Modbus

Development kit for Linux

V1.0.1 June 2024

Edited by Cindy Huang

Warranty

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

Warning

ICP DAS Inc. assume no liability for any damage consequent to the use of this product. ICP DAS Inc. reserves the right to change this manual at any time without notice. The information furnished by ICP DAS Inc. is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS Co., Ltd. for its use, nor for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

Names are used for identification purposes only and maybe registered trademarks of their respective companies.

Contact US

If you have any problem, please feel free to contact us.

You can count on us for quick response.

Email: service@icpdas.com

Contents

1. Getting Started

This chapter provides a guided tour that describes the steps needed to know, download, install and configure the basic procedures for the user working with the LinPAC SDK for the first time.

1.1. Introduction the Linux SDK

This section will discuss some of the techniques that are adopted in the LinPAC SDK, including detailed explanations that describe how to easily use the LinPAC SDK.

AM335x SDK for example, the LinPAC SDK is based on Cygwin and is also a Linux-like environment for Microsoft Windows systems, and provides a powerful GCC cross-compiler and an IDE (Integrated Development Environment) that enables LinPAC applications to be quickly developed. Therefore, once an application has been created, the LinPAC SDK can be used to compile it into an executable file that can be run on the LinPAC embedded controller.

There are two choices available to you.

PAC type / Step	LP-8x41/51xx LP-2241M/5231/8x2x/9x2x	LP-2841M/LX-8000/LX9000
0.	Download SDK on Windows or Linux PC	Download SDK on LinPAC
1.	Find demo ‘helloworld.c’ in SDK	Find demo ‘helloworld.c’ in SDK
2.	Compile the demo on Windows or Linux PC using SDK	Compile the demo on LinPAC directly
3.	Upload and execute the demo on LinPAC	Execute the application on LinPAC

1.2. Download the LinPAC SDK

The LinPAC series is the Linux-based PAC and support DCON and Modbus protocols, user can install the LinPAC SDK from the FTP site of ICP DAS. The following table lists the file path of the Modbus tool, please visit the website for more information about LinPAC SDK.

LinPAC		Download Path
PXA270	LP-8x41	https://www.icpdas.com/en/download/show.php?num=982&model=LP-8441-EN
iMX8MM	LP-2841	https://www.icpdas.com/en/download/show.php?num=8723&model=LP-2841M
AM335x Series	LP-224x LP-52xx	https://www.icpdas.com/en/download/show.php?num=1195&model=LP-5231M
	LP-8x2x LP-9x2x	https://www.icpdas.com/en/download/show.php?num=915&model=LP-9821
X86/E38xx Series	LX-Series	https://www.icpdas.com/en/download/show.php?num=904&model=LX-9371

Note:

1) There are four independent LinPAC SDKs above, and different LinPAC cannot share both source files, library file and compiled files, user should be download the respective LinPAC SDK versions from the target manager and use them.
2) We recommend user to change user ID to become **root** by '**sudo**' or '**su**' command.
3) Linux 64-bit operating system lacks 32-bit support libraries. If your Linux PC is 64-bit OS, you must install 32-bit libraries on your system before you run the 32-bit version of the LinPAC SDK (Linux version).

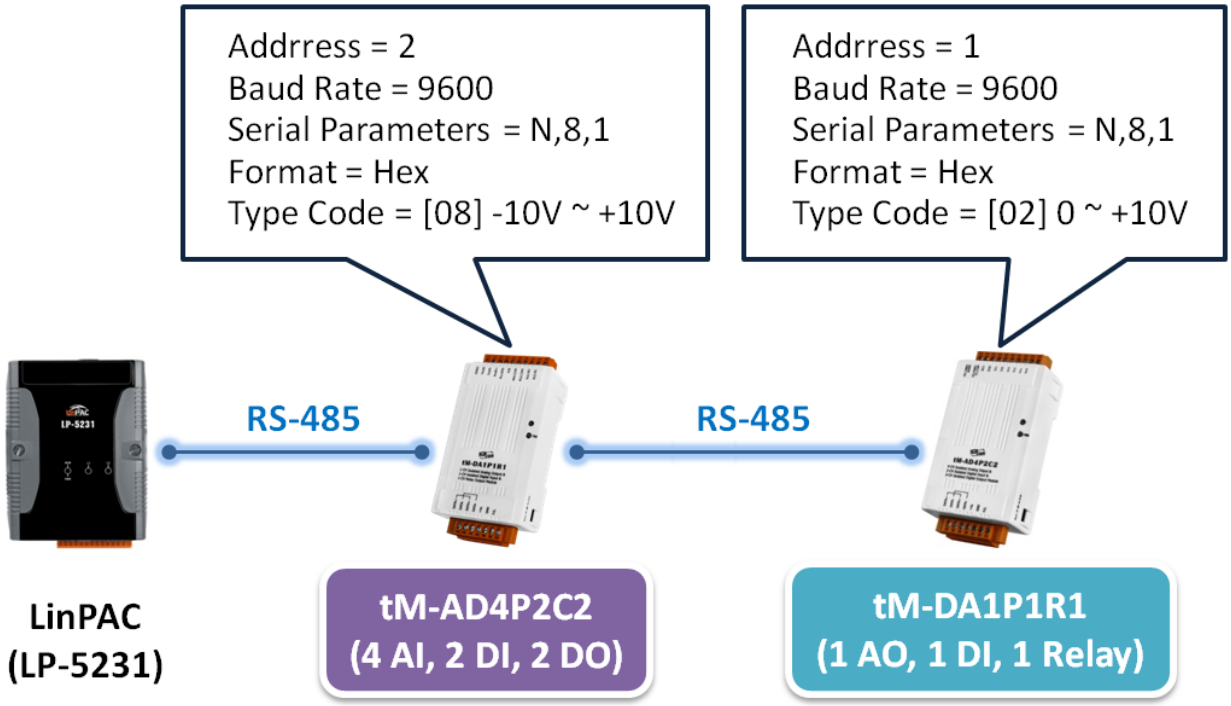
OS platform	File path of AM335x SDK
Extract the .exe file into to the C:\ driver.	
Windows	C: \cygwin\LinPAC_AM335x_SDK\examples**xvboard** or C: \cygwin\LinPAC_AM335x_SDK\examples**modbus**
Extract the .bz2 file into to the root (/) directory.	
Linux	root@LinuxPC-ICPDAS: /icpdas/linpac_am335x_sdk/i8k/examples/**xvboard/** or root@LinuxPC-ICPDAS: /icpdas/linpac_am335x_sdk/i8k/examples/**modbus/**

