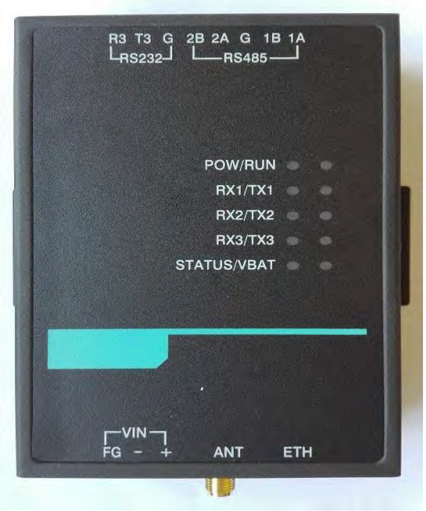
# KPM202 Communication Manager

Embedded Computer Based on RISC Architecture with 2 Serial Ports, 1 Network Ports, GPRS\3G \4G and SD Card Slots



**Product Name**: **Embedded computer**

**Category**: Gateway, communication manager, collector, concentrator

**Product description**: KPM202 is an embedded computer based on a RISC architecture five-stage pipelined chip as the main processor. The CPU is a system-level single-chip ARM926EJS core, built-in 64MB DDR2, providing encryption and decryption software, providing AES, DES/3DES encryption and decryption, up to 300MHz frequency. The system provides wired network communication and also provides wireless GPRS communication. It has the characteristics of small size, low power consumption, high efficiency, and is suitable for power collectors, HMI, industrial control, gateways and other occasions.

## 1.Hardware specifications

**Main system**

**NUC970 CPU:**

32bit ARM926EJ-S, 300MHz Clocked, 1.1MIPS / MHz, Up to 300MHz

16KB I-cache，16KB D-cache

Support MMU, support JTAG Debug

RAM: 64Mbyte DDR2 Mbyte56SDRAM

Up to 150MHz SDRAM clock

Flash：128Mbyte NAND Flash, up to 512Mbyte

Supports SLC, MLC type NAND FLASH

**Encryption:** Supports PRNG/DES/3DES/AES/SHA/HMAC encryption up to 256-bit encryption

**Software Encryption :** Built-in WDT, overflow time less than 14 seconds, support for idle wake-up and power-down wake-up

**RTC:** Real-time clock, built-in battery

**Serial interface**

**RS232:** 1-way RS232 communication port, built-in ESD protection, fully isolated protection design

**RS485:** 2-way RS485 communication port, built-in ESD protection, fully isolated protection design

**Network**

**LAN:** 1-way 10/100Mbps Adaptive Industrial Ethernet, RJ45

**Protection:** 15KV TVS protection

**GPRS**

**RF Band:** 850/900/1800/1900MHz

**Interface:** 2 SIM card interfaces (parallel connection), 1 antenna interface

**3G (optional WCDMA)**

**Technical system:** WCDMA/HSDPA/GSM/GPRS/EDGE

**RF Band:** 2100/1900/850MHz

**Peak rate:** 3.6Mbps (downlink) /384Kbps (upstream)

**Peak rate:** 100Mbps (downlink) / 50M bps (upstream)

**WIFI（WLAN）**

**Compatible standards:** 802.11a/b/g

**RF Type**: DSSS, CCK, OFDM

**Transmission rate:** 150Mbps

**Transmission distance:** 200 meters (open area without shelter)

**Dry node:** logic 0 (short to GND), logic 1 (disconnected)

**Wet node:** logic 0 (below 3VDC), logic 1 (10 ～ 30 VDC)

**Protection:** 4KV photoelectric isolation; 36 VDC overvoltage protection

**Power requirements**

**Power input:** 6-24 VDC, 12 VDC recommended

**Power Consumption:** 250 mA @ 12 VDC, 3W

**Mechanical properties**

**Shell:** metal shell

**Dimensions:** 120x75x30mm

**working environment**

**Operating temperature:**-25℃-+70℃-

**storage temperature:**-30℃-+75℃-

## 2.Software specifications

### 2.1 System Overview

Linux is a mature and stable network operating system. Embedding Linux into an embedded device has many advantages. First of all, Linux's source code is open, anyone can get and modify, use it to develop their own products. Second, Linux is customizable, and its system kernel is only about 134KB. A core program with a Chinese system and graphical user interface can also achieve less than 1MB, and is also stable. In addition, it is compatible with most Unix systems, and application development and porting are fairly easy. At the same time, due to its good portability, people have successfully run Linux on hundreds of hardware platforms.

### 2.2 Linux's main advantages as an embedded operating system

1) Can be applied to various hardware platforms. Linux has been ported to a variety of hardware platforms. For funds, time-limited research and development projects are attractive. The prototype can be developed on a standard platform and then ported to specific hardware, speeding up the software and hardware development process. Linux uses a unified framework to configure the hardware without any licensing or business partnerships. The source code is freely available. This makes using Linux as an operating system free from any copyright disputes. There is no doubt that this will save a lot of development costs. The built-in network support itself, and the current embedded system requires higher and higher network support. The debug modularity of Linux makes adding parts very easy.

2) Linux is a general-purpose operating system that is similar to Unix, is kernel-based, has full memory access control, and supports a large number of hardware (including most existing chips such as X86, Alpha, ARM, and Motorola). . All program source code is publicly available, and anyone can modify it and distribute it under the GNU General Public License. In this way, developers can customize the operating system to suit their specific needs.

3) Linux comes with well-developed development tools that Unix users are familiar with. Almost all Unix system applications have been ported to Linux. Linux also provides powerful networking features with a variety of selectable window managers (XWindows). Its powerful language compilers GCC, C++, etc. can also be easily obtained, not only mature and complete, but also easy to use.

### 2.3KPM202 system features

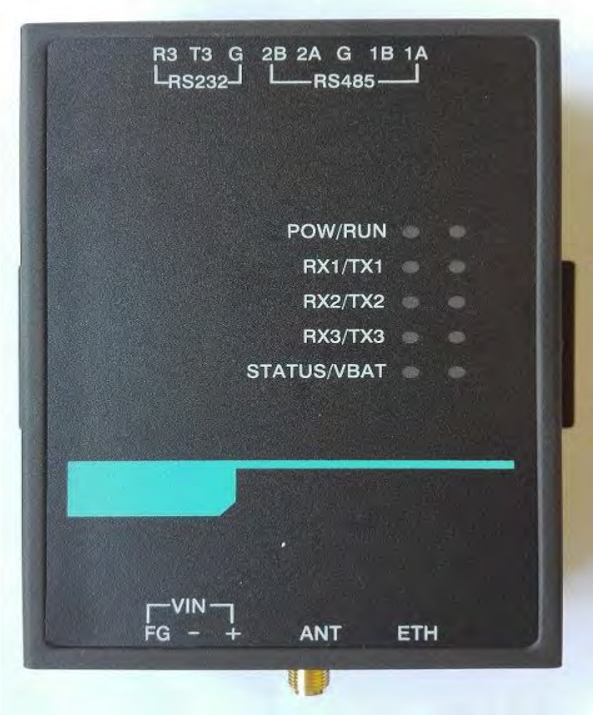
The KPM202 comes pre-installed with the nuvoton NUV970-based Linux operating system, version 3.10.101. Meet POSIX standards or UNIX-like applications. For the system-specific hardware devices, the kernel provides a simple, easy-to-use driver interface that can speed up the user's application development.

The KPM202 system software system is divided into three parts, namely Bootloader, Linux kernel and rootfs. Bootloader is an open source project that complies with the terms of the GPL. UBuot is mainly used to boot the kernel. It supports NFS mount and NAND Flash boot. The linux kernel is the bottom layer of the entire operating system, responsible for the entire hardware driver, and provides various system requirements. The core functionality; rootfs is a method and data structure for specifying files on a disk or partition, that is, a method of organizing files on a disk.

