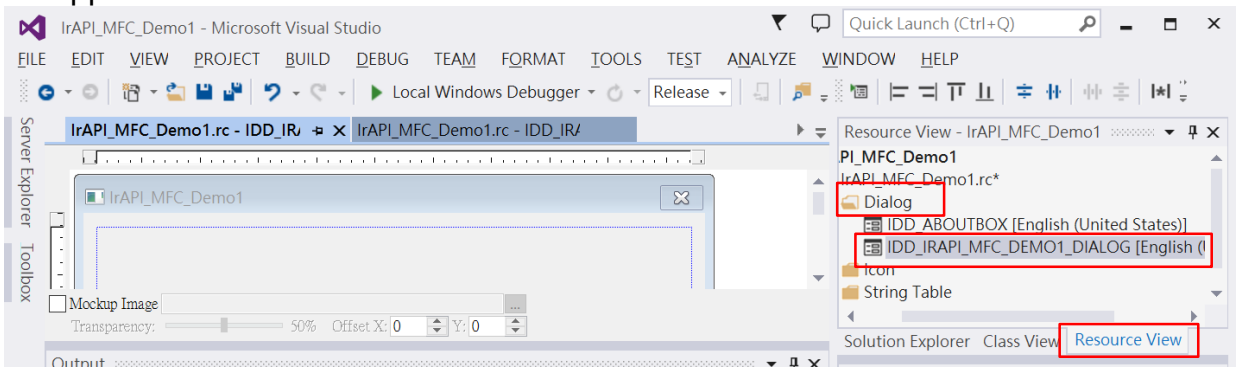


Figure A-9

c. Add Button control and Edit control to the dialog as follows.
Button Control:

ID property = “IDC_BTN_API_VER”
Caption property = “API Version”

Edit Control:

ID property = “IDC_EDIT_APIVER_RTN”

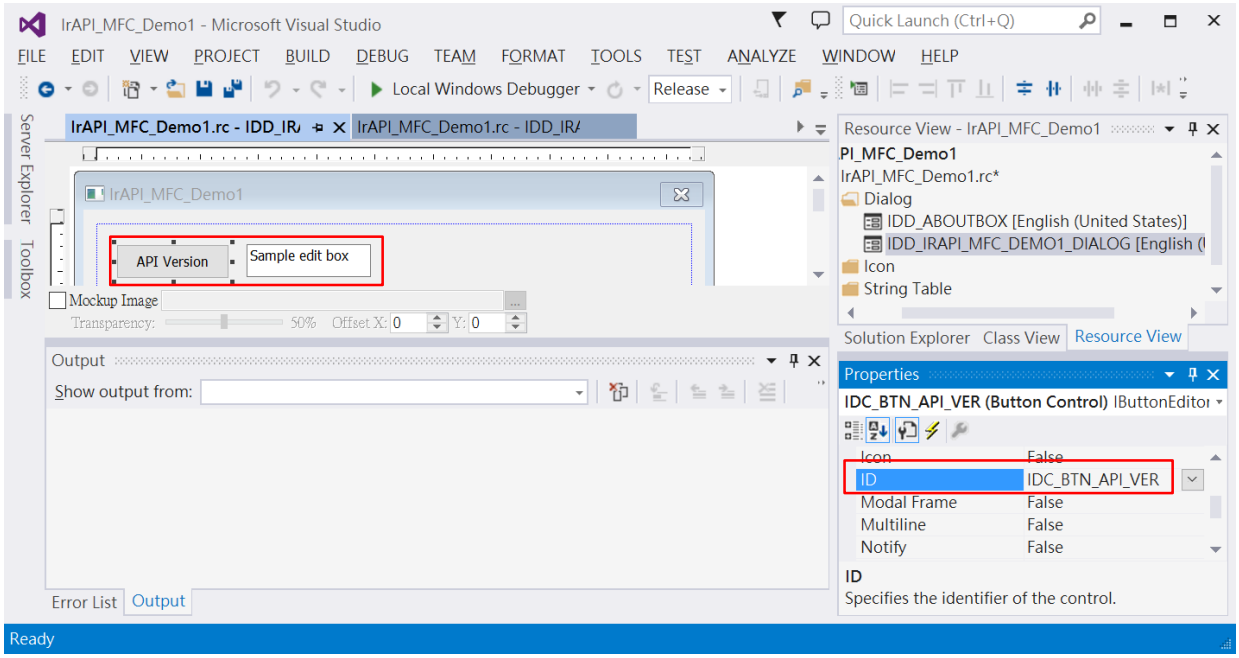


Figure A-10

d. Double-click the Button Control, it goes to the clicked event handler of the button as shown in Figure A-11. Copy the following statements to the event handler.

```
CString verStr;
CEdit *pEdit;
unsigned long vernum = 0;

vernum = IR_GetAPIVersion();
verStr.Format(_T("V%d.%d.%d.%d"), (BYTE)(vernum >> 24), (BYTE)(vernum >>
16), (BYTE)(vernum >> 8), (BYTE)vernum);

pEdit = (CEdit *)this->GetDlgItem(IDC_EDIT_APIVER_RTN);
pEdit->SetWindowTextW((LPCTSTR)verStr);
```

