M2M RTU Library

User's Manual

Warranty

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

Warning

ICP DAS assume no liability for damages consequent to 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, nor for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

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

Tables of Content

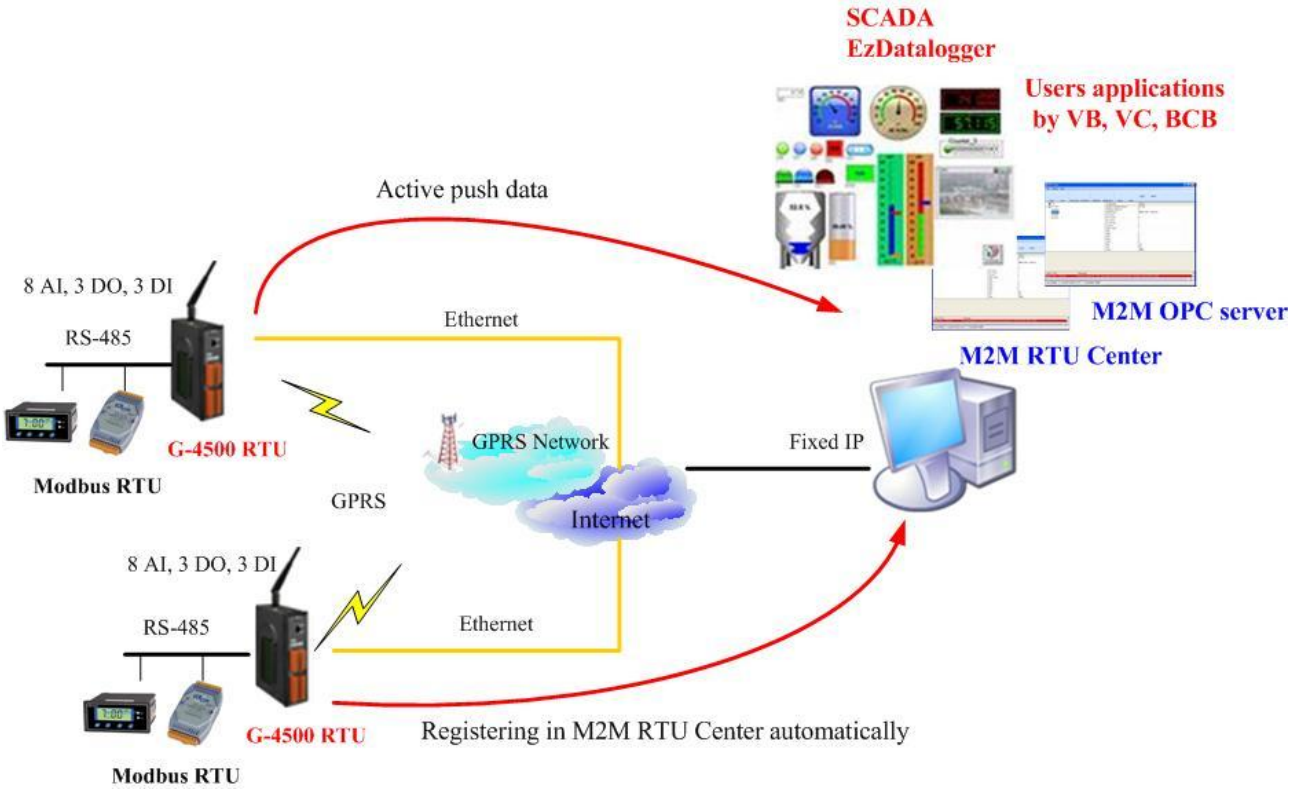
1. Overview 3
2. Flowchart of using M2M APIs 7
3. Install Library and Demo 8
4. Function Description 12
5. Data format definiens 34
6. Type code definiens 35

1. Overview

ICP DAS M2M RTU Library is a software tool package for M2M RTU products. It provides the seamless connection with M2M RTU products (G-4500 RTU, GT-540…) of ICP DAS for the user-designed system. With the APIs in this library, programmer can access M2M RTU devices by public software development environments, like VC, VB, BCB, visual studio.Net… It is easy to integrate these GPRS RTU devices to various applications including real the remote data, database management system.
Therefore, the Library can help users to apply the ICP DAS M2M RTU products in their applications to monitor the data and sends them out in real time to the control center through GPRS or Ethernet Network. Also, by combining a GPS (optional) with M2M GPRS RTU, they suddenly become a tracking system which you can often find out in the car system, marine system, etc.

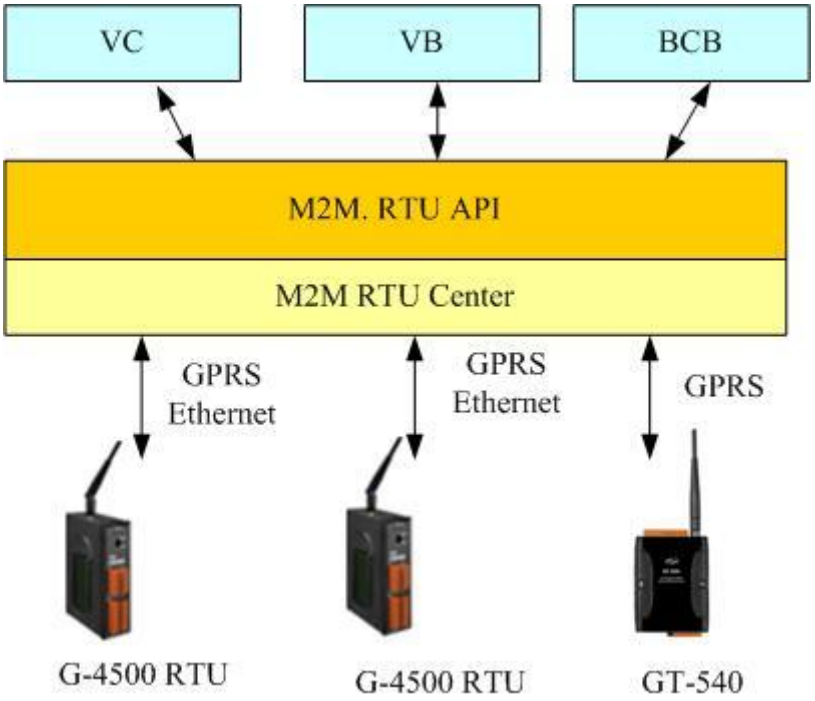
Features

- Provide simple API functions for users to reduce the development time
- Easy to perform M2M RTU devices status monitoring and control
- Help users to connect to any Modbus device to GPRS/Ethernet
- Easily manage and control distributed remote devices via GPRS/Ethernet
- Support for M2M GPRS RTU products of ICP DAS



Software Architecture

In order to reduce the load of developing application, the architecture of M2M RTU solutions for users is as the following figure. M2M RTU Center software would manage the connection from M2M RTU devices. M2M RTU Library can help engineers to access data from M2M RTU Center software. Therefore, programmer just focuses on the applications



Support Hardware

Product Type	Description
G-4500-SIM300 CR	Tri-band M2M Mini-Programmable Automation Controller (RoHS)
G-4500D-SIM300 CR	Tri-band M2M Mini-Programmable Automation Controller with LCD display (RoHS)
G-4500P-SIM300 CR	Tri-band M2M Mini-Programmable Automation Controller with GPS function (RoHS)
G-4500PD-SIM300 CR	Tri-band M2M Mini-Programmable Automation Controller with LCD display and GPS function (RoHS)
G-4500-SIM340 CR	Quad-band M2M Mini-Programmable Automation Controller (RoHS)
G-4500D-SIM340 CR	Quad-band M2M Mini-Programmable Automation Controller with LCD display (RoHS)
G-4500P-SIM340 CR	Quad-band M2M Mini-Programmable Automation Controller with GPS function (RoHS)
G-4500PD-SIM340 CR	Quad-band M2M Mini-Programmable Automation