## 3.Interface definition

### 1. Power Interface (VIN)

| NO. | Identifier | Function Description |
|---|---|---|
| 1 | FG+ | Shielded ground, protected ground, can not be connected |
| 2 | _ | System power ground |
| 3 | + | System power supply, input voltage range DC6~24V, DC12V recommended |



### 2. SIM card interface (SIM CARD)

| NO. | Identifier | Function Description |
|---|---|---|
| 1 | SIM CARD | 2G, 3G 4G SIM card interface, support Mobile, Unicom Telecom card and Wifi |

### 3. Antenna Interface (ANT)

| NO. | Identifier | Function Description |
|---|---|---|
| 1 | GPRS | 2G, 3G SMA antenna interface |

### 4. Network Interface (ETH)

| NO. | Identifier | Function Description |
|---|---|---|
| 1 | E0_TX+ | Ethernet E0_TX+ |
| 2 | E0_TX- | Ethernet E0_TX- |
| 3 | E0_TX+ | Ethernet E0_TX+ |
| 4 | NC | Undefined |
| 5 | NC | Undefined |
| 6 | E0_RX- | Ethernet E0_RX- |
| 7 | NC | Undefined |
| 8 | NC | Undefined |

### 5. RS485 Interface (RS485)

| NO. | Identifier | Function Description |
|---|---|---|
| 1 | 1A | The first channel RS485 communication A port |
| 2 | 1B | The first channel RS485 communication B port |
| 3 | G | Grounding |
| 4 | 2A | Second channel RS485 communication A port |
| 5 | 2B | Second channel RS485 communication B port |

### 6. RS232 Interface (RS232)

| NO. | Identifier | Function Description |
|---|---|---|
| 1 | R3 | RS232 communication RX port |
| 2 | T3 | RS232 communication TX port |
| 3 | G | Grounding |

### 7. RS232 Interface (RS232)

| NO. | Identifier | Function Description |
|---|---|---|
| 1 | POW/RUN | RS232 communication RX port |
| 2 | RX1/TX1 | RS232 communication TX port |
| 3 | RX2/TX2 | Grounding |
| 4 | RX2/TX3 | |
| 5 | STATUE/VBAT | |

# 4.System software

## 1. Basic operation of the BootLoader

### 1.1 Introduction

◎Bootloader uses u-boot and its main functions are:

◎Parameter setting management

◎Download files to flash or ram via serial or network

◎Start the linux operating system

◎Query CPU, memory and other system parameters

◎Memory read and write

### 1.2 u-boot main commands

1.2.1 Parameter Settings

1.2.1.1 Query Parameters

Printenv

1.2.1.2 Setting Parameters

Setenv

Such as:

Setenv ipaddr 192.168.1.177 (Setting the IP address)

Setenv serverip 192.168.1.95 (Sets the IP address of the tftp server, that is, the local IP address)

1.2.1.3 Saving Parameters

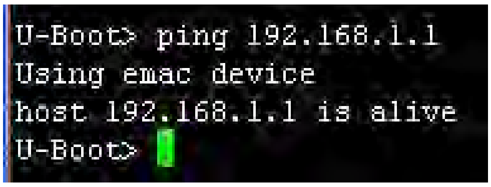
Saveenv

After the parameters are set, they are only saved in the memory. This operation is required to save the configured parameters in Flash. After the save is complete, it takes effect after the restart.

### 1.2.2 Network Commands

1.2.2.1 Testing Network Connections

Ping



```
U-Boot> ping 192.168.1.1
Using emac device
host 192.168.1.1 is alive
U-Boot>
```

### 1.2.2.2 Connecting TFTP Server

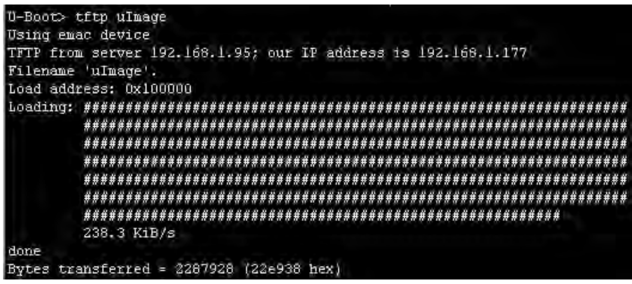
Tftp addr filename

Download the specified file filename from TFTP Server to the specified memory start address in the continuous space starting with addr

Such as:

Tftp uImage

Download file uImage from TFTP Server to continuous space starting at memory address 0x100000 (use default address 0x100000 by default)



```
U-Boot> tftp uImage
Using emac device
TFTP from server 192.168.1.95; our IP address is 192.168.1.177
Filename 'uImage'.
Load address: 0x100000
Loading: #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #########################################################
         238.3 KiB/s
done
Bytes transferred = 2287928 (22e938 hex)
```

### 1.2.3 The nandflash Command

1.2.3.1 Delete Command

Nand erase start len

Delete the specified starting address, the length of the nandflash space of the specified length

Such as:

Nand erase 200000 400000

Delete space starting from address 0x200000 and delete length 0x400000

Note: The length must be a multiple of 0x20000

### 1.2.3.2 Write Command

Nand write source start len

Writes the specified memory start address and contents of the specified length to the nandflash space of the specified address

Such as:

Nand write 100000 200000 280000

Writes the contents in the memory starting address 0x100000 to the space where the nandflash start address is 0x200000, and the write length is 0x280000

Note: The length must be a multiple of 0x20000

1.2.4 Startup Command

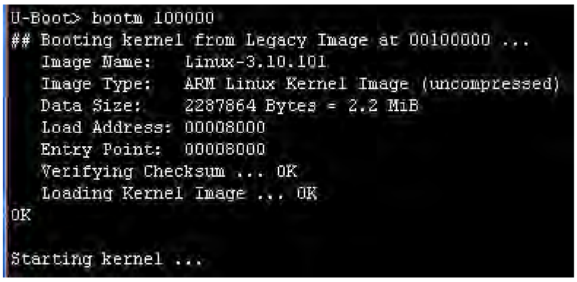
Bootm

Start the linux operating system

Such as:

Bootm 100000

Start the linux kernel starting at memory address 0x100000 (provided that the linux kernel has been downloaded to SDRAM starting address 0x100000)



```
U-Boot> bootm 100000
## Booting kernel from Legacy Image at 00100000 ...
   Image Name:   Linux-3.10.101
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    2287864 Bytes = 2.2 MiB
   Load Address: 00008000
   Entry Point:  00008000
   Verifying Checksum ... OK
   Loading Kernel Image ... OK
OK

Starting kernel ...
```

**1.2.5 Reset Command**

Reset

Reset the system and restart u-boot

**1.2.6 Help**

Help

Display help information

## 3. File system

The file system is based on the method of naming, storing, organizing, and retrieving a definition file on a logical unit on a storage device. If a Linux does not have a root file system, it cannot be started properly. Therefore, we need to create a root file system for Linux and store it on NAND FLASH. This system uses the UBI file system

3.1 Linux root file system directory

3.1.1 /bin(binary): Contains all standard commands and applications

3.1.2 /dev(device): File interface containing peripherals. Under Linux, files and devices are accessed in the same way. Each device on the system has a corresponding device file in /dev.

3.1.3 /etc (etcetera): The directory contains the system configuration files and other system files, for example, /etc/fstab (file system table) records the filesystem to mount at startup.

3.1.4 /home: Normal User Home Directory

**2. Linux kernel**

The LT8103 series uses a standard embedded Linux system to enable the migration of POSIX-compliant or UNIX-like applications to the system. The system is the main feature:

◎Support full TCP/IP protocol, support PPP protocol

◎Supports Telnet, providing users with the ability to complete remote host

◎work on the local computer

◎Support System V IPC

◎Support for shared memory

◎Software supports floating-point operations

◎Support Memory Technology Device technology, use NandFlash as system

◎storage medium

◎File system:

♦Ext2 linux file system

♦NFS network file system

♦Udev device file system

♦Proc kernel file system

♦Fat dos file system

♦UBI file system

◎Device driver

♦USB driver

♦Network driver

♦UART serial port driver

♦RTC real-time clock

♦Watchdog (WTD) driver

♦SD/MMC card driver

♦IIC drive

3.1.5 /lib(library): Store the system's most basic library files

3.1.6 /mnt: Where a User Mounts a File System Temporarily

3.1.7 /proc: A virtual system provided by Linux. It is generated in memory when the system is started. Users can directly access these files to obtain system information.

3.1.8 /root: Superuser Home Directory

3.1.9 /sbin: Stores system management programs such as fsck, mount, etc.