For more information related to **modbusRequest()** API function, please refer to:
https://www.icpdas.com/web/product/download/pac/linux/lp-5000/document/manual/xv-board_linux_api_reference_manual_en.pdf

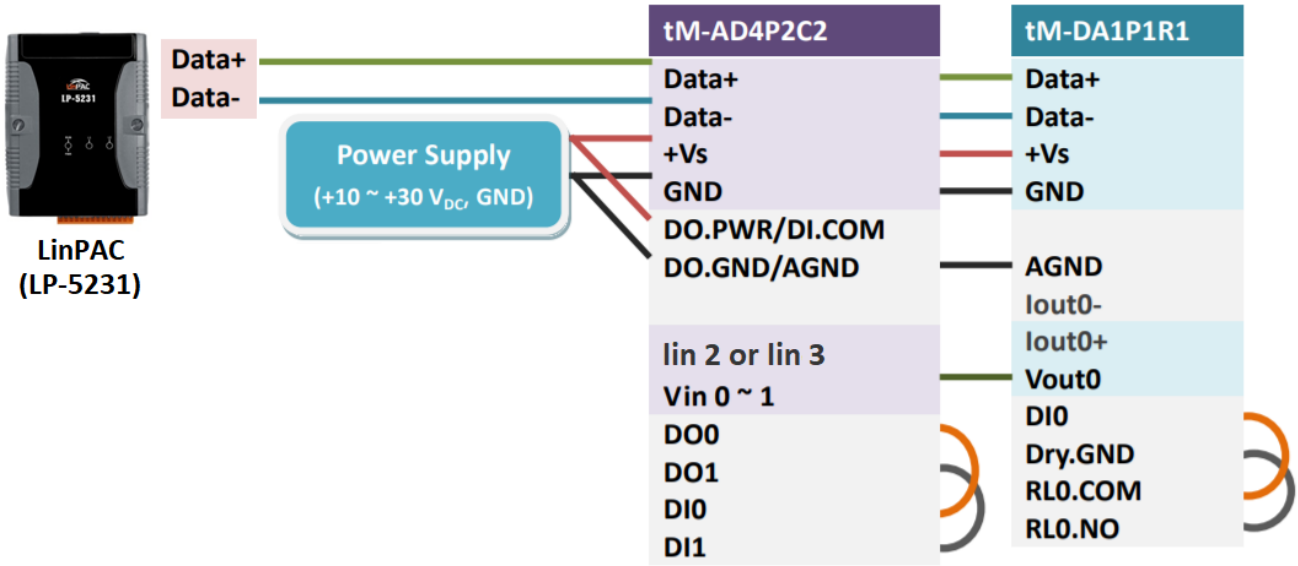
2. Modbus Development Tool

ICP DAS LinPAC family is embedded with a flexible and open-source Linux system which can be used to control tM modules via DCON or Modbus protocols. This section illustrates three kinds of Modbus software tools for users to develop various applications.

In this case, using the PAC (LP-5231) and two modules (tM-DA1P1R1 and tM-AD4P2C2) for testing. The parameters for the two modules will be set as follows:



<u>Hardware Wiring for this example</u>:



2.1 Applications of C Language on LinPAC

Step 1: Use the command to **query** the Net-ID of the tM-AD4P2C2 module.

Command:

getModbus <comport> <baudrate> <NetID> <command> <address> <count> <timeout(ms)>

```
#   getmodbus   2   9600   1   4   484   1   100
```

Step 2: Use the command to **modify** the Net-ID of the tM-AD4P2C2 module.

Command:

setModbus <comport> <baudrate> <NetID> <command> <address> <count> <value> <timeout(ms)>

```
#   setmodbus   2   9600   1   16   484   1   2   100      // Set the NetID as 2
```

Note:

1. Refer to Appendix C - Modbus Register Mapping (Base 1) to set the address of the device. https://www.icpdas.com/web/product/download/io_and_unit/rs-485/tm/document/manual/tM_series_en.pdf#page=136&zoom=100,53,57

2. The base address for tM series module is **0** (i.e., Base 0). For example, the Modbus registers 40485 is used to read/write module address (i.e., Net-ID). In this case, you should use the address 484 to get or set the Net-ID.

Here is the result of running:

```
root@LP-5231:~# getmodbus 2 9600 1 4 484 1 100
1
root@LP-5231:~# setmodbus 2 9600 1 16 484 1 2 100
root@LP-5231:~# getmodbus 2 9600 2 4 484 1 100
2
```

TEST THE AI/AO CHANNEL

Wiring the AO channel of tM-DA1P1R1 to the AI channel of tM-AD4P2C2, and the **setmodbus.c** and **getmodbus.c** programs can be used to test AI/AO. Follow the steps below:

Step 1: Use the command to **set** the AO value of the tM-DA1P1R1 module.

```
#    setmodbus 2 9600 1 16 32 1 65535 100        //Output 10 V
```

Step 2: Use the command to **read** the AO value of the tM-DA1P1R1 module.

```
#    getmodbus 2 9600 1 3 32 1 100
```

Step 3: Use the command to **read** the AI value of the tM-AD4P2C2 module.

```
#    getmodbus 2 9600 2 4 0 1 100
```

Here is the result of running:

```
root@LP-5231:~# setmodbus 2 9600 1 16 32 1 65535 100
root@LP-5231:~# getmodbus 2 9600 1 3 32 1 100
65535
root@LP-5231:~# getmodbus 2 9600 2 4 0 1 100
32767
```

TEST THE DI/DO CHANNEL

Wiring the DO channel of tM-DA1P1R1 to its DI channel, follow the steps below:

Step 1: Use the command to **set** the DO status of the tM-DA1P1R1 module.

```
#   setmodbus 2 9600 1 15 0 1 1 100      //Set the status of DO channel to "ON"
```

Step 2: Use the command to **read** the DO status of the tM-DA1P1R1 module.

```
#   getmodbus 2 9600 1 1 0 1 100
```

Step 3: Use the command to **read** the DI status of the tM-DA1P1R1 module.

```
#   getmodbus 2 9600 1 2 32 1 100
```

Here is the result of running:

```
root@LP-5231:~# setmodbus 2 9600 1 15 0 1 1 100
wCount=1 iCount=8 iIndex=0
root@LP-5231:~# getmodbus 2 9600 1 1 0 1 100
1
root@LP-5231:~# getmodbus 2 9600 1 2 32 1 100
1
```

Wiring the DO channel of tM-AD4P2C2 to its DI channel, follow the steps below:

Step 1: Use the command to **set** the DO status of the tM-AD4P2C2 module.

```
#   setmodbus 2 9600 2 15 0 1 1 100      //Set the status of DO channel to "ON"
```