	Controller with LCD display and GPS function (RoHS)
GD-4500-SIM340 CR	Quad-band M2M Mini-Programmable Automation Controller (RoHS)
GD-4500D-SIM340 CR	Quad-band M2M Mini-Programmable Automation Controller with LCD display (RoHS)
GD-4500P-SIM340 CR	Quad-band M2M Mini-Programmable Automation Controller with GPS function (RoHS)
GD-4500PD-SIM340 CR	Quad-band M2M Mini-Programmable Automation Controller with LCD display and GPS function (RoHS)

System Requirements

Hardware requirement

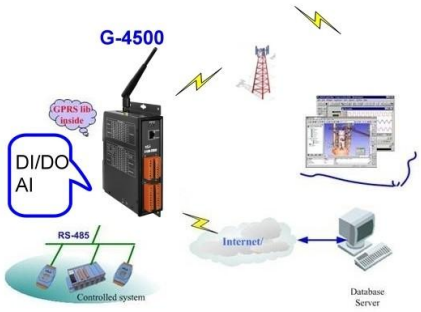
CPU Intel Pentium (Pentium 4 or above)
RAM 512 MB
HD 500 MB

Software requirement

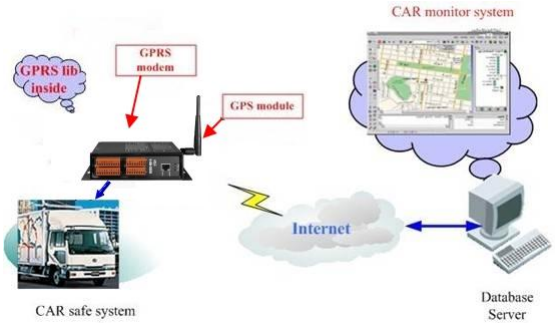
OS Windows 2000, Windows XP, Windows Vista, Windows 7
Microsoft .NET Framework Version 2.0 or above

Application

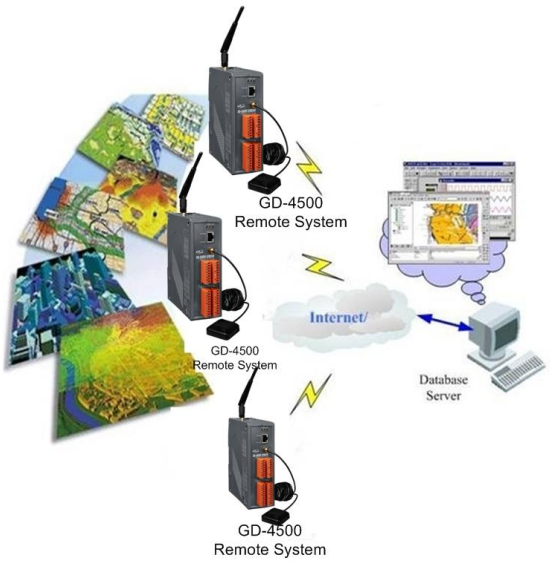
- Digital Signage
- Energy Management
- HVAC & Refrigeration
- Security & Access Control
- Vehicle tracking system