3.1.10 /tmp(temporary): Store temporary files generated when different programs are executed

3.1.11 /usr(user): Store User Applications and Files

3.1.12 /program: Operation script and test instance for storing peripherals

### 3.2 BusyBox

BusyBox was originally written by Bruce Perens in 1996 for the Debian GNU/Linux installation disk. The goal is to create a bootable GNU/Linux system on a floppy disk that can be used as an installation disk and rescue disk. A floppy disk can save about 1.4-1.7 MB of content, so there is not much space left for the Linux kernel and related user applications.

BusyBox exposes the fact that many standard Linux tools can share many common elements. For example, many file-based tools (such as grep and find) need to search for files in the directory. When these tools are merged into an executable program, they can share these same elements, so that smaller executables can be produced. In fact, BusyBox can pack about 3.5MB of tools into about 200KB in size. This provides more features for bootable disks and embedded devices using Linux.

Using BusyBox is a good way to reduce the size of the root file system, because it provides many basic instructions for the system, and its size is small. As we all know, the Swiss Army Knife is world-famous for its compactness, light weight, and numerous functions. It has become an indispensable tool for soldiers of various countries and is widely used in the folk. BusyBox is also called the "Swiss Army Knife" in the embedded Linux field.

### 3.3 LT8103 Main Commands

The LT8103 uses BusyBox to reduce system overhead and retains some major and commonly used commands

#### 3.3.1 File List

Ls

See the current directory

Ls+ directory name (including path)

Check the file in the specified directory name

Instructions:

Ls //View current directory

Ls /home/user / / View the file in user

#### 3.3.2 Replacing the Current Directory

Cd

Instructions:

Cd dir / / Replace to the current directory dir directory

Cd / / / change to the root directory

Cd .. // Switch to the upper directory

#### 3.3.3 Copy

Cp

Instructions:

Cp src des //Copy the file src to des

Cp /root/src ./ / copy the file src under root to the current directory

Cp -av src_dir des_dir //Copy the directory src_dir to des_dir, the contents of the two directories are the same

Cp -fr src_dir des_dir //Copy the entire directory in non-link mode. When the src_dir directory has a symbolic link, the two directories are different.

#### 3.3.4 System Time Date

Date

Instructions:

Date //Display current system date and time

Date –s 12:34:56 //Set the system time to 12:34:56

Date –s 2010-01-02 //set the system date to 2010-01-02

#### 3.3.5 Distribution View

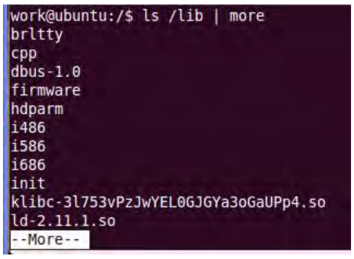
More

Instructions:

More //Paging commands, which are usually piped to content

Ls /filename | more //By piped to more, paginate view the file under directory filename

For a directory with more files, use the above command to view the page, press Enter to scroll down one line, press the Space bar to scroll down one page, and press q to exit to view. Such as:



```
work@ubuntu:/$ ls /lib | more
brltty
cpp
dbus-1.0
firmware
hdparm
i486
i586
i686
init
klibc-3l753vPzJwYEL0GJGYa3oGaUPp4.so
ld-2.11.1.so
--More--
```

#### 3.3.6 Echoes

Echo

Instructions:

Echo "message" / / display a string of characters

Echo "message message2" // displays discontinuous strings

#### 3.3.7 Mounting

Mount

Instructions:

Mount -t nfs 192.168.1.100:/nfs /mnt / / Mount the nfs service's shared directory / nfs to the local / mnt directory

#### 3.3.8 Move

Mv

Instructions:

Mv src des // rename the file src to des

Mv /file1/src /file2/ // Move the src file under file1 to the file2 directory

#### 3.3.9 Deletion

Rm

Instructions:

Rm file_name // Delete a file called file_name

Rm -rf dir //Delete the entire directory named dir in the current directory

#### 3.3.10 Search

Grep

Instructions:

Grep -ir "chars" * / / find the string chars in all files in the current directory, and ignore the size set (-i), to find the subdirectory (-r) including the current directory.

### 3.3.11 Find

Find

Instructions:

Find /path-name file / / find the file named file in the path directory

### 3.3.12 Viewing the Content of a File

Cat

Instructions:

Cat file / / display the contents of the file file (in ASCII code)

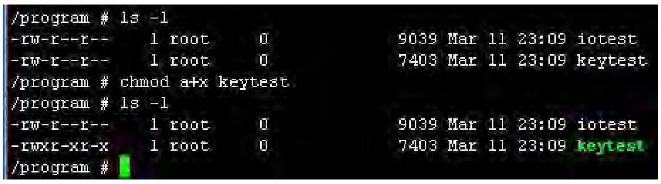
### 3.3.13 permission modification

Chmod

Instructions:

Chmod a+x file //Set the file file to executable, the script file must be set this way, otherwise you need to use bash file to execute. The parameter a allows all users to have executable permissions on this file.

For example, add executable permissions to the keytest file:



```
/program # ls -l
-rw-r--r--    1 root     0            9039 Mar 11 23:09 iotest
-rw-r--r--    1 root     0            7403 Mar 11 23:09 keytest
/program # chmod a+x keytest
/program # ls -l
-rw-r--r--    1 root     0            9039 Mar 11 23:09 iotest
-rwxr-xr-x    1 root     0            7403 Mar 11 23:09 keytest
/program # 
```

Similarly, you can use the chmod a+r file and chmod a+w file commands respectively to set read and write permissions for the file.

### 3.3.14 Compression and Decompression

tar

Decompression method of use:

tar -xjvf filename.tar.bz2 //Unzip the file filename.tar.bz2 to the current directory

tar -xzvf filename.tar.gz //Unzip the file filename.tar.gz to the current directory

Compression method:

tar -cjvf filename.tar.bz2 /filename //Compress the file or folder filename to filename.tar.bz2

tar -czvf filename.tar.gz /filename //Compress file or folder filename to filename.tar.gz

Parameter Description:

-x: Extract the compressed file

-c: create a compressed file

-j: Extract or compress files with the .bz2 suffix

-z: Extract or compress files with the .gz suffix

-v: Displays the extracted or compressed file

-f: Specifies the file name of the package

### 3.3.15 Editing

vi

Instructions:

vi file //Edit file file

In the vi state, enter the command is first press ctrl + c, and then enter

:q //exit

:q! //Exit does not save

:wq //Save and exit

:w //write to file

:r file // Write to file without asking

%s/oldchars/newchars/g // Replace all strings oldchars with newchars

### 3.3.16 Creating a Node

Mknod

Instructions:

Mknod /dev/tty1 c 4 1 //Create the character device tty1, the major device number is 4, the slave device number is 1, ie the first tty terminal.

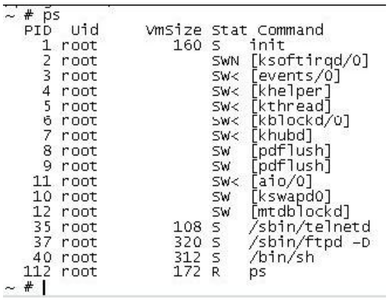
### 3.3.17 Process View

ps

Instructions:

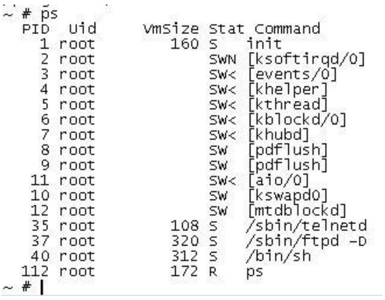
ps / / display the current system process information