Step 2: Use the command to **read** the DO status of the tM-AD4P2C2 module.

```
#   getmodbus 2 9600 2 1 0 1 100
```

Step 3: Use the command to **read** the DI status of the tM-AD4P2C2 module.

```
#   getmodbus 2 9600 2 2 32 1 100
```

Here is the result of running:

```
root@LP-5231:~# setmodbus 2 9600 2 15 0 1 1 100
wCount=1 iCount=8 iIndex=0
root@LP-5231:~# getmodbus 2 9600 2 1 0 1 100
1
root@LP-5231:~# getmodbus 2 9600 2 2 32 1 100
1
```

2.2 Applications of Python Language on LinPAC

The LinPAC series supports the Python programming language. The user can find the Modbus tool for testing tM series module from the official website of Python. In this example, the LP-5231 module is connected to the tM-DA1P1R1 module and the modbus-tk tool is used for accessing Modbus registers.

For more information about the modbus-tk tool, visit https://github.com/ljean/modbus-tk.
Follow these steps to install the software and test the module:

Step 1: Use the command to check if the version of Python is 2.5 or later.

```
#   python --version
```

Here is the result of running:

```
root@LP-5231:~# python --version
Python 2.7.3
root@LP-5231:~#
```

Step 2: Use the command to install pyserial module.

```
#   pip install pyserial
```

Step 3: Use the command to download the modbus-tk package.

```
#   wget https: //github.com/ljean/modbus-tk/archive/master.zip
```

Step 4: Use the command to unzip the modbus-tk package.

```
#   unzip master.zip
```

Step 5: Use these commands to install the modbus-tk tool.

```
#   cd modbus-tk-master
#   python setup.py build
#   python setup.py install
```

Step 6: Use the command to check if "pyserial" and "modbus-tk" have been installed successfully.

```
#   pip list
```

Here is the result of running:

```
root@LP-5231:~/modbus-tk-master# pip list
Package    Version
---------- ------------
distribute 0.6.24dev-r0
modbus-tk  0.5.8
pip        18.1
pyserial   3.4
setuptools 0.6rc11
```

Step 7: Find the "rtumaster_example.py" demo program provided by modbus-tk.

```
root@LP-5231:~# cd modbus-tk-master/examples/
root@LP-5231:~/modbus-tk-master/examples# ls
modbus_system_monitor.py  rtumaster_example.py  tcpmaster_example.py
mysimu.py                 rtuslave_example.py   tcpslave_example.py
root@LP-5231:~/modbus-tk-master/examples#
```

Step 8: Modify the parameters of the rtumaster_example.py demo program.

```
import modbus_tk
import modbus_tk.defines as cst
from modbus_tk import modbus_rtu
#PORT = 1
PORT = '/dev/ttyO2'
def main():
    """main"""
    logger = modbus_tk.utils.create_logger("console")
    try:
        #Connect to the slave
        master = modbus_rtu.RtuMaster(
            serial.Serial(port=PORT, baudrate=9600, bytesize=8, parity='N', stopbits=1, xonxoff=0)
        )
        master.set_timeout(5.0)
        master.set_verbose(True)
        logger.info("connected")
        logger.info(master.execute(1, cst.READ_HOLDING_REGISTERS, 32, 1))
        #
        #
        #
        #
```

⇨ **The communication port**

The communication parameters

The Modbus command

master.execute(NetID, function code, address, count)

Note: The base address for tM series module is '0' (Base 0).

Step 9: Execute the demo program to read the AO value of the tM-DA1P1R1 module. The results will be displayed as illustrated in the figure below.

```
root@LP-5231:~/modbus-tk-master/examples# python rtumaster_example.py
2018-12-10 17:47:25,575  INFO    modbus_rtu.__init__      MainThread    RtuMaster /dev/ttyO2 is opened
2018-12-10 17:47:25,578  INFO    rtumaster_example.main  MainThread    connected
2018-12-10 17:47:25,580  DEBUG   modbus.execute  MainThread     -> 1-3-0-32-0-1-133-192
2018-12-10 17:47:25,606  DEBUG   modbus.execute  MainThread     <- 1-3-2-255-255-185-244
2018-12-10 17:47:25,607  INFO    rtumaster_example.main  MainThread     (65535,)
```

2.3 Applications of Perl Language on LinPAC

The LinPAC series supports the Perl programming language. For testing tM series module, users can find the Modbus tool from the official website of Perl. In this example, the LP-5231 module is connected to the tM-DA1P1R1 module and the Device-Modbus-RTU tool is used for accessing Modbus registers. Follow the steps to install the software and test the module:

Step 1: Download and unzip the Device-Modbus-RTU package (Device-Modbus-RTU-0.022.tar.gz) from the website https: //metacpan.org/release/Device-Modbus-RTU

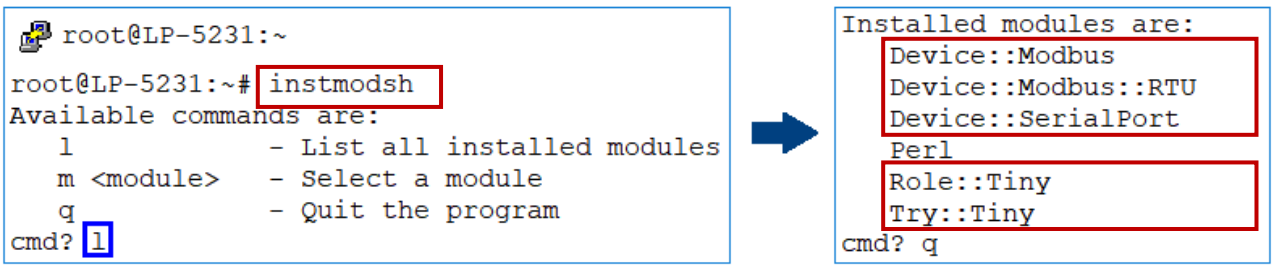
Step 2: Use the command to install the dependent module for Device-Modbus-RTU.

```
#   sudo cpan Role: : Tiny Try: : Tiny Device: : SerialPort Device: : Modbus
```

Step 3: Use the command to install the Device-Modbus-RTU tool.

```
#   cd Device-Modbus-RTU-0.022
#   perl Makefile.PL
#   make
#   make test
#   make install
```

Step 4: Use the 'instmodsh' command to check if the Perl module has been installed successfully.