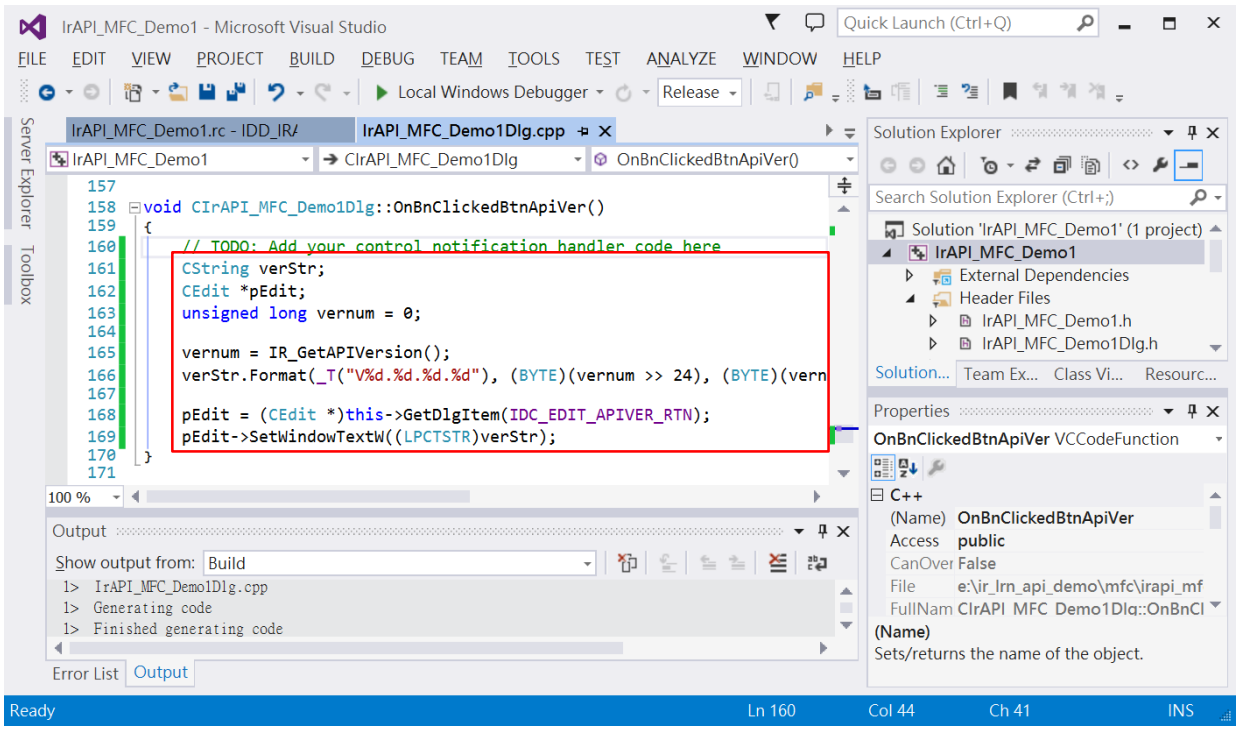


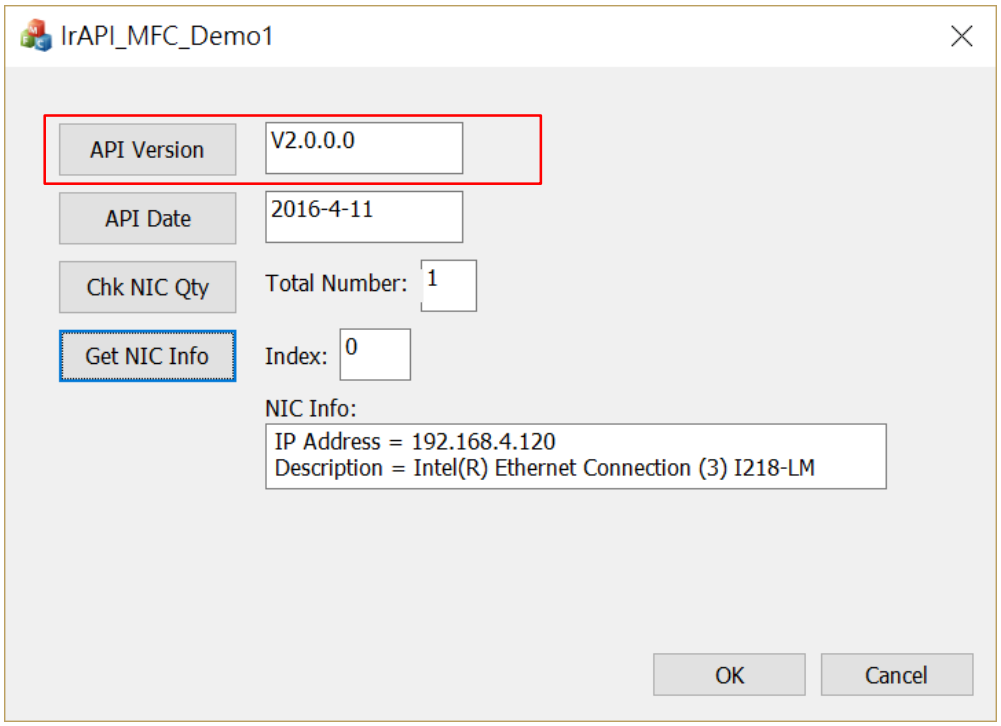
Figure A-11

e. Build the project. Click Menu [BUILD] => [Build IrAPI_MFC_Demo1]

f. Copy irlrnapi.dll & Uart.dll to the Debug or Release folder of the solution.

Figure A-12

g. Run the application executable.



A-13

Technical Support

If you have problems about the IR API library, please contact ICP DAS for technical support.
Email: service@icpdas.com