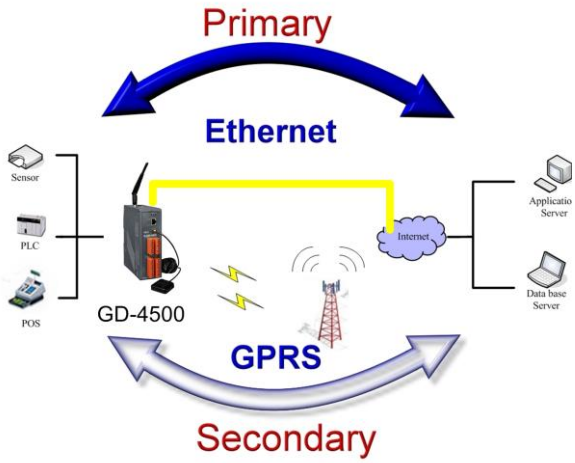
Remote Control/Monitor System



Car Monitor System

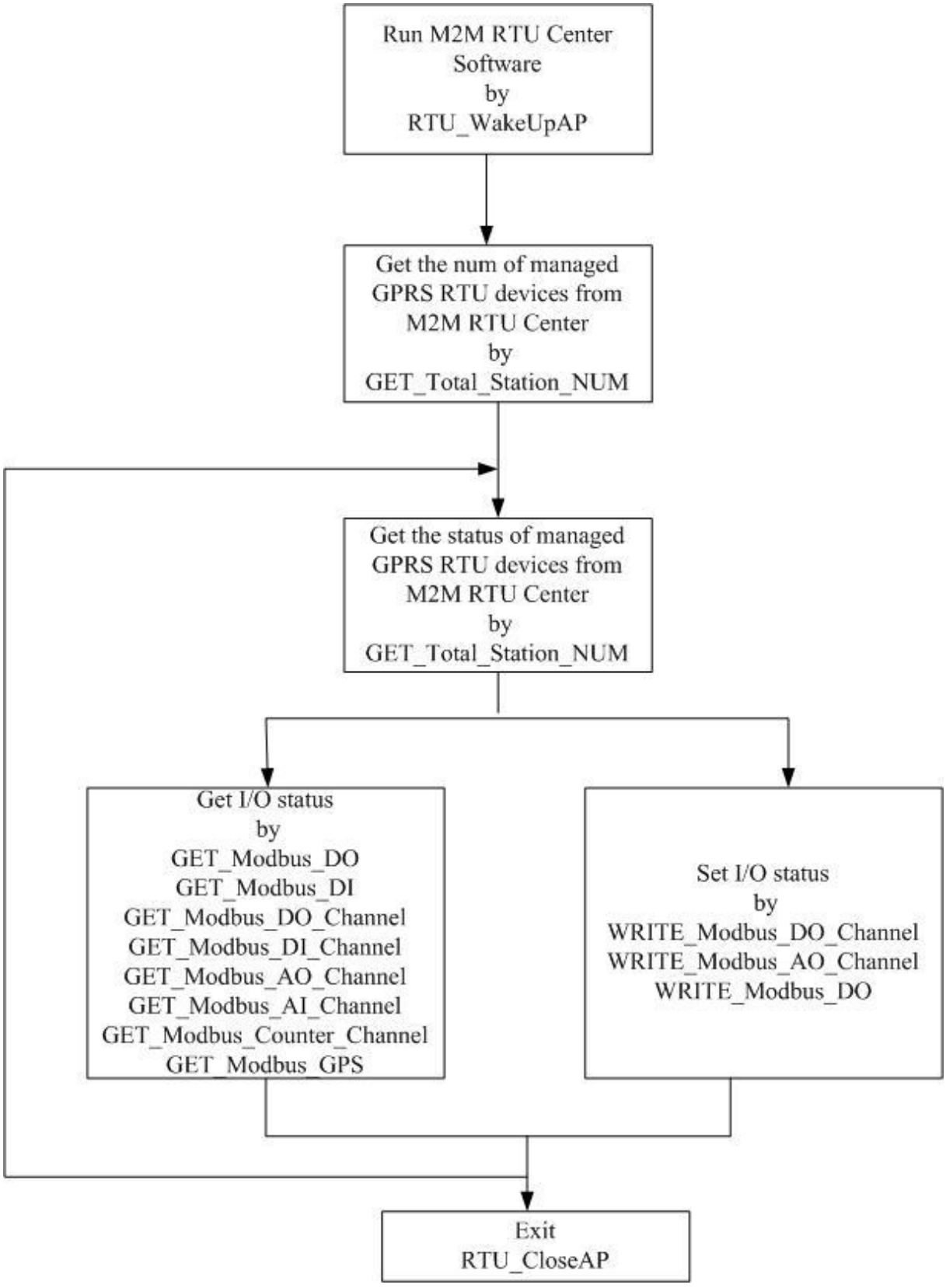


GIS system



Redundant Communication System

2. Flowchart of using M2M APIs

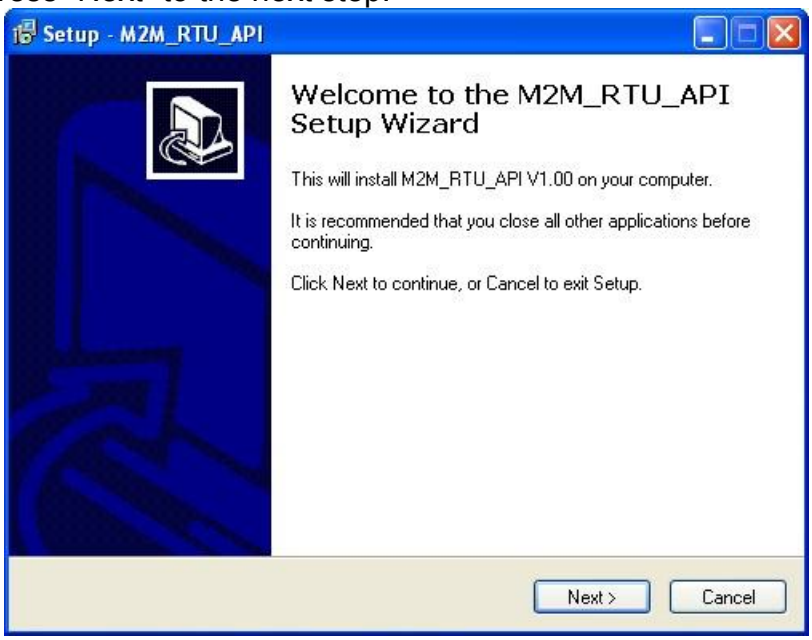


3. Install Library and Demo

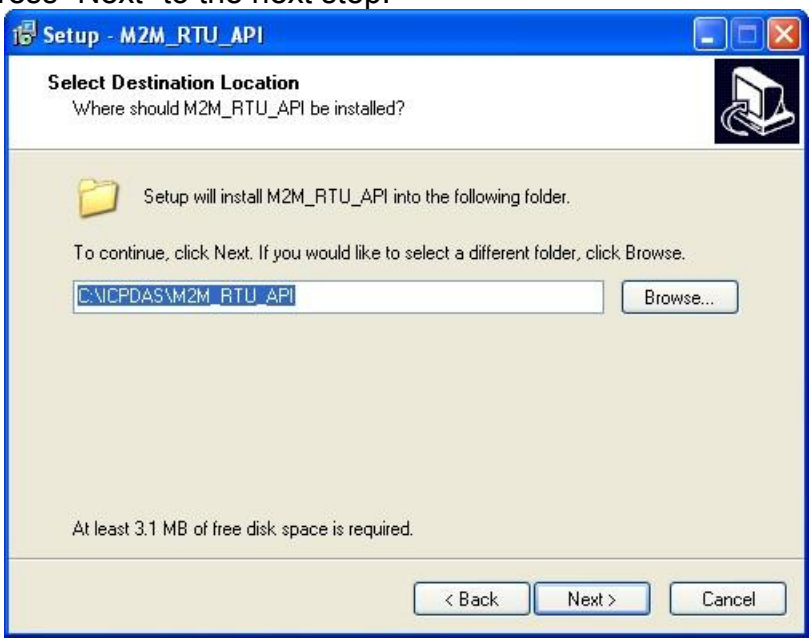
M2M_RTU_API
|
|--\Manual
|--\M2M_RTU_API_Manual.pdf
|
|--\Software
|--\RTU_Center
| |--\RTU_Center.exe
| |--\M2M_RTU.dll
| |
|--\Demo
| |--\VB6
| |--\VC6
| |--\DOT_NET

The install figure is as follows:

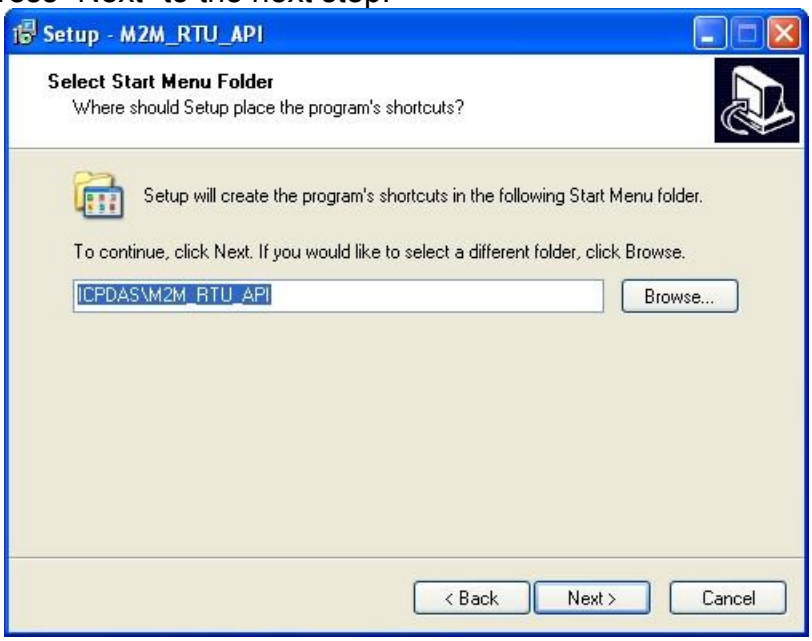
- Press "Next" to the next step.



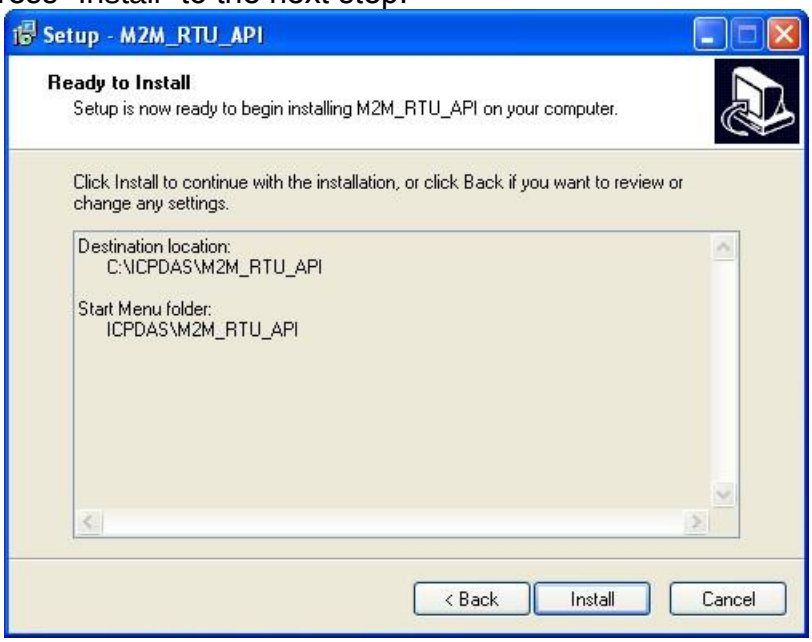
◆ Press "Next" to the next step.