Step 5: Find the "**write_new_addr.pl**" and "**simple_client_rtu.pl**" demo programs provided by Device-Modbus-RTU.

```
root@LP-5231:~# cd Device-Modbus-RTU-master/examples/
root@LP-5231:~/Device-Modbus-RTU-master/examples# ls
arduino_client.ino  server_rtu.pl  simple_client_rtu.pl  write_new_addr.pl
```

Step 6: Modify the parameters of demo programs.

❒ Modify the COM port setting in the “write_new_addr.pl” and “simple_client_rtu.pl” scripts.

```
my $client = Device: : Modbus: : RTU: : Client->new(
    port        => '/dev/ttyO2',        // The number of COM port
    baudrate    => 9600,                // Bits per second
    parity   => 'none',                 // Parity check
);
```

❒ Modify the Modbus command setting in the "write_new_addr.pl" script to set the AO value.

```
my $req = $client->write_single_register(       // Modbus function code
    unit        => 1,                   // The NetID of Slave device
    address     => 32,                  // The channel address
    value    => 65535                   // Set the AO vale
);
```

❒ Modify the Modbus command setting in the “simple_client_rtu.pl” script to read the AO value.

```
my $req = $client->read_holding_registers(          // Modbus function code
    unit        => 1,                   // The NetID of Slave device
    address     => 32,                  // The channel address
    quantity    => 1,                   // The number of channels to read
);
```

The "write_new_addr.pl" demo program:

```
#! /usr/bin/env perl
use Device::Modbus;
use Device::Modbus::RTU::Client;
use Data::Dumper;
use strict;
use warnings;
use v5.10;
my $client = Device::Modbus::RTU::Client->new(
    port     => '/dev/ttyO2',
    baudrate => 9600,
    parity   => 'none',
);
my $req = $client->write_single_register(
    unit     => 1,
    address  => 32,
    value    => 65535
);
say "->" . Dumper $req;
$client->send_request($req);
my $resp = $client->receive_response;
say "<-" . Dumper $resp;
```

➔ **The COM port setting**

➔ **The Modbus command**

Note: The base address for tM series module is ‘0’ (Base 0).

Step 7: Execute the demo program to control tM series module.

(1) The results of executing the "write_new_addr.pl" program.

```
root@LP-5231:~/Device-Modbus-RTU-master/examples# perl write_new_addr.pl
->$VAR1 = bless( {
                   'unit' => 1,
                   'function' => 'Write Single Register',
                   'value' => 65535,
                   'address' => 32,
                   'code' => 6
                 }, 'Device::Modbus::Request' );
<-$VAR1 = bless( {
                   'unit' => 1,
                   'crc' => 45193,
                   'message' => bless( {
                                        'function' => 'Write Single Register',
                                        'value' => 65535,
                                        'address' => 32,
                                        'code' => 6
                                      }, 'Device::Modbus::Response' )
                 }, 'Device::Modbus::RTU::ADU' );
```

(2) The results of executing the “simple_client_rtu.pl” program.

```
root@LP-5231:~/Device-Modbus-RTU-master/examples# perl simple_client_rtu.pl
->$VAR1 = bless( {
                   'unit' => 1,
                   'function' => 'Read Holding Registers',
                   'quantity' => 1,
                   'address' => 32,
                   'code' => 3
                 }, 'Device::Modbus::Request' );
<-$VAR1 = bless( {
                   'unit' => 1,
                   'crc' => 62649,
                   'message' => bless( {
                                        'bytes' => 2,
                                        'function' => 'Read Holding Registers',
                                        'values' => [
                                                      65535
                                                    ],
                                        'code' => 3
                                      }, 'Device::Modbus::Response' )
                 }, 'Device::Modbus::RTU::ADU' );
```

A. How to compile application including libmodbus library on Windows Platform

DOWNLOAD AND INSTALL

To compile Libmodbus under Windows, user need to install MinGW and MSYS then select the common packages (gcc, automake, libtool, etc) as below:

- **Libmodbus**: https://github.com/stephane/libmodbus/releases
- **MinGW for Windows Platform**: https://sourceforge.net/projects/mingw/
- **MSYS for Windows Platform** : https://www.msys2.org/

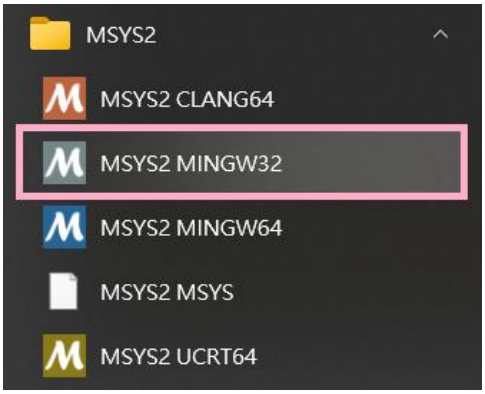
ENVIRONMENT VARIABLE CONFIGURATION

The PATH variable defines the search path for running commands. Therefore, user need to modify the **C:\msys64\etc\profile** file, add the cross compile folder to environment variable "**PATH**" under the "MSYS" variable.

Here is an example of **LP-8x21**:

```
PATH=$PATH:/'c/cygwin/LinPAC_AM335x_SDK/Linaro_GCC_4.7/bin:/c/Cygwin/LinPAC_AM335x_SDK/Linaro_GCC_4.7/arm-linux-gnueabihf/bin:/c/cygwin/LinPAC_AM335x_SDK/Linaro_GCC_4.7/arm-linux-gnueabihf/libc/usr/lib/opkg/alternatives:/c/cygwin/LinPAC_AM335x_SDK/Linaro_GCC_4.7/arm-linux-gnueabihf/libc/usr/lib/pkgconfig'
```

Open mingw32.exe shell launcher (Click the '**Start**' menu → '**MSYS2**' → '**MSYS2 MINGW32**')



Or enter to C:\msys64\ directory, and click the mingw32.exe

Using the **exprot** command to check the environment variable of PATH.

```
~
Cindy@RD1-Freda-Chen MINGW32 ~
$ export |grep $PATH
declare -x PATH="/mingw32/bin:/usr/local/bin:/usr/bin:/bin:/c/Windows/System32:/c/Windows:/c/W
indows/System32/Wbem:/c/Windows/System32/WindowsPowerShell/v1.0/:/usr/bin/site_perl:/usr/bin/v
endor_perl:/usr/bin/core_perl:/c/cygwin/LinPAC_AM335x_SDK/Linaro_GCC_4.7/bin:/c/Cygwin/LinPAC_
AM335x_SDK/Linaro_GCC_4.7/arm-linux-gnueabihf/bin:/c/cygwin/LinPAC_AM335x_SDK/Linaro_GCC_4.7/a
rm-linux-gnueabihf/libc/usr/lib/opkg/alternatives:/c/cygwin/LinPAC_AM335x_SDK/Linaro_GCC_4.7/a
rm-linux-gnueabihf/libc/usr/lib/pkgconfig"

Cindy@RD1-Freda-Chen MINGW32 ~
$
```