As shown below:



```
~ # ps
  PID  Uid        VmSize Stat Command
    1 root           160 S   init
    2 root               SWN [ksoftirqd/0]
    3 root               SW< [events/0]
    4 root               SW< [khelper]
    5 root               SW< [kthread]
    6 root               SW< [kblockd/0]
    7 root               SW< [khubd]
    8 root               SW  [pdflush]
    9 root               SW  [pdflush]
   11 root               SW< [aio/0]
   10 root               SW  [kswapd0]
   12 root               SW  [mtdblockd]
   35 root           108 S   /sbin/telnetd
   37 root           320 S   /sbin/ftpd -D
   40 root           312 S   /bin/sh
  112 root           172 R   ps
~ # |
```

ps-ef //Display all system process information



```
~ # ps
  PID  Uid        VmSize Stat Command
    1 root           160 S   init
    2 root               SWN [ksoftirqd/0]
    3 root               SW< [events/0]
    4 root               SW< [khelper]
    5 root               SW< [kthread]
    6 root               SW< [kblockd/0]
    7 root               SW< [khubd]
    8 root               SW  [pdflush]
    9 root               SW  [pdflush]
   11 root               SW< [aio/0]
   10 root               SW  [kswapd0]
   12 root               SW  [mtdblockd]
   35 root           108 S   /sbin/telnetd
   37 root           320 S   /sbin/ftpd -D
   40 root           312 S   /bin/sh
  112 root           172 R   ps
~ # |
```

### 3.3.18 Kill Process

Kill

Instructions:

Kill -9 250 // Kill program with mile number 250

### 3.3.19 Setting Environment Variables

Export

Instructions:

Export LC_ALL=zh_CN.GB2312 //Set the value of the environment variable LC_ALL to zh_CN.GB2312

### 3.3.20 Startup Information Display

Dmesg

Instructions:

Dmesg / / display kernel startup and driver loading information

### 3.3.21 Network Settings Command

Ifconfig

Instructions:

Ifconfig / / check the status of the network card, the implementation of the network card and the local loop (lo, loopback)

Ifconfig eth0 192.168.1.230 netmask 255.255.255.0 // indicates that the address of the network card 1 is 192.168.1.230, the mask is 255.255.255.0, and if not, the default is 255.255.255.0.

Note: The development board has missing network IP, which is achieved by using ifconfig in the startup script /etc/inin.d/rcS file

Ifconfig eth0 down //Close network card 1

Ifconfig eth0 up // start network card 1

Set the MAC address:

Ifconfig eth0 down

Ifconfig eth0 hw ether xx:xx:xx:xx:xx:xx

Ifconfig eth0 up

Where xx:xx:xx:xx:xx:xx is the set MAC address, such as 00:12:34:56:78:90, note that the first byte must be even

### 3.3.22 Setting Up a Gateway

Route

Instructions:

Route //Display current routing settings

Route add default gw 192.168.1.1 //Set 192.168.1.1 as the default route

Route del default //Delete the default route

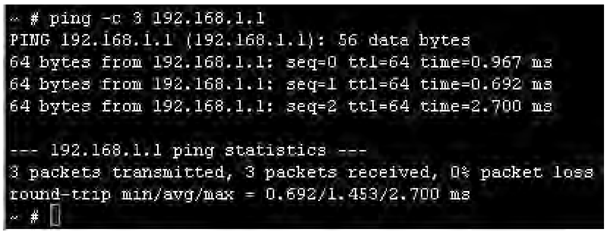
Route add –net 192.168.1.0 netmask 255.255.255.0 gw 192.168.1.1 dev eth0

### 3.3.23 Testing Network Connectivity

Ping

Instructions:

Ping –c 3 192.168.1.1 //Send three consecutive test packets to 192.168.1.1 to verify that the network is connected. If the connection is normal, the result is as follows:



```
~ # ping -c 3 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: seq=0 ttl=64 time=0.967 ms
64 bytes from 192.168.1.1: seq=1 ttl=64 time=0.692 ms
64 bytes from 192.168.1.1: seq=2 ttl=64 time=2.700 ms

--- 192.168.1.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.692/1.453/2.700 ms
~ #
```

### 3.3.24 RTC Clock Commands

Hwclock

Instructions:

Hwlock //Display current RTC time

Hwclock -s / / synchronize the current RTC time to Linux system time

Hwclock -w //Time to synchronize Linux system time to RTC

### 3.3.25 System Reset Command

Reboot

Instructions:

Reboot //System restart

## 5.Application

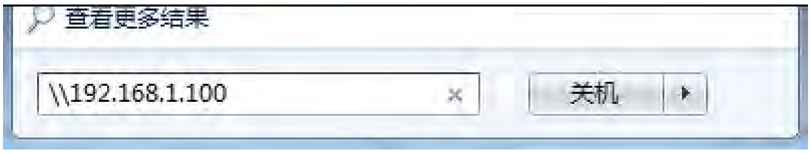
### 1. Development environment

Ubuntu 10.04, eclipse

Cross Compilation Tools: arm-linux-gnueabihf-gcc (Version 4.8.4)

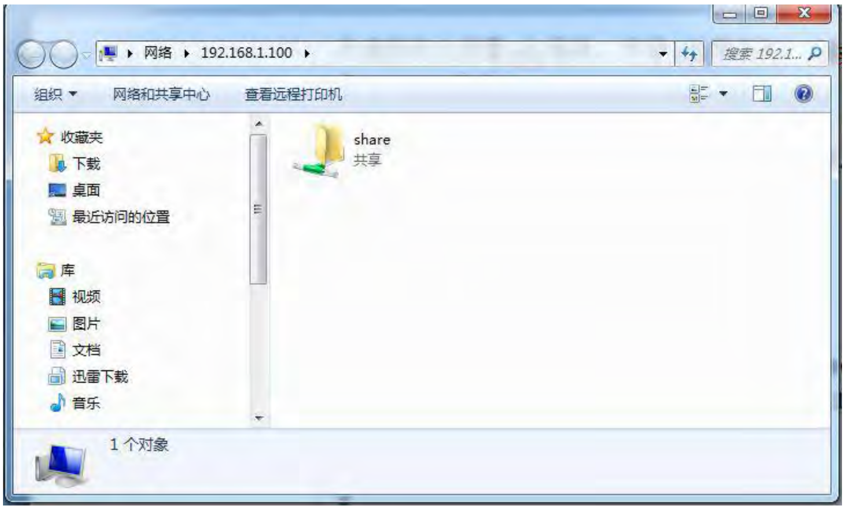
### 2 virtual machine and windows share directory

2.1 The virtual machine in the CD-ROM starts the samba service and can share the directory with Windows for easy file copying. In the Windows Start menu, select Run, enter \\192.168.1.101 (this IP is the IP address of the virtual machine), and then press Enter



```
\\192.168.1.100
```

2.2 The first use requires authentication of login information. Enter the password in the popup window. The username and password are the same as those for logging in to the VM.



**3. Use NFS network file system**

3.1 Server

Set up the NFS service on the virtual machine and open the NFS path, such as /nfs

3.2 Terminal Equipment

Run the command: mount –t nfs 192.168.1.100:/nfs /mnt –o nolock

/mnt is the content of the /nfs directory of the virtual machine (Note: 192.168.1.100 is the IP address of the virtual machine)

**4. Boot from boot**

4.1 Create a startup script

Create a startup.sh file on the target board file system/program and enter the following:

#! /bin/sh

/prog/myprog # startup program myprog and path

4.2 Modifying Attributes

Chmod +x startup.sh

4.3 Restart the system

Reboot

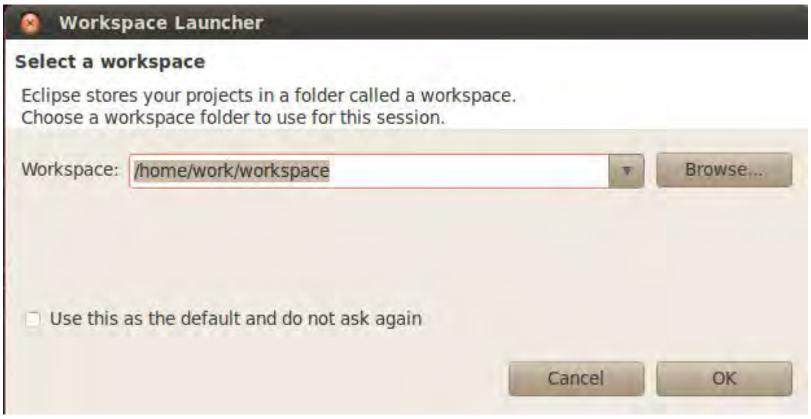
**5. Eclipse instructions**

5.1 Create a simple helloworld program under the virtual machine

In the terminal input: eclipse, waiting for the programming software to start

```
work@ubuntu:~$ eclipse
```