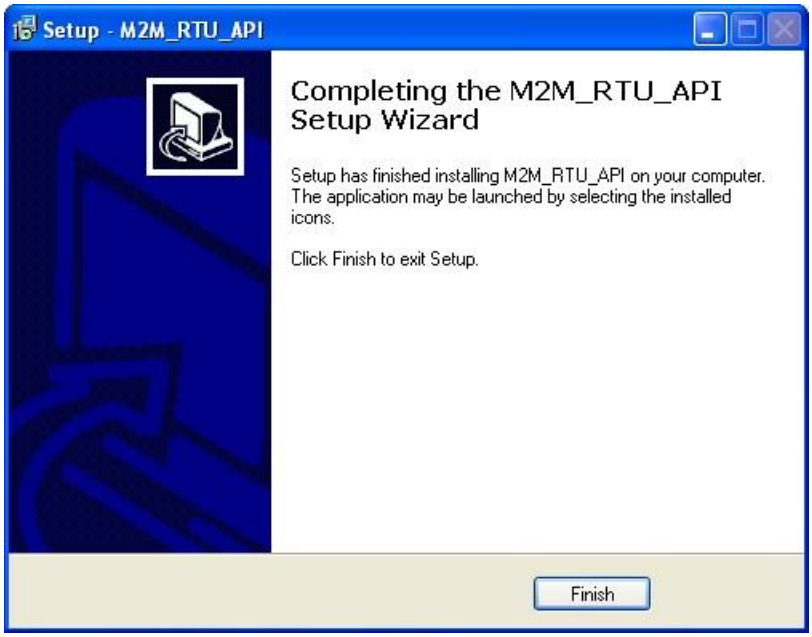
◆ Press "Next" to the next step.



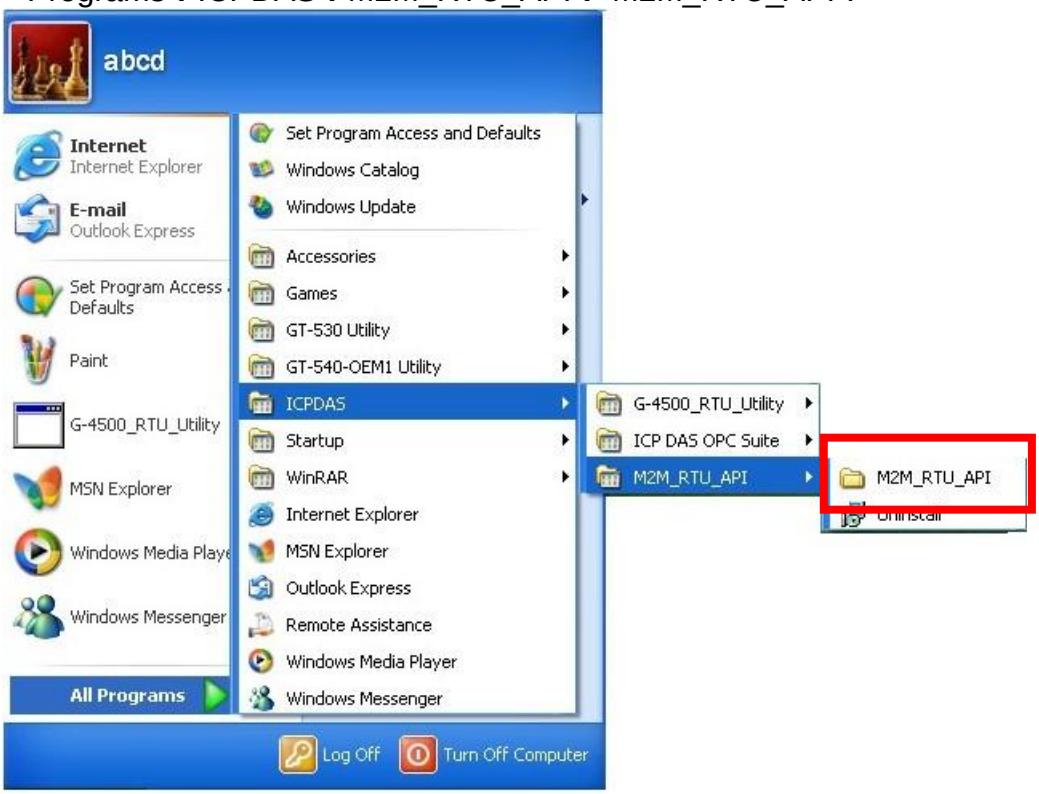
- Press "Install" to the next step.



- Press "Finish".



- Launch M2M RTU API from the start menu "Start→All Programs→ICPDAS→M2M_RTU_API→ M2M_RTU_API".



4. Function Description

DLL Function Definition and Description

All the functions provided in the M2M_RTU.dll are listed in the following table and detail information for every function is presented in the next sub-section. However, in order to make the descriptions more simply and clear, the attributes for the both the input and output parameter functions are given as **[input]** and **[output]** respectively, as shown in the following table.

Keyword	Set parameter by user before calling this function	Get the data from this parameter after calling this function
[input]	Yes	No
[output]	No	Yes

Functions Table

No.	Function Name	Description
1	GET_Total_Station_NUM	Get the number of M2M RTU devices connecting to M2M RTU Center.
2	GET_StationID_Total_Info	Get the information of total RTU stations from system M2M RTU Center.
3	GET_Modbus_DO	Get the DO status of the specified Modbus device of RTU devices.
4	GET_Modbus_DI	Get the DI status of the specified Modbus device of the RTU devices.
5	GET_Modbus_DO_Channel	Get the DO number of the specified Modbus device connecting the RTU devices.
6	GET_Modbus_DI_Channel	Get the DI number of the specified Modbus device connecting the RTU devices.
7	GET_Modbus_AO_Channel	Get the AO number of the specified Modbus device connecting the RTU devices.
8	GET_Modbus_AI_Channel	Get the AI number of the specified Modbus device connecting the RTU devices.
9	GET_Modbus_Counter_Channel	Get the AI number of the specified Modbus device connecting the RTU devices.
10	GET_Modbus_GPS	Get the AI number of the specified Modbus device connecting the RTU devices.
11	WRITE_Modbus_DO_Channel	Write the DO channel of the specified Modbus device connecting the RTU devices.
12	WRITE_Modbus_AO_Channel	Write the DO channel of the specified Modbus device connecting the RTU devices.
13	WRITE_Modbus_DO	Write the DO total channels Modbus device connecting the RTU devices.
14	RTU_WakeUpAP	Open M2M RTU Center software

15	RTU_CloseAP	Close M2M RTU Center software
16	Get_RTUWDT	Return the WDT value of the specified GPRS RTU.
17	GetDllVersion	Get the version of this DLL.
18	GetDllDate	Get the date of this DLL
19	GET_DateTime	Get data and time of last package