Update MSYS2 and Install Packages

I. Installation: pacman –Syu

II. Installation: pacman –Su

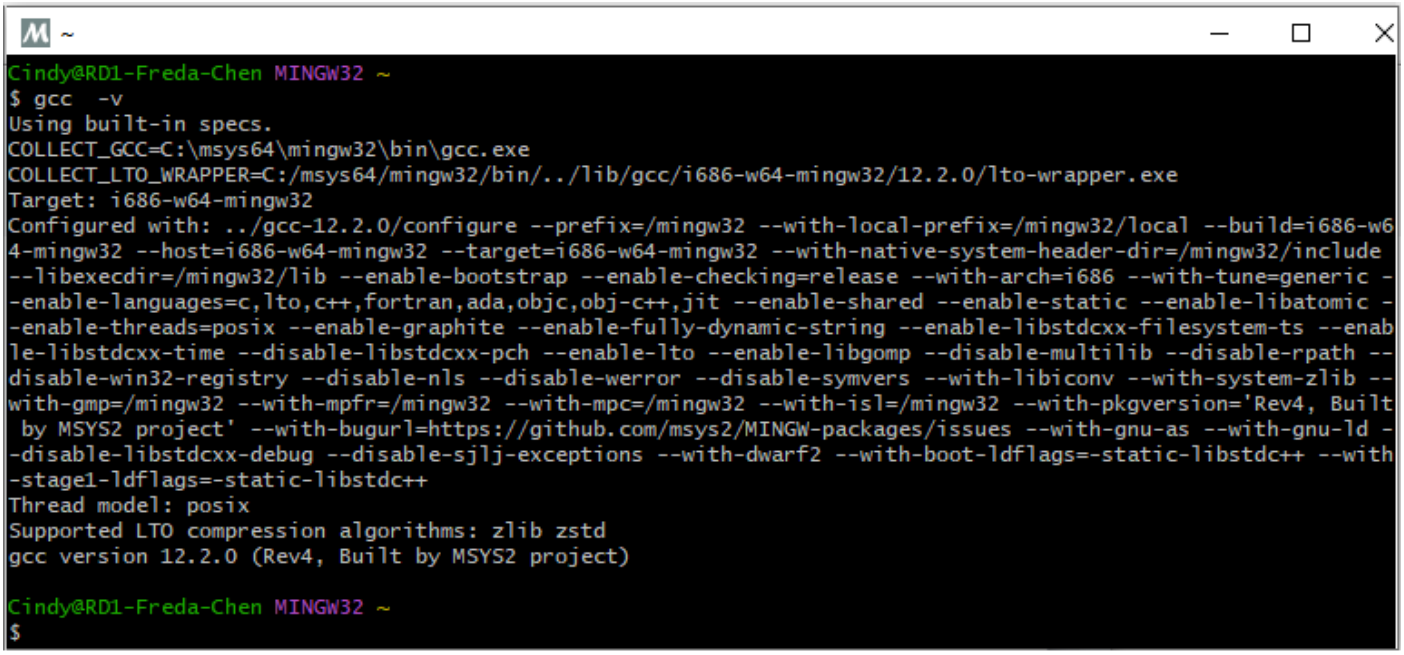
III. Installation: pacman –S autoconf-wrapper

