How to use LinPAC to access M-7000 remote I/O modules with Modbus Protocol?

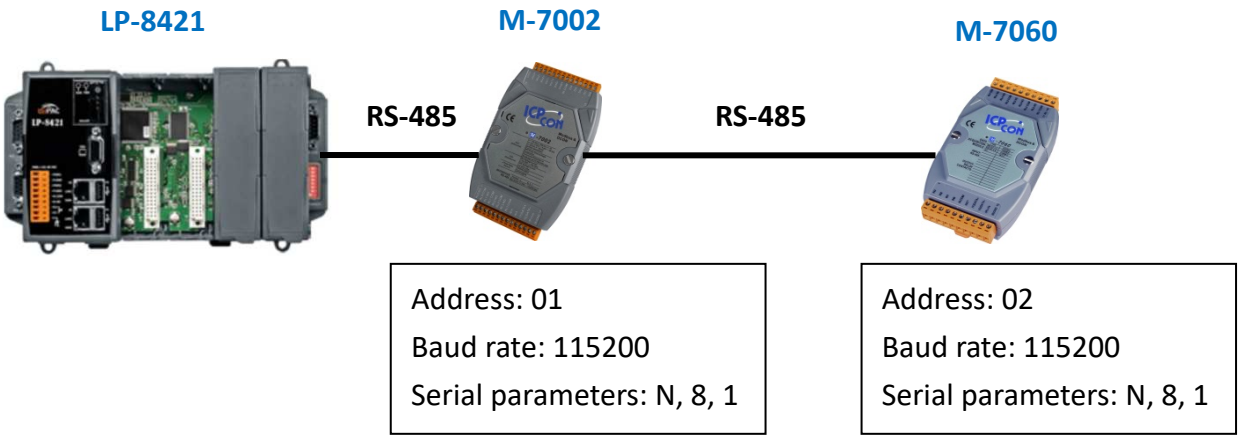
Applies to:			No. L2-002
Platform	Software operating system	OS version	Classification
All LinPACs	All version	All version	Installation & Configuration

Modbus is a data communications protocol developed by Modicon Inc. in 1979 for use with its controllers. This protocol has become a de facto standard and is now commonly using for connecting industrial electronic devices. The development and update of Modbus protocol are now managed by the Modbus Organization, please visit https://www.modbus.org/ for detailed information.

M-7000 series modules support the Modbus RTU, it uses the RS-485 as its wiring type. The communication baud rate ranges from 1200 bps to 115200 bps. The parity, data bits and stop bits are fixed as no parity, 8 data bits and 1 stop bit.

This article illustrates how to use LinPAC to access M-7000 series modules with LinPAC SDK and libmodbus respectively, using M-7002 and M-7060 connected to the COM2 port (/dev/ttyS0) on the LP-8x2x for demonstration.

Note: DCON Utility also supports the M-7000 series modules, please tick the "Modbus RTU" checkbox when setting the communication configuration, more information can be found at: http://www.icpdas.com/en/product/guide+Software+Utility_Driver+DCON_Utility_Pro



Copyright © 2020 ICP DAS Co., Ltd. All Rights Reserved.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

I. LinPAC SDK

User can use the programs built-in LinPAC to make use of the M-7000 modules, these are also included in the LinPAC SDK. Table 1 shows the basic function of Modbus protocol, which is necessary when sending a Modbus request:

Function code	Description
01 (0x01)	Read coils
02 (0x02)	Read discrete inputs
03 (0x03)	Read holding registers
04 (0x04)	Read input registers
05 (0x05)	Write single coil
06 (0x06)	Write single register
15 (0x0F)	Write multiple coils
16 (0x10)	Write multiple register

Table 1

User can refer to the manual of the modules to find the functions being supported.

For function code 01 to 04, use ***getmodbus*** command, as illustrated in Figure 1.

Example 1: Read AI0 value by FC04 with M-7002 module which net id is 01 and baud rate is 115200 from the COM2 comport.