Structure definition

```
struct OPC_Total_Info
{
	unsigned int iStationID;
	unsigned int iModbus_Num;
	unsigned int iModule;
	unsigned char cModule_Name[40];
	unsigned int uiModule_Name_Len;
	char Modbus_Module_Name[MODBUS_DEVICE_MAX_SIZE][24];
	BYTE Slave_ID[MODBUS_DEVICE_MAX_SIZE];
	BYTE DI_Num[MODBUS_DEVICE_MAX_SIZE];
	BYTE DO_Num[MODBUS_DEVICE_MAX_SIZE];
	BYTE AI_Num[MODBUS_DEVICE_MAX_SIZE];
	BYTE AO_Num[MODBUS_DEVICE_MAX_SIZE];
	BYTE Counter_Num[MODBUS_DEVICE_MAX_SIZE];
	BYTE bEnGPS[MODBUS_DEVICE_MAX_SIZE];

}
```

Return code of function

```
#define _NOERROR                        0
#define _ERROR                          -1   //
#define _CON_FAILED                     1    //Connection is fail
#define _MB_DATA_FAILED                 2    //Modbus data is fail.
#define _INIT_FAILED                    3    //Initial is fial.
#define _STATIONID_OR_SLAVEID_ERR       4    //can't find the match StationID or
  SlaveID

#define STATION_NUM_MAX_SIZE            128  //support max. Station number
#define MODBUS_DEVICE_MAX_SIZE          11   //One station suppoers max. Modbus
devices.
```

4.1 GET_Total_Station_NUM

Description:

This function is used to get the number of M2M RTU devices connecting to M2M RTU Center.

Syntax:

int GET_Total_Station_NUM(unsigned int *iStation_Num);

Parameter:

iStation_Num: [output] The total number of GPRS RTU devices connecting to M2M RTU Center.

Return:

0: No Error
3: Initial Failed

Example:

```
unsigned Int iStation_Num = 0;
GET_Total_Station_NUM(&iStation_Num);
Print("The tatal station number are %d.", iSatation_Num);
```

4.2 GET_Total_Station_NUM

Description:

This function is used to get the information of the defined M2M RTU device connecting to M2M RTU Center.

Syntax:

int GET_StationID_Total_Info(unsigned int iNum, struct OPC_Total_Info *StationID_Total_Info)

Parameter:

iNum: [input] The total number of GPRS RTU devices connecting to M2M RTU Center.

StationID_Total_Info: [output] The information of the specified GPRS RTU device.

```
struct OPC_Total_Info
{
    unsigned int iStationID;
    unsigned int iModbus_Num;
    unsigned int iModule;.
    char Modbus_Module_Name[MODBUS_DEVICE_MAX_SIZE][24];
    BYTE Slave_ID[MODBUS_DEVICE_MAX_SIZE];
    BYTE DI_Num[MODBUS_DEVICE_MAX_SIZE];
    BYTE DO_Num[MODBUS_DEVICE_MAX_SIZE];
    BYTE AI_Num[MODBUS_DEVICE_MAX_SIZE];
    BYTE AO_Num[MODBUS_DEVICE_MAX_SIZE];
    BYTE Counter_Num[MODBUS_DEVICE_MAX_SIZE];
    BYTE bEnGPS[MODBUS_DEVICE_MAX_SIZE];
};
```

Item	Description	
iStationID	The Station ID of module	
iModbus_Num	The total number of Modbus of module	
Slave_ID[MODBUS_DEVICE_MAX_SIZE];	The slave ID of assign modbus device	
DI_Num[MODBUS_DEVICE_MAX_SIZE]	The DI number of assign modbus device	
DO_Num[MODBUS_DEVICE_MAX_SIZE];	The DO number of assign modbus device	
AI_Num[MODBUS_DEVICE_MAX_SIZE]	The AI number of assign modbus device	
AO_Num[MODBUS_DEVICE_MAX_SIZE]	The AO number of assign modbus device	
Counter_Num[MODBUS_DEVICE_MAX_SIZE]	The Counter number of assign modbus device	
bEnGPS[MODBUS_DEVICE_MAX_SIZE]	The GPS number of assign modbus device	

Return:
0: No Error
1: don't establish the connection
2: Modbus Data are invalid
4: Station ID or Modbus Slave ID is invalid.

Example:

```
unsigned int iStation_Num = 0;
unsigned int i = 0;
unsigned int j = 0;
struct OPC_Total_Info opc_total_info;


GET_Total_Station_NUM(&iStation_Num);
Print("The tatal station number are %d.\r\n", iSatation_Num);

for(i=0; i< iStation_Num; i++)
{
    GET_StationID_Total_Info(i, & opc_total_info);
    Print("StationID=%d\r\n", opc_total_info. iStationID);
    Print("The total number of Modbus of module=%d\r\n", opc_total_info. iModbus_Num);

    for(j=0; j< opc_total_info. iModbus_Num; j++)
    {
        Print("Slave_ID=%d\r\n", opc_total_info. Slave_ID[j]);
        Print(" DI_Num=%d\r\n", opc_total_info.DI_Num[j]);
        Print(" DO_Num=%d\r\n", opc_total_info.DO_Num[j]);
        Print(" AI_Num=%d\r\n", opc_total_info.AI_Num[j]);
        Print(" AO_Num=%d\r\n", opc_total_info.AO_Num[j]);
        Print(" Counter_Num=%d\r\n", opc_total_info. Counter_Num [j]);
        Print(" bEnGPS =%d\r\n", opc_total_info. bEnGPS [j]);

    }
}
```

Note: If the Slave ID is 255, it represents the local IO of G-4500 or GT-540.

4.3 GET_Modbus_DO

Description:

This function is used to get the DO value of the defined Modbus RTU in M2M RTU connecting to M2M RTU Center.

Syntax:

int GET_Modbus_DO(unsigned int iStationID, BYTE *bModbus_Module_Name, BYTE bModbus_SlaveID, unsigned long *Value);

Parameter:

bModbus_Module_Name: [input]
iStationID: [input]
bModbus_SlaveID: [input]
Value: [output]

Return:

0: No Error
1: don't establish the connection
2: Modbus Data are invalid
4: Station ID or Modbus Slave ID is invalid.

Example:

```
unsigned int iStationID = 1; //StationID: 1
BYTE bModbus_Module_Name[] = "Local IO";
BYTE bModbus_SlaveID = 255;    //local IO
unsigned long Value = 0;

GET_Modbus_DO(iStationID, bModbus_Module_Name, bModbus_SlaveID, &Value);
Print("StationID:1,SlaveID:255(local total DO)=%lu\r\n", Value);
```

Note: If the Slave ID is 255, it represents the local IO of G-4500 or GT-540.

4.4 GET_Modbus_DI

Description:

This function is used to get the DI value of the defined the Modbus RTU devices in M2M RTU connecting to M2M RTU Center.

Syntax:

int GET_Modbus_DI(unsigned int iStationID, BYTE *bModbus_Module_Name, BYTE bModbus_SlaveID, unsigned long *Value);

Parameter:

iStationID: [input]
bModbus_Module_Name: [input]
bModbus_SlaveID: [input]
Value: [output]

Return:

0: No Error
1: don't establish the connection
2: Modbus Data are invalid
4: Station ID or Modbus Slave ID is invalid.

Example:

```
unsigned int iStationID = 1; //StationID: 1
BYTE bModbus_Module_Name[] = "Local IO";
BYTE bModbus_SlaveID = 255;    //local IO
unsigned long Value = 0;

GET_Modbus_DI(iStationID, bModbus_Module_Name, bModbus_SlaveID, &Value);
Print("StationID:1,SlaveID:255(local total DI)=%lu\r\n", Value);
```

Note: If the Slave ID is 255, it represents the local IO of G-4500 or GT-540.