IV. Installation: pacman -S mingw-w64-i686-toolchain

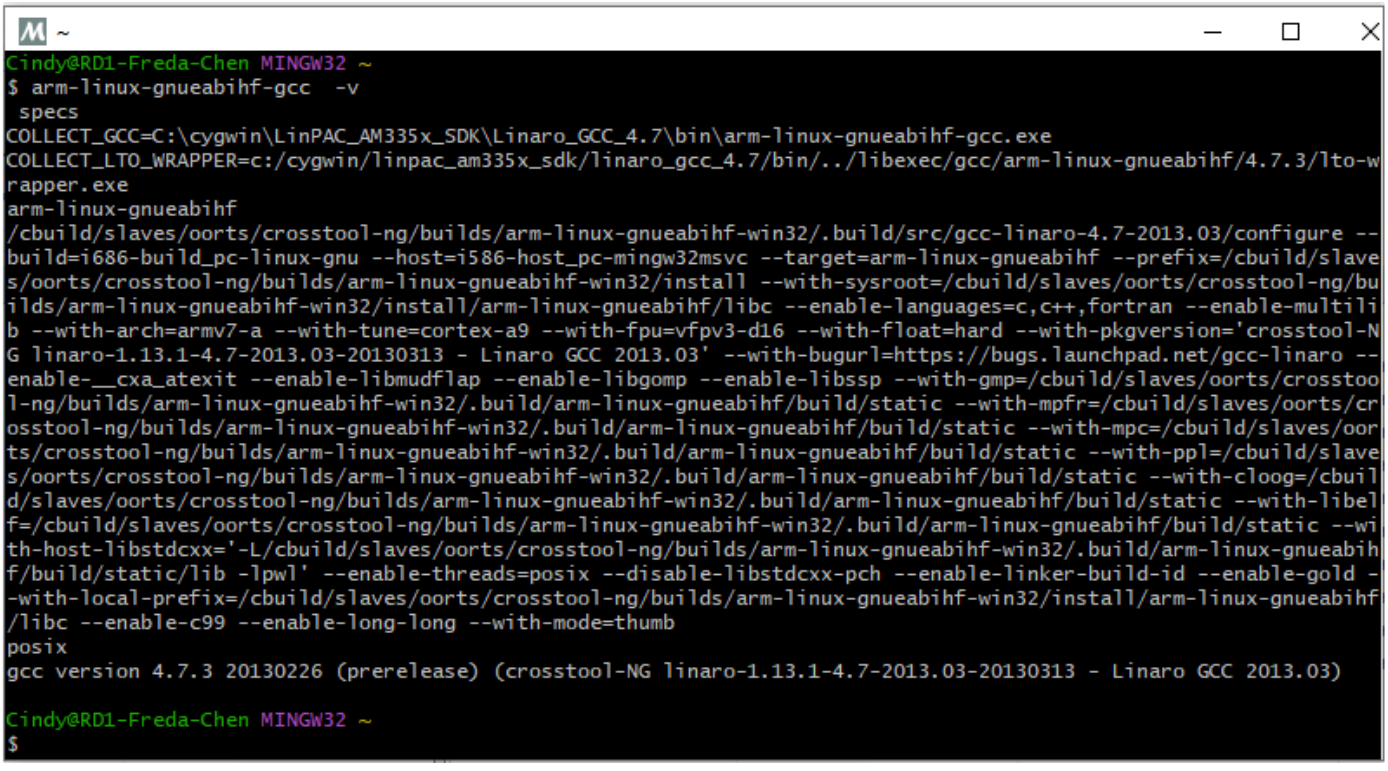
V. Installation: pacman -S autotools

```
/c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10
Cindy@RD1-Freda-Chen MINGW32 /c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10
$ pacman -S autotools
resolving dependencies...
looking for conflicting packages...

Packages (10) automake-wrapper-11-4  automake1.11-1.11.6-6  automake1.12-1.12.6-6
              automake1.13-1.13.4-7  automake1.14-1.14.1-6  automake1.15-1.15.1-4
              automake1.16-1.16.5-1  libltdl-2.4.6-14  libtool-2.4.6-14  autotools-2022.01.16-2

Total Download Size:    3.38 MiB
Total Installed Size:  10.98 MiB

:: Proceed with installation? [Y/n] y
:: Retrieving packages...
 automake1.15-1.15.1-4-any       513.4 KiB  86.5 KiB/s 00:06 [############################] 100%
 automake1.16-1.16.5-1-any       526.3 KiB  86.4 KiB/s 00:06 [############################] 100%
 automake1.12-1.12.6-6-any       503.1 KiB  75.6 KiB/s 00:07 [############################] 100%
 automake1.11-1.11.6-6-any       490.2 KiB   341 KiB/s 00:01 [############################] 100%
 libltdl-2.4.6-14-x86_64          32.0 KiB  23.5 KiB/s 00:01 [############################] 100%
 libtool-2.4.6-14-x86_64         388.1 KiB   235 KiB/s 00:02 [############################] 100%
 automake1.13-1.13.4-7-any       501.5 KiB  61.1 KiB/s 00:08 [############################] 100%
 automake1.14-1.14.1-6-any       503.1 KiB  60.5 KiB/s 00:08 [############################] 100%
 automake-wrapper-11-4-any         4.4 KiB  1210   B/s 00:04 [############################] 100%
 autotools-2022.01.16-2-any        2.5 KiB   651   B/s 00:04 [############################] 100%
 Total (10/10)                     3.4 MiB   261 KiB/s 00:13 [############################] 100%
(10/10) checking keys in keyring                           [############################] 100%
(10/10) checking package integrity                         [############################] 100%
(10/10) loading package files                              [############################] 100%
(10/10) checking for file conflicts                        [############################] 100%
(10/10) checking available disk space                      [############################] 100%
:: Processing package changes...
( 1/10) installing automake1.11                            [############################] 100%
( 2/10) installing automake1.12                            [############################] 100%
( 3/10) installing automake1.13                            [############################] 100%
( 4/10) installing automake1.14                            [############################] 100%
( 5/10) installing automake1.15                            [############################] 100%
( 6/10) installing automake1.16                            [############################] 100%
( 7/10) installing automake-wrapper                        [############################] 100%
( 8/10) installing libltdl                                 [############################] 100%
( 9/10) installing libtool                                 [############################] 100%
(10/10) installing autotools                               [############################] 100%
:: Running post-transaction hooks...
(1/1) Updating the info directory file...
```

Using the **'gcc –v'** command to check the gcc version.



Using the **'arm-linux-gnueabihf-gcc –v'** command to check the cross compiler.



MSYS2 Packages : https://packages.msys2.org/groups/mingw-w64-i686-toolchain

COMPILE LIBMODBUS LIBRARY AND EXAMPLES

Run **'sh autogen.sh'** first to generate the configure script.

```
/c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10
Cindy@RD1-Freda-Chen MINGW32 /c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10
$ sh autogen.sh
libtoolize: putting auxiliary files in AC_CONFIG_AUX_DIR, 'build-aux'.
libtoolize: linking file 'build-aux/ltmain.sh'
libtoolize: putting macros in AC_CONFIG_MACRO_DIRS, 'm4'.
libtoolize: linking file 'm4/libtool.m4'
libtoolize: linking file 'm4/ltoptions.m4'
libtoolize: linking file 'm4/ltsugar.m4'
libtoolize: linking file 'm4/ltversion.m4'
libtoolize: linking file 'm4/lt~obsolete.m4'
configure.ac:33: installing 'build-aux/compile'
configure.ac:56: installing 'build-aux/config.guess'
configure.ac:56: installing 'build-aux/config.sub'
configure.ac:32: installing 'build-aux/install-sh'
configure.ac:32: installing 'build-aux/missing'
src/Makefile.am: installing 'build-aux/depcomp'
parallel-tests: installing 'build-aux/test-driver'

------------------------------------------------------
Initialized build system. You can now run ./configure
------------------------------------------------------
```

To go to the libmodbus-3.1.8 directory, and type:

```
Cindy@RD1-Freda-Chen MINGW32 /c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.8
$ mkdir   linpac

Cindy@RD1-Freda-Chen MINGW32 /c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.8
$ sh configure   CC=arm-linux-gnueabihf-gcc   --host=arm-linux-gnueabihf   --enable-static
--prefix=$(pwd)/linpac
```

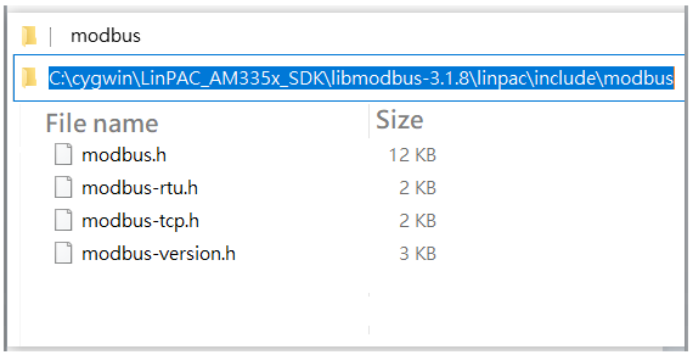
Using the **'make'** command to compile programs.