Command: # getmodbus comport baudrate netid command addr count timeout

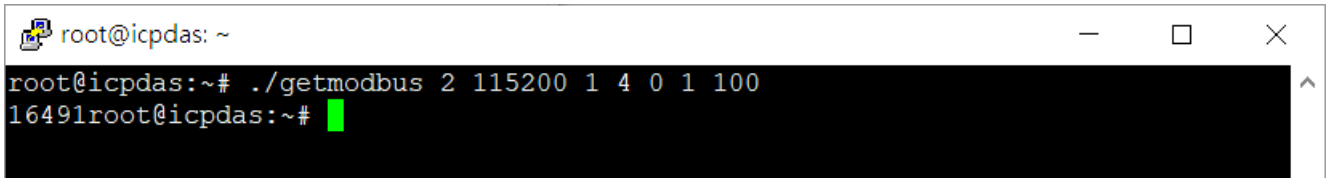


Figure 1

For function code 05, 06, 15 and 16, use ***setmodbus*** command, as illustrated in Figure 2.

Example 2: Set DO0~DO3 value to 10 (1010_2) by FC15 with M-7060 module which net id is 02 and baud rate is 115200 from the COM2 comport.

Command: # setmodbus comport baudrate netid command addr count value timeout

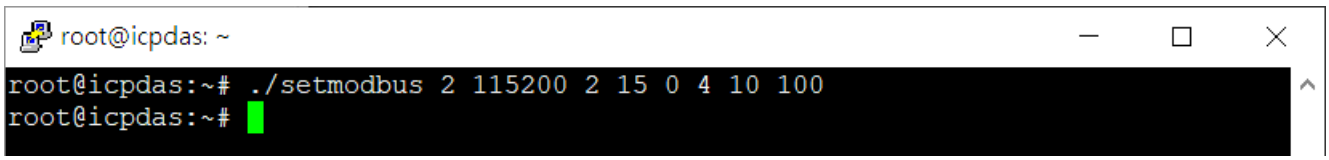


Figure 2

Copyright © 2020 ICP DAS Co., Ltd. All Rights Reserved.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

The two programs at page 2 use the function ***modbusRequest*** to send Modbus RTU request:

int modbusRequest (char cPort, char cNetID, char cFunction, WORD wAddr,
WORD wCount, char szBuf, WORD wBufLen, WORD wTimeout, WORD *wT)

Parameters:

cPort:	[Input] The number of the COM port.
cNetID:	[Input] The NetID for the device. The default slave address for M-7000 modules is '1'.
cFunction:	[Input] The Modbus RTU protocol function code.
wAddr:	[Input] The channel address, mapping in decimal format.
wCount:	[Input] The number of channels.
szBuf[ ]:	[Input/Output] Used to set or read back the value from a function code.
wBufLen:	[Input] The length of the szBuf[ ].
wTimeout:	[Input] The Timeout setting. The normal value is 100 milliseconds.
*wT:	[Output] The total duration of the send/receive interval. Unit=1 ms.

Return Values:

0:	The function was successfully processed.
Other:	The processing failed.

To edit the configuration of M-7000 series modules by LinPAC, please use the function code ***70(0x46) Read/Write Module Settings***, Table 2 lists some common sub-function code of M-7000 series modules, user can refer to manual of specific modules to find more supported sub-function code:

Sub-function code	Description
00 (0x00)	Reads the name of the module
04 (0x04)	Sets the address
05 (0x05)	Reads the communication settings
06 (0x06)	Sets the communication settings
32 (0x20)	Reads the firmware version information

Table 2

Note: Please use the sub-function code value to replace the wAddr value when calling **modbusRequest** with the function code ***70(0x46) Read/Write Module Settings***, and set wCount value to be zero:

int modbusRequest (char cPort, char cNetID, 70, WORD wSubFunction, 0,
char szBuf, WORD wBufLen, WORD wTimeout, WORD *wT)

Copyright © 2020 ICP DAS Co., Ltd. All Rights Reserved.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

II. libmodbus

libmodbus is a free software library to send/receive data with a device which respects the Modbus protocol. This library is written in C and designed to run on Linux, Mac OS X, FreeBSD and QNX and Windows. Please visit the official website www.libmodbus.org to get more information.

Simple example

```
modbus_t *ctx;
int rc;
uint8_t *tab_ bits;

// COM2 = /dev/ttyS0, Baud rate = 115200, N = no parity, data bits = 8 bits, stop bits = 1 bit
ctx = modbus_new_rtu("/dev/ttyS0", 115200, 'N', 8, 1);
modbus_set_slave(ctx, SERVER_ID);    // Set net ID of the module which user wants to connect with.
if (modbus_connect(ctx) == -1) {
        fprintf(stderr, "Connection failed: %s\n",
        modbus_free(ctx);
        return -1;
}

// Send request here, the functions provided list in Table 3.
rc = modbus_write_bit(ctx, 0, 1);    // function 05
if (rc != 1) {
        printf("ERROR modbus_write_bit (%d)\n", rc);
}else{
        rc = modbus_read_bits(ctx, 0, 1, tab_ bits);    // function 01
        if (rc != 1)
             printf("ERROR modbus_write_bit (%d)\n", rc);
}
modbus_close(ctx);
modbus_free(ctx);
```