4.5 GET_Modbus_DO_Channel

Description:

This function is used to get the specified DO channel value of the defined Modbus RTU devices in M2M RTU connecting to M2M RTU Center.

Syntax:

int GET_Modbus_DO_Channel(unsigned int iStationID, BYTE *bModbus_Module_Name, BYTE bModbus_SlaveID, BYTE channel, BYTE *Value);

Parameter:

iStationID: [input]
bModbus_Module_Name: [input]
bModbus_SlaveID: [input]
channel: [input]
Value: [output]

Return:

0: No Error
1: don't establish the connection
2: Modbus Data are invalid
4: Station ID or Modbus Slave ID is invalid.

Example:

```
unsigned int iStationID = 1; //StationID: 1
BYTE bModbus_Module_Name[] = "Local IO";
BYTE bModbus_SlaveID = 255;    //local DO
BYTE channel = 1; //Ch1 of local DO
BYTE Value = 0;

GET_Modbus_DO_Channel(iStationID, bModbus_Module_Name, bModbus_SlaveID, channel, &Value);
Print("StationID:1,SlaveID:255(local DO ch1)=%d\r\n", Value);
```

Note: If the Slave ID is 255, it represents the local IO of G-4500 or GT-540.

4.6 GET_Modbus_DI_Channel

Description:

This function is used to get the specified DI channel value of the defined Modbus RTU in M2M RTU connecting to M2M RTU Center.

Syntax:

int GET_Modbus_DI_Channel(unsigned int iStationID, BYTE *bModbus_Module_Name, BYTE bModbus_SlaveID, BYTE channel, BYTE *Value);

Parameter:

iStationID: [input]
bModbus_Module_Name: [input]
bModbus_SlaveID: [input]
channel: [input]
Value: [output]

Return:

0: No Error
1: don't establish the connection
2: Modbus Data are invalid
4: Station ID or Modbus Slave ID is invalid.

Example:

```
unsigned int iStationID = 1; //StationID: 1
BYTE bModbus_Module_Name[] = "Local IO";
BYTE bModbus_SlaveID = 255;    //local DO
BYTE channel = 0; //Ch1 of local DO
BYTE Value = 0;

GET_Modbus_DI_Channel(iStationID, bModbus_Module_Name, bModbus_SlaveID, channel, &Value);
Print("StationID:1,SlaveID:255(local DI ch0)=%d\r\n", Value);
```

Note: If the Slave ID is 255, it represents the local IO of G-4500 or GT-540.

4.7 GET_Modbus_AO_Channel

Description:

This function is used to get the specified AO channel value of the defined Modbus RTU in M2M RTU connecting to M2M RTU Center.

Syntax:

int GET_Modbus_AO_Channel(unsigned int iStationID, BYTE *bModbus_Module_Name, BYTE bModbus_SlaveID, BYTE channel, BYTE *DataFormat, BYTE *Type, unsigned short *Value);

Parameter:

iStationID: [input]
bModbus_Module_Name: [input]
bModbus_SlaveID: [input]
channel: [input]
DataFormat: [output]
Value: [output]

Return:

0: No Error
1: don't establish the connection
2: Modbus Data are invalid
4: Station ID or Modbus Slave ID is invalid.

Example:

```
unsigned int iStationID = 1; //StationID: 1
BYTE bModbus_Module_Name[] = "Local IO";
BYTE bModbus_SlaveID = 255;    //local AO
BYTE channel = 1; //Ch1 of local AO
BYTE DataFormat;
BYTE Type;
unsigned short Value = 0;

GET_Modbus_AO_Channel(iStationID, bModbus_Module_Name, bModbus_SlaveID, channel, &DataFormat, &Type, &Value);
Print("StationID:1,SlaveID:255(local AO ch1)=%d\r\n", Value);
```

Note: If the Slave ID is 255, it represents the local IO of G-4500 or GT-540.

4.8 GET_Modbus_AI_Channel

Description:

This function is used to get the specified AI channel value of the defined Modbus RTU in M2M RTU connecting to M2M RTU Center.

Syntax:

int GET_Modbus_AI_Channel(unsigned int iStationID, BYTE *bModbus_Module_Name, BYTE bModbus_SlaveID, BYTE channel, BYTE *DataFormat, BYTE *Type, unsigned short *Value);

Parameter:

iStationID: [input]
bModbus_Module_Name: [input]
bModbus_SlaveID: [input]
channel: [input]
DataFormat: [output]
Value: [output]

Return:

0: No Error
1: don't establish the connection
2: Modbus Data are invalid
4: Station ID or Modbus Slave ID is invalid.

Example:

```
unsigned int iStationID = 1; //StationID: 1
BYTE bModbus_Module_Name[] = "Local IO";
BYTE bModbus_SlaveID = 255;    //local AI
BYTE channel = 0; //Ch0 of local AI
BYTE DataFormat;
BYTE Type;
unsigned short Value = 0;

GET_Modbus_AI_Channel(iStationID, bModbus_Module_Name, bModbus_SlaveID, channel, &DataFormat, &Type, &Value);
Print("StationID:1,SlaveID:255(local AI ch0)=%d\r\n", Value);
```

Note: If the Slave ID is 255, it represents the local IO of G-4500 or GT-540.

4.9 GET_Modbus_Counter_Channel

Description:

This function is used to get the specified counter channel value of the defined Modbus RTU in M2M RTU connecting to M2M RTU Center.

Syntax:

int GET_Modbus_Counter_Channel(unsigned int iStationID, BYTE *bModbus_Module_Name, BYTE bModbus_SlaveID, BYTE channel, unsigned int *Value);

Parameter:

iStationID: [input]
bModbus_Module_Name: [input]
bModbus_SlaveID: [input]
channel: [input]
Value: [output]

Return:

0: No Error
1: don't establish the connection
2: Modbus Data are invalid
4: Station ID or Modbus Slave ID is invalid.

Example:

```
unsigned int iStationID = 1; //StationID: 1
BYTE bModbus_Module_Name[] = "Local IO";
BYTE bModbus_SlaveID = 255;    //local Counter
BYTE channel = 0; //Ch0 of local Counter
unsigned int Value = 0;

GET_Modbus_Counter_Channel(iStationID, bModbus_Module_Name, bModbus_SlaveID, channel, &Value);
Print("StationID:1,SlaveID:255(local Counter ch0)=%lu\r\n", Value);
```

Note: If the Slave ID is 255, it represents the local IO of G-4500 or GT-540.

4.10 GET_Modbus_GPS

Description:

This function is used to get the GPS information of the M2M RTU connecting to M2M RTU Center.

Syntax:

int GET_Modbus_GPS(unsigned int iStationID, BYTE *bModbus_Module_Name, BYTE bModbus_SlaveID, char *Value);

Parameter:

iStationID: [input]
bModbus_Module_Name: [input]
bModbus_SlaveID: [input]
Value: [output]

Return:

0: No Error
1: don't establish the connection
2: Modbus Data are invalid
4: Station ID or Modbus Slave ID is invalid
5: Module didn't enable GPS function

Example:

```
unsigned int iStationID = 1; //StationID: 1
BYTE bModbus_Module_Name[] = "Local IO";
BYTE bModbus_SlaveID = 255;    //local Counter
char Value[255];

GET_Modbus_GPS(iStationID, bModbus_Module_Name, bModbus_SlaveID, Value);
Print("StationID:1,SlaveID:255(local GPS)=%s\r\n"), Value);
```

Note: If the Slave ID is 255, it represents the local IO of G-4500 or GT-540.