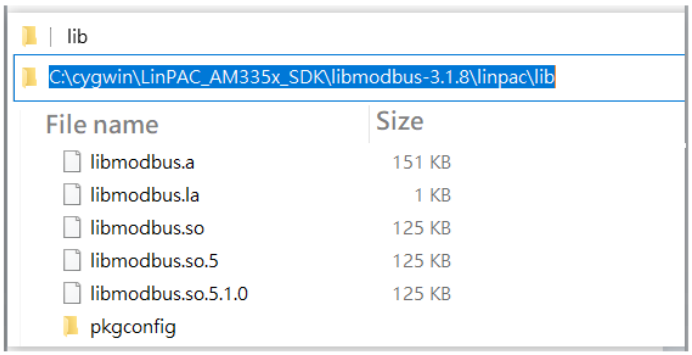
```
/c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10

Cindy@RD1-Freda-Chen MINGW32 /c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10
$ make
make --no-print-directory all-recursive
Making all in src
  CC       modbus.lo
  CC       modbus-data.lo
  CC       modbus-rtu.lo
  CC       modbus-tcp.lo
  CCLD     libmodbus.la
Making all in tests
make  all-am
  CC       bandwidth-server-one.o
  CCLD     bandwidth-server-one
  CC       bandwidth-server-many-up.o
  CCLD     bandwidth-server-many-up
  CC       bandwidth-client.o
  CCLD     bandwidth-client
  CC       random-test-server.o
  CCLD     random-test-server
  CC       random-test-client.o
  CCLD     random-test-client
  CC       unit-test-server.o
  CCLD     unit-test-server
  CC       unit-test-client.o
  CCLD     unit-test-client
  CC       version.o
  CCLD     version
```

Using the **'make install'** command to install the library and its header files.

```
/c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10
Cindy@RD1-Freda-Chen MINGW32 /c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10
$ make  install
Making install in src
 /usr/bin/mkdir -p '/c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10/linpac/lib'
 /bin/sh ../libtool   --mode=install /usr/bin/install -c   libmodbus.la '/c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10/linpac/lib'
libtool: install: /usr/bin/install -c .libs/libmodbus.so.5.1.0 /c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10/linpac/lib/libmodbus.so.5.1.0
libtool: install: (cd /c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10/linpac/lib && { cp -pR -f libmodbus.so.5.1.0 libmodbus.so.5 || { rm -f libmodbus.so.5 && cp -pR libmodbus.so.5.1.0 libmodbus.so.5; }; })
libtool: install: (cd /c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10/linpac/lib && { cp -pR -f libmodbus.so.5.1.0 libmodbus.so || { rm -f libmodbus.so && cp -pR libmodbus.so.5.1.0 libmodbus.so; }; })
libtool: install: /usr/bin/install -c .libs/libmodbus.lai /c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10/linpac/lib/libmodbus.la
libtool: install: /usr/bin/install -c .libs/libmodbus.a /c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10/linpac/lib/libmodbus.a
libtool: install: chmod 644 /c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10/linpac/lib/libmodbus.a
libtool: install: arm-linux-gnueabihf-ranlib /c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10/linpac/lib/libmodbus.a
libtool: finish: PATH="/mingw32/bin:/usr/local/bin:/usr/bin:/bin:/c/Windows/System32:/c/Windows:/c/Windows/System32/Wbem:/c/Windows/System32/WindowsPowerShell/v1.0/:/usr/bin/site_perl:/usr/bin/vendor_perl:/usr/bin/core_perl:/c/cygwin/LinPAC_AM335x_SDK/Linaro_GCC_4.7/bin:/c/Cygwin/LinPAC_AM335x_SDK/Linaro_GCC_4.7/arm-linux-gnueabihf/bin:/c/cygwin/LinPAC_AM335x_SDK/Linaro_GCC_4.7/arm-linux-gnueabihf/libc/usr/lib/opkg/alternatives:/c/cygwin/LinPAC_AM335x_SDK/Linaro_GCC_4.7/arm-linux-gnueabihf/libc/usr/lib/pkgconfig:/sbin" ldconfig -n /c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10/linpac/lib
/c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10/libtool: line 1737: ldconfig: command not found
----------------------------------------------------------------------
Libraries have been installed in:
   /c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10/linpac/lib

If you ever happen to want to link against installed libraries
in a given directory, LIBDIR, you must either use libtool, and
specify the full pathname of the library, or use the '-LLIBDIR'
flag during linking and do at least one of the following:
   - add LIBDIR to the 'LD_LIBRARY_PATH' environment variable
     during execution
   - add LIBDIR to the 'LD_RUN_PATH' environment variable
     during linking
   - use the '-Wl,-rpath -Wl,LIBDIR' linker flag
   - have your system administrator run these commands:


See any operating system documentation about shared libraries for
more information, such as the ld(1) and ld.so(8) manual pages.
----------------------------------------------------------------------
 /usr/bin/mkdir -p '/c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10/linpac/include/modbus'
 /usr/bin/install -c -m 644 modbus.h modbus-version.h modbus-rtu.h modbus-tcp.h '/c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10/linpac/include/modbus'
Making install in tests
make[2]: Nothing to be done for 'install-exec-am'.
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Nothing to be done for 'install-exec-am'.
 /usr/bin/mkdir -p '/c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10/linpac/share/doc/libmodbus'
 /usr/bin/install -c -m 644 AUTHORS NEWS README.md '/c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10/linpac/share/doc/libmodbus'
 /usr/bin/mkdir -p '/c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10/linpac/lib/pkgconfig'
 /usr/bin/install -c -m 644 libmodbus.pc '/c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.10/linpac/lib/pkgconfig'
```

Here is the result of complete installing libmodbus-3.1.8 on Windows platform.