Function code	Function provided
01 (0x01)	int **modbus_read_bits**(modbus_t *ctx, int addr, int nb, uint8_t *dest)
02 (0x02)	int **modbus_read_input_bits**(modbus_t *ctx, int addr, int nb, uint8_t *dest)
03 (0x03)	int **modbus_read_registers**(modbus_t *ctx, int addr, int nb, uint16_t *dest)
04 (0x04)	int **modbus_read_input_registers**(modbus_t *ctx, int addr, int nb, uint16_t *dest)
05 (0x05)	int **modbus_write_bit**(modbus_t *ctx, int addr, int status)
06 (0x06)	int **modbus_write_register**(modbus_t *ctx, int addr, const uint16_t value)
15 (0x0F)	int **modbus_write_bits**(modbus_t *ctx, int addr, int nb, const uint8_t *src)
16 (0x10)	int **modbus_write_registers**(modbus_t *ctx, int addr, int nb, const uint16_t *src)
17 (0x11)	int **modbus_report_slave_id**(modbus_t *ctx, int max_dest, uint8_t *dest)

Table 3

Copyright © 2020 ICP DAS Co., Ltd. All Rights Reserved.

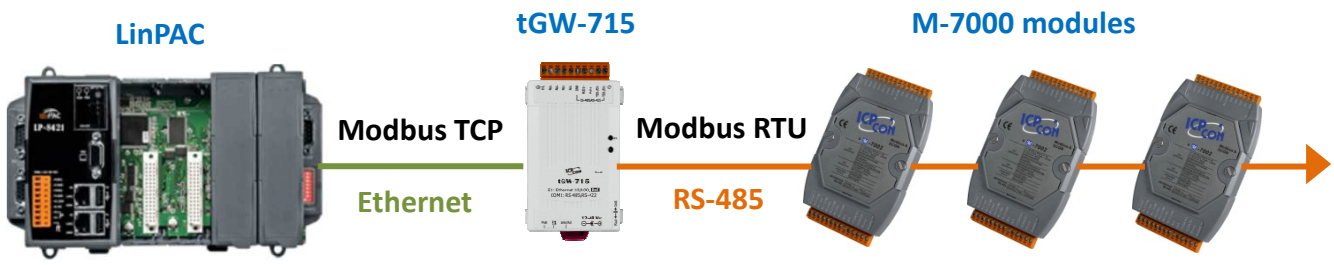
Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

libmodbus provides some tests in ***tests*** directory, user can run them to test or edit them to fit the needs. Figure 3 shows the result of running ***random-test-client.c*** program (with some changes to print more information) from *tests* directory. The module used is M-7060, which only has 4 DI and 4DO.

```
root@icpdas: ~
root@icpdas:~# ./random-test-client
<DO test>
modbus_write_bit test:
success
modbus_read_bits single test:
success
modbus_write_bits test:
success
modbus_read_bits test:
success
```

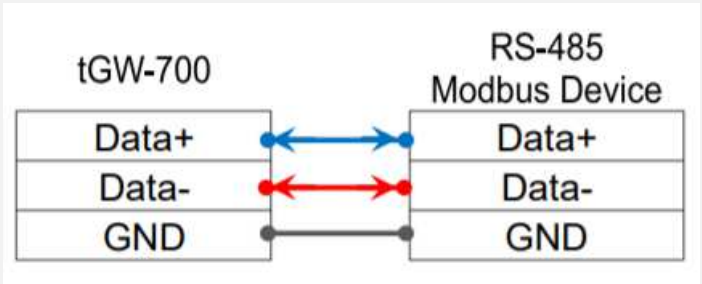
Figure 3

Note: For users that prefer to use Modbus TCP, please use a **Modbus TCP to RTU gateway** to connect between LinPAC and M-7000 series modules.



Modbus TCP to Modbus RTU gateway

The wiring between Modbus TCP to RTU gateway and M-7000 modules should be like:



Use ***modbus_t* modbus_new_tcp(const char *ip, int port)*** to replace the *Modbus_new_rtu()* function, the port should be set as 502.

Copyright © 2020 ICP DAS Co., Ltd. All Rights Reserved.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000