4.11 WRITE_Modbus_DO_Channel

Description:

This function is used to set the specified DO channel value of the defined Modbus RTU in M2M RTU connecting to M2M RTU Center.

Syntax:

int WRITE_Modbus_DO_Channel(unsigned int iStationID, BYTE *bModbus_Module_Name, BYTE bModbus_SlaveID, BYTE bChannel, BYTE bValue);

Parameter:

iStationID: [input]
bModbus_Module_Name: [input]
bModbus_SlaveID: [input]
bChannel: [input]
Value: [output]

Return:

0: No Error
1: don't establish the connection
2: Modbus Data are invalid
4: Station ID or Modbus Slave ID is invalid.

Example:

```
unsigned int iStationID = 1; //StationID: 1
BYTE bModbus_Module_Name[] = "Local IO";
BYTE bModbus_SlaveID = 255;    //local Counter
BYTE bChannel = 0;
BYTE Value = 1;
int iRet = 0;

iRet = WRITE_Modbus_DO_Channel(iStationID, bModbus_Module_Name,
bModbus_SlaveID, bChannel, Value);
Print("WRITE_Modbus_DO_Channel(StationID:1,SlaveID:255)(local DO ch0)=%d\r\n",
iRet);
```

Note: If the Slave ID is 255, it represents the local IO of G-4500 or GT-540.

4.12 WRITE_Modbus_AO_Channel

Description:

This function is used to set the specified AO channel value of the defined Modbus RTU in M2M RTU connecting to M2M RTU Center.

Syntax:

int WRITE_Modbus_AO_Channel(unsigned int iStationID, BYTE *bModbus_Module_Name, BYTE bModbus_SlaveID, BYTE bChannel, unsigned short usValue);

Parameter:

iStationID: [input]
bModbus_Module_Name: [input]
bModbus_SlaveID: [input]
bChannel: [input]
usValue: [output]

Return:

0: No Error
1: don't establish the connection
2: Modbus Data are invalid
4: Station ID or Modbus Slave ID is invalid.

Example:

```
unsigned int iStationID = 1; //StationID: 1
BYTE bModbus_Module_Name[] = "Local IO";
BYTE bModbus_SlaveID = 255;    //local Counter
BYTE bChannel = 0;
unsigned short Value = 1;
int iRet = 0;

iRet = WRITE_Modbus_AO_Channel(iStationID, bModbus_Module_Name,
bModbus_SlaveID, bChannel, Value);
Print("WRITE_Modbus_AO_Channel(StationID:1,SlaveID:255)(local AO ch0)=%d\r\n",
iRet);
```

Note: If the Slave ID is 255, it represents the local IO of G-4500 or GT-540.

4.13 WRITE_Modbus_DO

Description:

This function is used to set the specified DO value of the defined Modbus RTU in M2M RTU connecting to M2M RTU Center.

Syntax:

int WRITE_Modbus_DO(unsigned int iStationID, BYTE *bModbus_Module_Name, BYTE bModbus_SlaveID, unsigned long ulValue);

Parameter:

iStationID: [input]
bModbus_Module_Name: [input]
bModbus_SlaveID: [input]
bChannel: [input]
usValue: [output]

Return:

0: No Error
1: don't establish the connection
2: Modbus Data are invalid
4: Station ID or Modbus Slave ID is invalid.

Example:

```
unsigned int iStationID = 1; //StationID: 1
BYTE bModbus_Module_Name[] = "Local IO";
BYTE bModbus_SlaveID = 255;    //local Counter
BYTE bChannel = 0;
unsigned long Value = 7;
int iRet = 0;

iRet = WRITE_Modbus_DO(iStationID, bModbus_Module_Name, bModbus_SlaveID, Value);
Print("WRITE_Modbus_DO(StationID:1,SlaveID:255)(total local DO)=%d\r\n", iRet);
```

Note: If the Slave ID is 255, it represents the local IO of G-4500 or GT-540.

4.14 RTU_WakeUpAP

Description:

This function is used to open M2M RTU Center software.

Syntax:

int RTU_WakeUpAP(BYTE bMethod);

Parameter:

bMethod: [input]

0: from OPC Server
1: from Other Server

Return:

0: No Error

Example:

```
BYTE bMethod = 1;
RTU_WakeUpAP(1);
```

4.15 RTU_CloseAP

Description:

This function is used to close M2M RTU Center software.

Syntax:

int RTU_CloseAP(void);

Parameter:

Return:

0: No Error

Example:

RTU_CloseAP();

4.16 Get_RTUWDT

Description:

This function is used to get the WDT value of M2M RTU Center. The M2M RTU Center will add 1 every second. If the Get_RTUWDT function doesn't receive add 1 every second, please restart the M2M RTU Center software.

Syntax:

unsigned int Get_RTUWDT(void);

Parameter:

Return:

0: No Error

Example:

Get_RTUWDT ();

4.17 GetDllVersion

Description:

This function is used to get the Dll version of API.

Syntax:

unsigned int GetDllVersion(void);

Parameter:

Return:

0: No Error

Example:

```
int iLib = 0;
iLib = GetDllVersion();
Print("GPRS_RTU.dll vesion %X.%02X\r\n", iLib>>8, iLib&0xff);
```

4.18 GetDllDate

Description:

This function is used to get the Dll date of API.

Syntax:

void GetDllDate(char *cDate);

Parameter:

Return:

0: No Error

Example:

```
char cDate[20];
GetDllDate(cDate);
Print("GPRS_RTU.dll date:%s\r\n", cDate);
```

4.19 GET_DateTime

Description:

This function is used to get last package date and time.

Syntax:

Int GET_DateTime(unsigned int iStationID, int *iYear, int *iMonth, int *iDay, int *iHour, int *iMinute, int *iSec);

Parameter:

iStationID: [input]
iYear: [output]
iMonth: [output]
iDay: [output]
iHour: [output]
iMinute: [output]
iSec: [output]

Return:

0: No Error
1: don't establish the connection
4: Station ID is invalid.

Example:

```
unsigned int iStationID = 1; //StationID: 1
int iYear = 0;
int iMonth = 0;
int iDay = 0;
int iHour = 0;
int iMinute = 0;
int iSec = 0;
GET_DateTime(iStationID, &iYear, &iMonth, &iDay, &iHour, &iMinute, &iSec);
Print("Year=%d\r\nMonth=%d\r\nDay=%d\r\nHour=%d\r\nMinute=%d\r\nSec=%d\r\n", iYear, iMonth, iDay, iHour, iMinute, iSec);
```

5. Data format definiens

Data format	Description
0	engineering unit.
1	2's complement
2	hexadecimal
3	% of FSR
254	Counter hex
255	Unknown

6. Type code definiens

AI Type code

AI Type Code	Input Type	Data Format	+F.S	-F.S.
00	-15 to +15 mV	Engineering unit	+15.000	-15.000
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
01	-50 to +50 mV	Engineering unit	+50.000	-50.000
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
02	-100 to +100 mV	Engineering unit	+100.000	-100.000
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
03	-500 to +500 mV	Engineering unit	+500.00	-500.00
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
04	-1 to +1 V	Engineering unit	+1.0000	-1.0000
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
05	-2.5 to +2.5 V	Engineering unit	+2.5000	-2.5000
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
06	-20 to +20 mA	Engineering unit	+20.000	-20.000
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
07	4 to 20 mA	Engineering unit	+20.000	+04.000
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
08	-10 to +10 V	Engineering unit	+10.000	-10.000
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
09	-5 to +5 V	Engineering unit	+5.0000	-5.0000
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
10	-1 to +1 V	Engineering unit	+1.0000	-1.0000
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000