5.2 Select the working path, you can use the default directory, you can also customize the directory.



Workspace Launcher

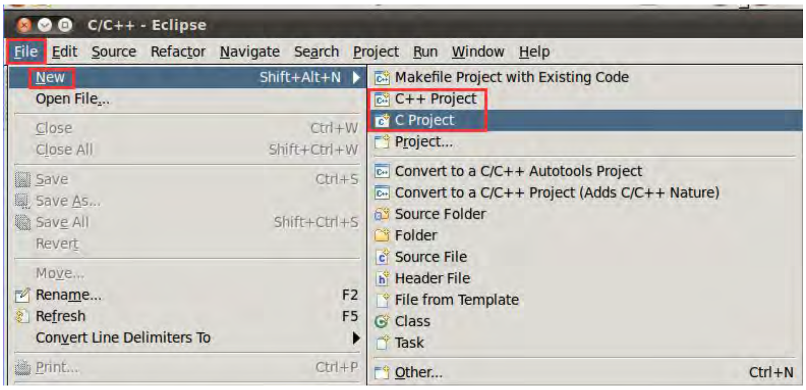
Select a workspace

Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

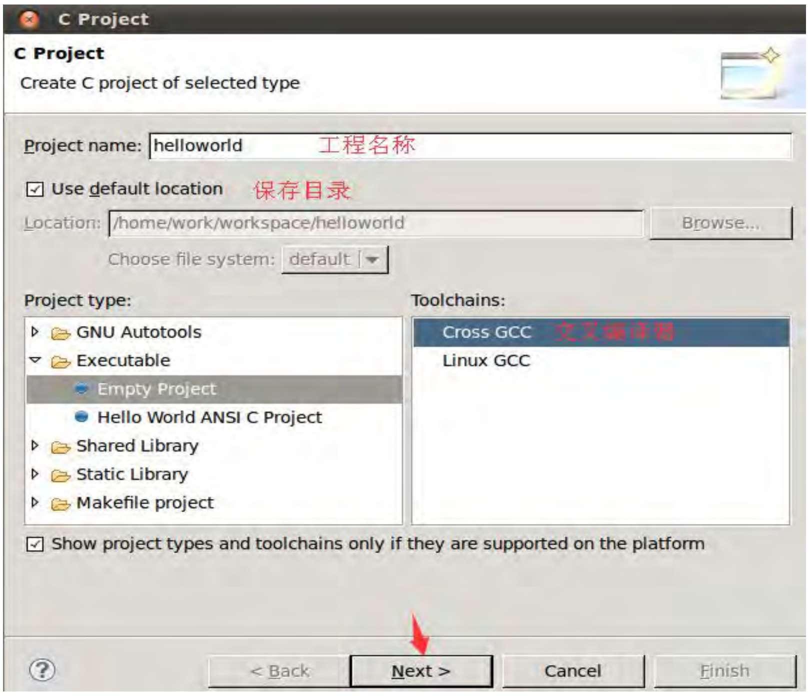
Workspace: /home/work/workspace

□ Use this as the default and do not ask again

5.3 new project: File New c project (C + + program choice c + + project), as shown



5.4 Enter the project name, select the save path, select cross-compile, as shown in the following figure, complete the next step.



C Project

C Project

Create C project of selected type

Project name: helloworld 工程名称

☑ Use default location 保存目录

Location: /home/work/workspace/helloworld

Choose file system: default

Project type:

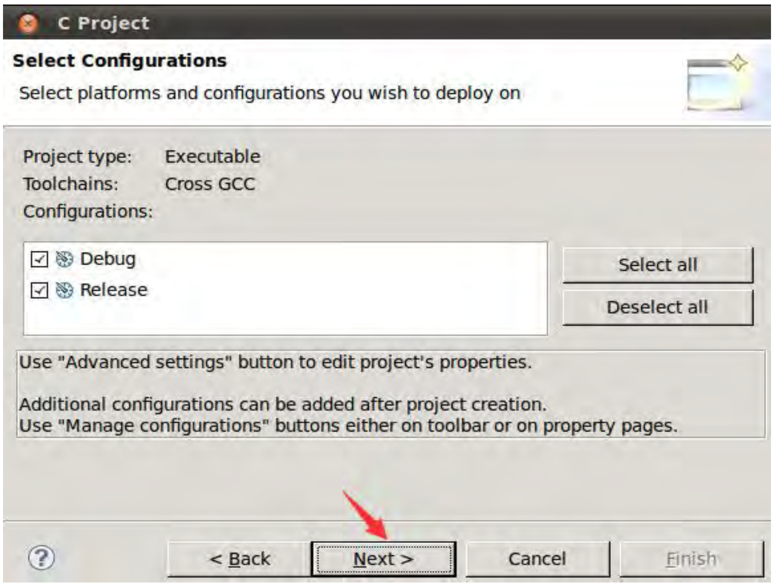
- GNU Autotools
- Executable
  - Empty Project
  - Hello World ANSI C Project
- Shared Library
- Static Library
- Makefile project

Toolchains:

- Cross GCC 交叉编译器
- Linux GCC

☑ Show project types and toolchains only if they are supported on the platform

5.5 This step uses the default configuration and continues to the next step.



C Project

Select Configurations

Select platforms and configurations you wish to deploy on

Project type: Executable

Toolchains: Cross GCC

Configurations:

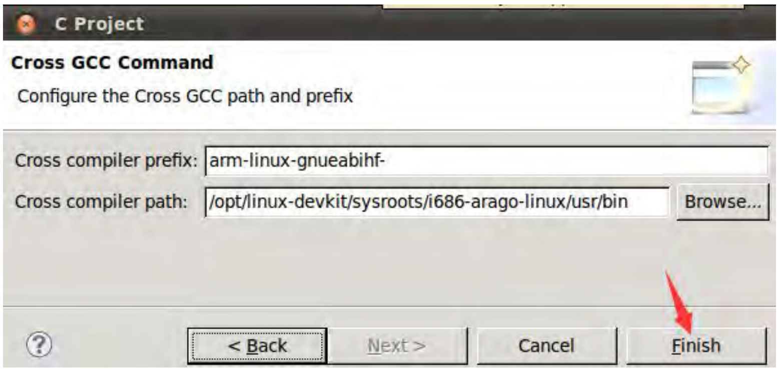
☑ Debug

☑ Release

Use "Advanced settings" button to edit project's properties.

Additional configurations can be added after project creation.
Use "Manage configurations" buttons either on toolbar or on property pages.

5.6 Fill in the cross compiler and path, as shown in the following figure (the CD provides a configured virtual machine system, the system includes a cross compiler), fill in and select finish, complete the project creation.



C Project

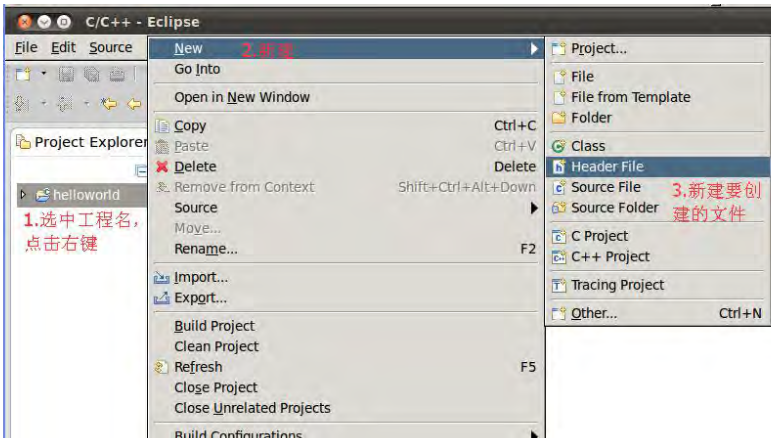
Cross GCC Command

Configure the Cross GCC path and prefix

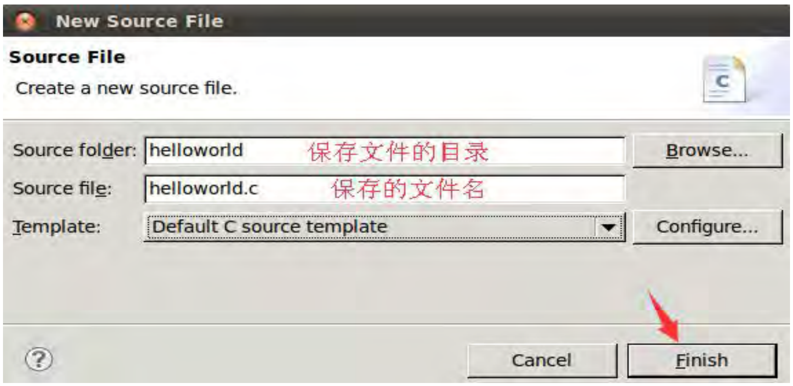
Cross compiler prefix: arm-linux-gnueabihf-

Cross compiler path: /opt/linux-devkit/sysroots/i686-arago-linux/usr/bin

5.7 Adding Files to a Project



5.8 input file name, complete the creation of the source file.