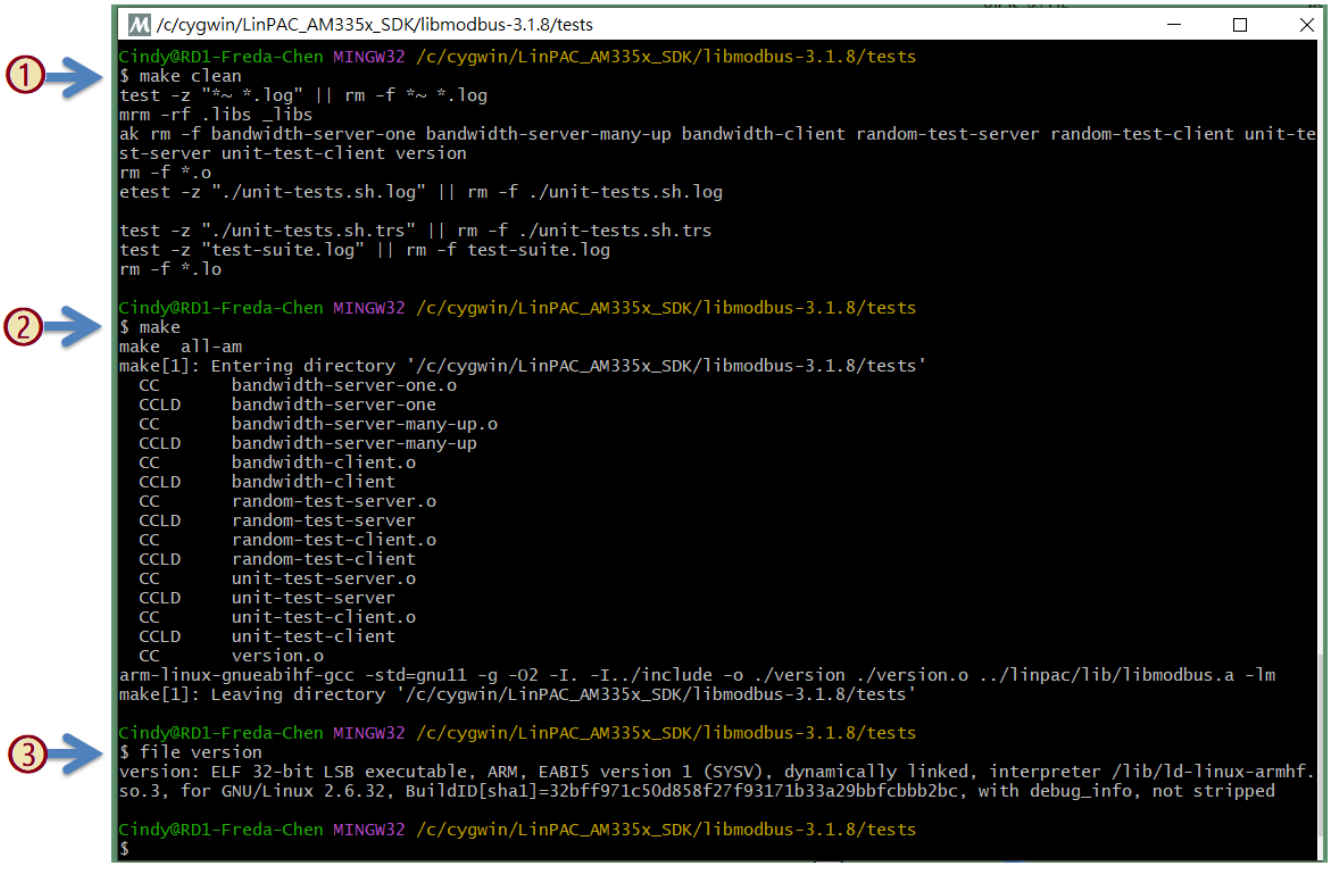
Compile modbus program with the following command manually:

```
Cindy@RD1-Freda-Chen MINGW32 /c/cygwin/LinPAC_AM335x_SDK/libmodbus-3.1.8/tests
$ arm-linux-gnueabihf-gcc -I../linpac/include/modbus -lm -o random-test-server.exe
random-test-server.c ../linpac/lib/libmodbus.a
```

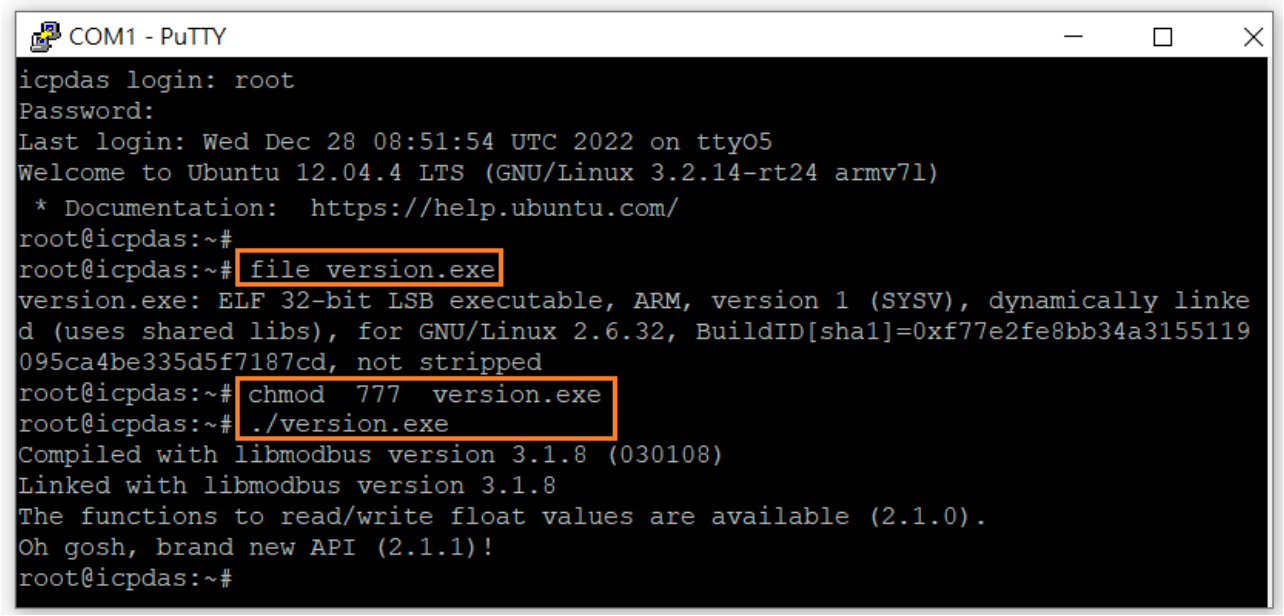
Or user can modify the **Makefile** file, add the following code:

- LDFLAGS = -lm
- CFLAGS = -g -O2 -I. -I../include
- LIBS = ../linpac/lib/libmodbus.a
- Change syntax of a makefile's contents --- 'version' for example.

```
676 #version$(EXEEXT): $(version_OBJECTS) $(version_DEPENDENCIES) $(EXTRA_version_DEPENDENCIES)
677 #   @rm -f version$(EXEEXT)
678 #   $(AM_V_CCLD)$(LINK) $(version_OBJECTS) $(version_LDADD) $(LIBS)
679
680 version: ./version.o
681     $(CC) $(CFLAGS) -o ./$@ ./version.o $(LIBS) $(LDFLAGS)
682     @rm -f ./version.o
```



UPLOAD EXECUTABLE FILE TO THE LINPAC



 E-mail: service@icpdas.com

B. Revision History

This chapter provides revision history information to this document.

The table below shows the revision history.

Revision	Date	Description
V1.0.0	June 2023	Initial issue
V1.0.1	March 2024	New: How to compile application including libmodbus library on Windows Platform?

E-mail: service@icpdas.com