11	-500 to +500 mV	Engineering unit	+500.00	-500.00
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
12	-150 to +150 mV	Engineering unit	+150.00	-150.00
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
13	-20 to +20 mA	Engineering unit	+20.000	-20.000
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
14	-210 to 760 °C	Engineering unit	+760.00	-210.00
		% of FSR	+100.00	-027.63
		2's comp HEX	7FFF	DCA2
15	-270 to 1372 °C	Engineering unit	+1372.0	-0270.0
		% of FSR	+100.00	-019.68
		2's comp HEX	7FFF	E6D0
16	-270 to 400 °C	Engineering unit	+400.00	-270.00
		% of FSR	+100.00	-067.5
		2's comp HEX	7FFF	A99A
17	-270 to 1000 °C	Engineering unit	+1000.0	-0270.0
		% of FSR	+100.00	-027.00
		2's comp HEX	7FFF	DD71
18	0 to 1768 °C	Engineering unit	+1768.0	+0000.0
		% of FSR	+100.00	+000.00
		2's comp HEX	7FFF	0000
19	0 to 1768 °C	Engineering unit	+1768.0	+0000.0
		% of FSR	+100.00	+000.00
		2's comp HEX	7FFF	0000
20	0 to 1802 °C	Engineering unit	+1820.0	+0000.0
		% of FSR	+100.00	+000.00
		2's comp HEX	7FFF	0000
21	-270 to 1300 °C	Engineering unit	+1300.0	-0270.00
		% of FSR	+100.00	-020.77
		2's comp HEX	7FFF	E56B
22	0 to 2320 °C	Engineering unit	+2320.0	+0000.0
		% of FSR	+100.00	+000.00
		2's comp HEX	7FFF	0000
23	-200 to 800 °C	Engineering unit	+800.00	-200.00
		% of FSR	+100.00	-025.00
		2's comp HEX	7FFF	E000

24	-200 to 100 °C	Engineering unit	+100.00	-200.00
		% of FSR	+050.00	-100.00
		2's comp HEX	4000	8000
25	-200 to 900 °C	Engineering unit	+900.00	-200.00
		% of FSR	+100.00	-022.22
		2's comp HEX	7FFF	E38E
26	0 to 20 mA	Engineering unit	+20.000	+00.000
		% of FSR	+100.00	+000.00
		2's comp HEX	FFFF	0000
27	-150 to +150 V	Engineering unit	+150.00	-150.00
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
28	-50 to +50 V	Engineering unit	+50.000	-50.000
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
32	-100 to 100 °C	Engineering unit	+100.00	-100.00
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
33	0 to 100 °C	Engineering unit	+100.00	-100.00
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	0000
34	0 to 200 °C	Engineering unit	+200.00	+000.00
		% of FSR	+100.00	+000.00
		2's comp HEX	7FFF	0000
35	0 to 600 °C	Engineering unit	+600.00	+000.00
		% of FSR	+100.00	+000.00
		2's comp HEX	7FFF	0000
36	-100 to 100 °C	Engineering unit	+100.00	-100.00
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
37	0 to 100 °C	Engineering unit	+100.00	+000.00
		% of FSR	+100.00	+000.00
		2's comp HEX	7FFF	0000
38	0 to 200 °C	Engineering unit	+200.00	+000.00
		% of FSR	+100.00	+000.00
		2's comp HEX	7FFF	0000
39	0 to 600 °C	Engineering unit	+600.00	+000.00
		% of FSR	+100.00	+000.00

		2's comp HEX	7FFF	0000
40	-80 to 100 °C	Engineering unit	+100.00	-080.00
		% of FSR	+100.00	-080.00
		2's comp HEX	7FFF	999A
41	0 to 100 °C	Engineering unit	+100.00	+000.00
		% of FSR	+100.00	+000.00
		2's comp HEX	7FFF	0000
42	-200 to 600 °C	Engineering unit	+600.00	-200.00
		% of FSR	+100.00	-033.33
		2's comp HEX	7FFF	D556
43	-20 to 150 °C	Engineering unit	+150.00	-020.00
		% of FSR	+100.00	-013.33
		2's comp HEX	7FFF	EEEF
44	0 to 200 °C	Engineering unit	+200.00	+000.00
		% of FSR	+100.00	+000.00
		2's comp HEX	7FFF	0000
45	-20 to 150 °C	Engineering unit	+150.00	-020.00
		% of FSR	+100.00	-013.33
		2's comp HEX	7FFF	EEEF
46	-200 to 200 °C	Engineering unit	+200.00	-200.00
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
47	-200 to 200 °C	Engineering unit	+200.00	-200.00
		% of FSR	+100.00	-100.00
		2's comp HEX	7FFF	8000
128	-200 to 600 °C	Engineering unit	+600.00	-200.00
		% of FSR	+100.00	-033.33
		2's comp HEX	7FFF	D556
129	-200 to 600 °C	Engineering unit	+600.00	-200.00
		% of FSR	+100.00	-033.33
		2's comp HEX	7FFF	D556
130	-50 to 150 °C	Engineering unit	+150.00	-050.00
		% of FSR	+100.00	-033.33
		2's comp HEX	7FFF	D556
131	-60 to 180 °C	Engineering unit	+180.00	-060.00
		% of FSR	+100.00	-033.33
		2's comp HEX	7FFF	D556

254 (Counter)	0 to 4294967295	Counter Value	Counter Value	Counter Value
255 (Customer)	0 to 0	Customer define	Customer define	Customer define

AOType code

AO Type Code	Input Type	Data Format	+F.S	-F.S.
00	0 to 20 mA	Engineering unit	+20.000	00.000
		% of FSR	+100.00	+000.00
		2's comp HEX	FFF	0000
01	4 to 20 mA	Engineering unit	+20.000	04.000
		% of FSR	+100.00	+000.00
		2's comp HEX	FFF	0000
02	0 to 10 V	Engineering unit	+10.000	00.000
		% of FSR	+100.00	+000.00
		2's comp HEX	FFF	0000
48	0 to 20 mA	Engineering unit	+20.000	00.000
		% of FSR	+100.00	+000.00
		2's comp HEX	FFF	0000
49	4 to 20 mA	Engineering unit	+20.000	04.000
		% of FSR	+100.00	+000.00
		2's comp HEX	FFF	0000
50	0 to 10 V	Engineering unit	+10.000	00.000
		% of FSR	+100.00	+000.00
		2's comp HEX	FFF	0000
51	-10 to +10 V	Engineering unit	+10.000	-10.000
		% of FSR	+100.00	+000.00
		2's comp HEX	FFF	0000
52	0 to +5V	Engineering unit	+05.000	-05.000
		% of FSR	+100.00	+000.00
		2's comp HEX	FFF	0000
53	-5 to +5V	Engineering unit	+05.000	-05.000
		% of FSR	+100.00	+000.00
		2's comp HEX	FFF	0000
255 (Customer)	0 to 0	Customer define	Customer define	Customer define

History of version

Revision

Version	By	Date	Description
1.00	Yide	2010/03/28	
1.01	Yide	2010/06/02	
1.02	Yide	2010/08/11	