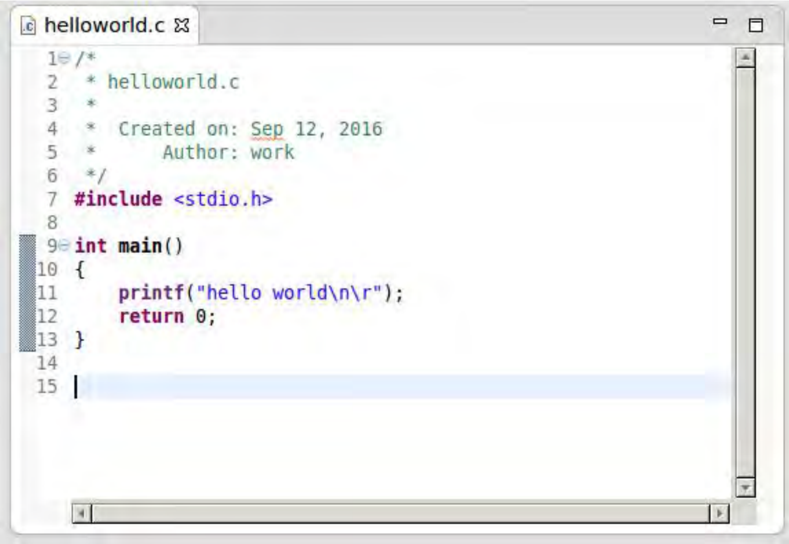


Source folder: helloworld 保存文件的目录

Source file: helloworld.c 保存的文件名

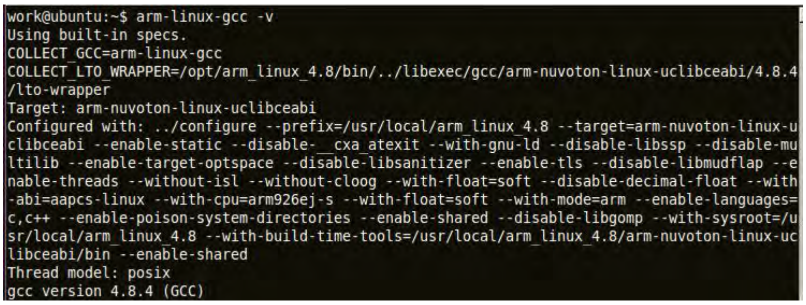
Template: Default C source template

5.9 Writing Code.



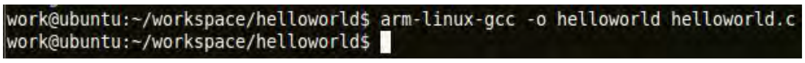
```
/*
 * helloworld.c
 *
 *  Created on: Sep 12, 2016
 *      Author: work
 */
#include <stdio.h>

int main()
{
    printf("hello world\n\r");
    return 0;
}

```

5.10 Save Files, Cross-Compilation, Compiler Version 4.8.4.



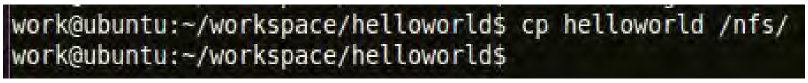
```
work@ubuntu:~$ arm-linux-gcc -v
Using built-in specs.
COLLECT_GCC=arm-linux-gcc
COLLECT_LTO_WRAPPER=/opt/arm_linux_4.8/bin/../libexec/gcc/arm-nuvoton-linux-uclibceabi/4.8.4
/lto-wrapper
Target: arm-nuvoton-linux-uclibceabi
Configured with: ../configure --prefix=/usr/local/arm_linux_4.8 --target=arm-nuvoton-linux-u
clibceabi --enable-static --disable-__cxa_atexit --with-gnu-ld --disable-libssp --disable-mu
ltilib --enable-target-optspace --disable-libsanitizer --enable-tls --disable-libmudflap --e
nable-threads --without-isl --without-cloog --with-float=soft --disable-decimal-float --with
-abi=aapcs-linux --with-cpu=arm926ej-s --with-float=soft --with-mode=arm --enable-languages=
c,c++ --enable-poison-system-directories --enable-shared --disable-libgomp --with-sysroot=/u
sr/local/arm_linux_4.8 --with-build-time-tools=/usr/local/arm_linux_4.8/arm-nuvoton-linux-uc
libceabi/bin --enable-shared
Thread model: posix
gcc version 4.8.4 (GCC)
```

Do not show any hints indicating that the compiler passed



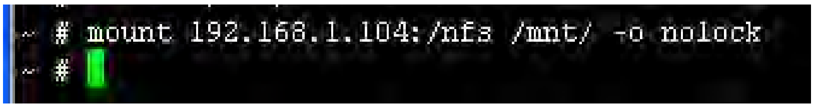
```
work@ubuntu:~/workspace/helloworld$ arm-linux-gcc -o helloworld helloworld.c
work@ubuntu:~/workspace/helloworld$ 
```

5.11 Copy the compiled file to the nfs directory (nfs server server installation, refer to the related documents of the virtual machine settings)



```
work@ubuntu:~/workspace/helloworld$ cp helloworld /nfs/
work@ubuntu:~/workspace/helloworld$
```

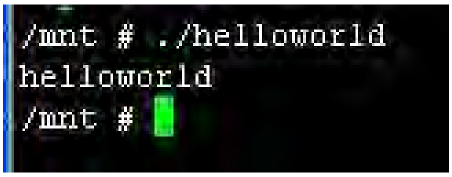
5.12 Mounting an nfs Network File System on an LT8103 Terminal (You do not need to mount it before restarting).



```
~ # mount 192.168.1.104:/nfs /mnt/ -o nolock
~ #
```

192.168.1.104 is the VM IP; /nfs is the virtual directory; /mnt is the LT8103 system directory.

5.13 /mnt directory, run the program.

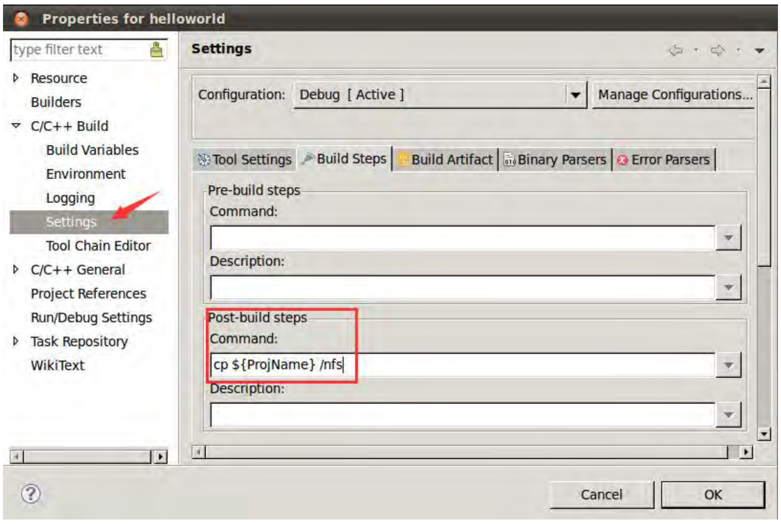


```
/mnt # ./helloworld
helloworld
/mnt #
```

**Attachment 1: eclipse parameter configuration instructions**

Attached 1.1 automatically copy files to the specified directory

Select the project menu →Properties→ C / C + + Build →Settings, select the tab Build Steps, in the Post-build steps Command box, enter: cp $ {projName} / nfs, OK completed.



Properties for helloworld

Settings

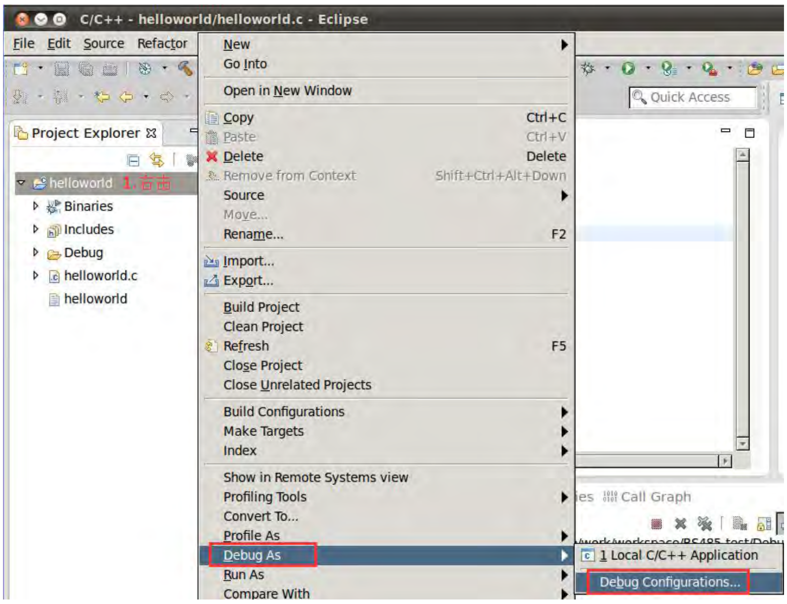
Configuration: Debug [ Active ]

Post-build steps

Command: cp ${ProjName} /nfs

Attached 1.2 debugging settings

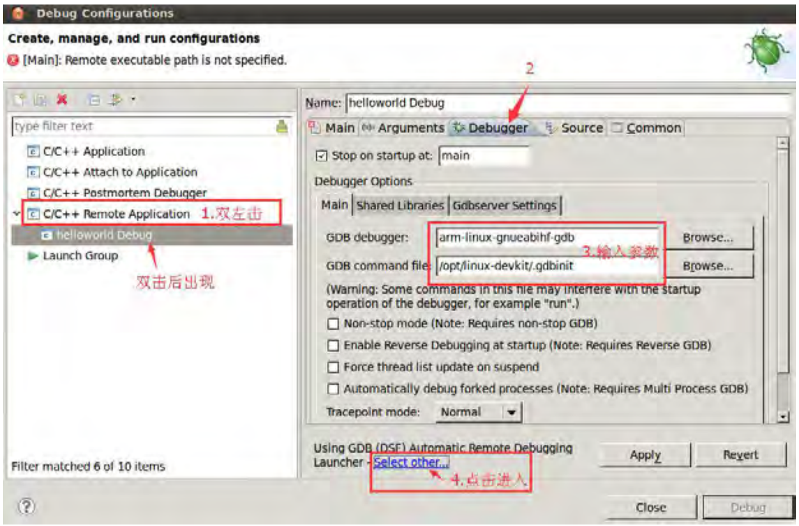
In the project browser, select the project name and select Debug As → Debug Configuration from the right-click menu.



Attached 1.2.1 In the dialog box that opens, configure according to the following picture, where the parameters of the third step are:

GDB debuger::arm-linux-gnueabihf-gdb

GDB command file::/opt/linux-devkit/.gdbinit


Debug Configurations

Create, manage, and run configurations

[Main]: Remote executable path is not specified.

Name: helloworld Debug

☑ Stop on startup at: main

GDB debugger: arm-linux-gnueabihf-gdb

GDB command file: /opt/linux-devkit/.gdbinit

(Warning: Some commands in this file may interfere with the startup operation of the debugger, for example "run".)

☐ Non-stop mode (Note: Requires non-stop GDB)

☐ Enable Reverse Debugging at startup (Note: Requires Reverse GDB)

☐ Force thread list update on suspend

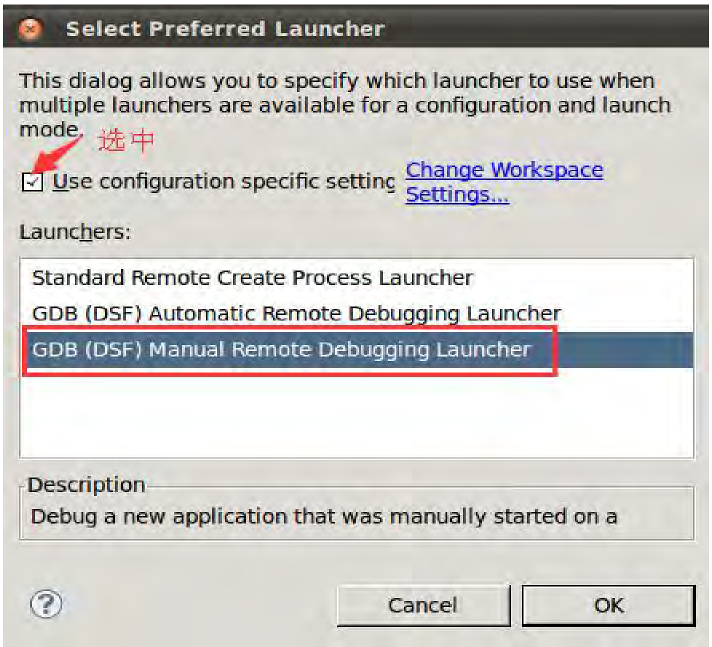
☐ Automatically debug forked processes (Note: Requires Multi Process GDB)

Tracepoint mode: Normal

Using GDB (DSF) Automatic Remote Debugging Launcher - Select other...

Filter matched 6 of 10 items

Attached 1.2.2 Select Loading Method as Manual Loading.



Select Preferred Launcher

This dialog allows you to specify which launcher to use when multiple launchers are available for a configuration and launch mode.

选中

☑ Use configuration specific setting

Launchers:

Standard Remote Create Process Launcher
GDB (DSF) Automatic Remote Debugging Launcher
GDB (DSF) Manual Remote Debugging Launcher

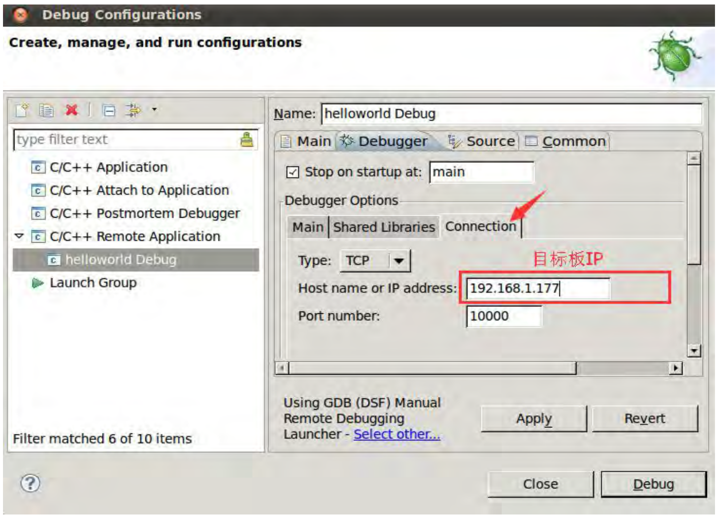
Description
Debug a new application that was manually started on a

Attached 1.2.3 Connection Properties:

Host name or IP address:192.168.1.177

Port number: 10000 (The target board must be connected and debugged in the same way)

Select apply to enable, select Close to exit



Debug Configurations

Create, manage, and run configurations

Name: helloworld Debug

☑ Stop on startup at: main

Debugger Options

目标板IP

Type: TCP

Host name or IP address: 192.168.1.177

Port number: 10000

Using GDB (DSF) Manual Remote Debugging Launcher - Select other...

Filter matched 6 of 10 items

Attached 1.2.4 target board configuration

1) Open the HyperTerminal and configure it to 115200bps, 8N1

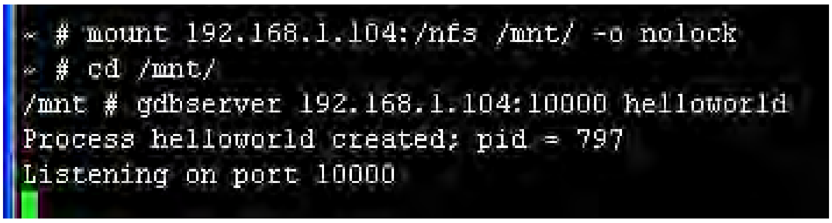
2) Execute the command: mount 192.168.1.104:/nfs /mnt -o nolock

// Mount the network file system, where 192.168.1.104 is the virtual machine IP address

3) Execute the command: gdbserver 192.168.1.104:10000 helloworld

//10000 is the port number, which must be the same as the port number in the eclipse configuration. helloworld is the name of the program to be debugged.

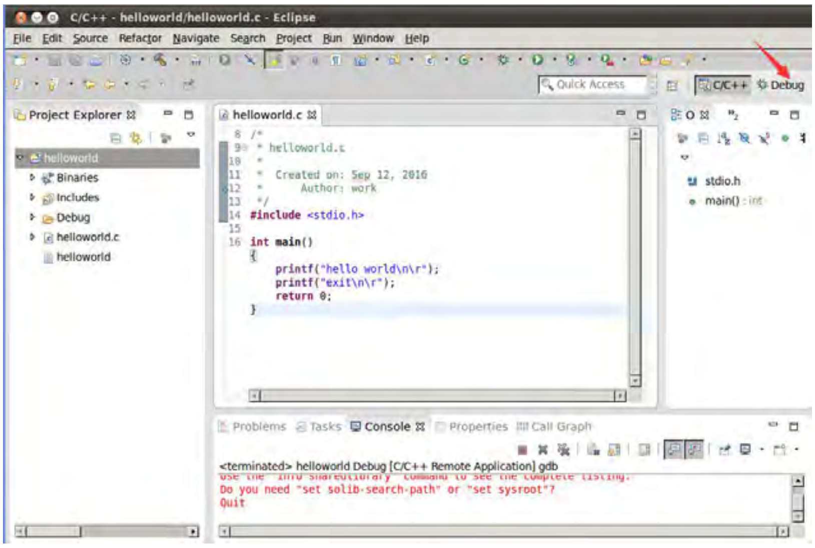
4) Execution is complete, gdbserver is already listening, as shown below



```
~ # mount 192.168.1.104:/nfs /mnt/ -o nolock
~ # cd /mnt/
/mnt # gdbserver 192.168.1.104:10000 helloworld
Process helloworld created; pid = 797
Listening on port 10000
```

Attached 1.2.5 Debugging on the Eclipse Side of a Virtual Machine

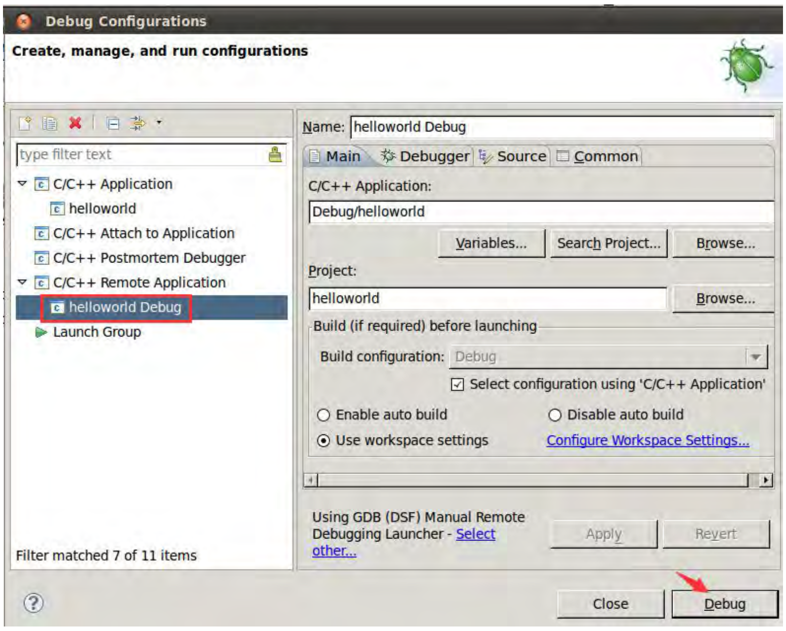
1) In the eclipse development environment, switch to Debug mode



```
/*
 * helloworld.c
 *
 *  Created on: Sep 12, 2016
 *      Author: work
 */
#include <stdio.h>

int main()
{
    printf("hello world\n\r");
    printf("exit\n\r");
    return 0;
}
```

```
<terminated> helloworld Debug [C/C++ Remote Application] gdb
Use the "info sharedlibrary" command to see the complete listing.
Do you need "set solib-search-path" or "set sysroot"?
Quit
```

Or select the menu Run →Debug Configurations, enter the configuration window, select helloworld Debug in the left window, click Debug in the lower right corner, you can debug mode (into the debug mode, the target board must first open gdbserver listen), if there is a window Pop up, please click Yes.



Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
  - helloworld
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
  - helloworld Debug
- Launch Group

Filter matched 7 of 11 items

Name: helloworld Debug

Main | Debugger | Source | Common

C/C++ Application:

Debug/helloworld

Variables... | Search Project... | Browse...

Project:

helloworld

Browse...

Build (if required) before launching

Build configuration: Debug

☑ Select configuration using 'C/C++ Application'

○ Enable auto build ○ Disable auto build

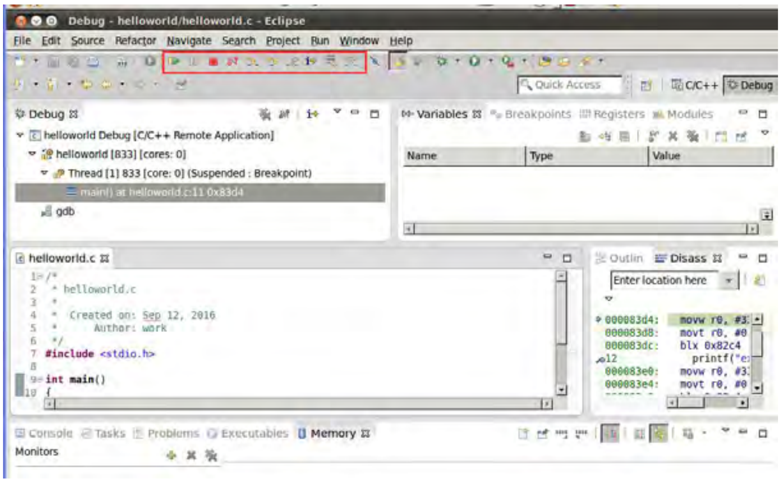
● Use workspace settings Configure Workspace Settings...

Using GDB (DSF) Manual Remote Debugging Launcher - Select other...

Apply | Revert

Close | Debug

2) Click the first icon for single-step operation.



Debug - helloworld/helloworld.c - Eclipse

Debug

- helloworld Debug [C/C++ Remote Application]
  - helloworld [833] [cores: 0]
    - Thread [1] 833 [core: 0] (Suspended : Breakpoint)
      - main() at helloworld.c:11 0x83d4
  - gdb

helloworld.c

```
/*
 * helloworld.c
 *
 *  Created on: Sep 12, 2016
 *      Author: work
 */
#include <stdio.h>

int main()
{
```

```
000083d4:   movw r0, #3:
000083d8:   movt r0, #0
000083dc:   blx 0x82c4
12            printf("e:
000083e0:   movw r0, #3:
000083e4:   movt r0, #0
```

In the system / program directory there is a corresponding script file, you can perform some simple tests

1. Use of RS485 interface driver

The corresponding driver for RS485-1 is ttyS1; the driver for RS485-2 is ttyS2

1.1 RS485 send data

Attachment: ssend.c

1.2 RS485 receive data

Attachment: sread.c

2. GPRS operation

The corresponding driver for GPRS is ttyS4

2.1 GPRS send and receive data

Attachment: at_u.c

2.2 GPRS power-on dialing

GPRS dial-up script for dial-on.sh, if you need to dial-up, you can start the script needs to be added to the script startup.sh, such as startup.sh does not exist, you can refer to Chapter 5 of the boot from the start step Add to.