

Software Guide

ICP DAS LP-51xx SDK

Implement industry control with Linux Technique

Version 2.0

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. assumes no liability for any damages 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

Copyright © 2011 by ICP DAS Co., Ltd. All rights are reserved.

Trademark

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

License

The user can use, modify and backup this software **on a single machine**. The user may not reproduce, transfer or distribute this software, or any copy, in whole or in part.

Contents

1. Introduction	4
2. Installation of LinPAC PXA270 SDK.....	6
2.1 Quick Installation of the LinPAC PXA270 SDK	7
2.1.1 Download/Install SDK on Linux	7
2.1.2 Download/Install SDK on Windows.....	8
2.1.3 Integrating SDK with Code::Blocks IDE	11
2.2 Introduction of the LinPAC PXA270 SDK	15
2.2.1 Introduction to Cygwin	15
2.2.2 Introduction to Cross-Compilation	15
2.2.3 Download the LinPAC PXA270 SDK	16
3. The Architecture of library in the LP-51xx	17
4. LP-51xx System Settings	19
4.1 LP-51xx Network Settings	19
4.1.1 Configuring the IP, Netmask and Gateway Addresses	19
4.1.2 DNS Setting	20
4.2 Usage a microSD Card	21
4.2.1 Mounting a microSD Card.....	21
4.2.2 Unmounting the microSD Card.....	21
4.2.3 Scanning and repairing microSD Card	22
4.3 Using a USB Storage Device.....	23
4.3.1 Mounting a USB Storage Device	23
4.3.2 Unmounting a USB Storage Device.....	24
4.4 Adjusting the VGA Resolution	24
4.5 Automatically executing applications at startup	25
4.5.1 Configuring a program to run at boot time	26
4.5.2 Disabling a program from running at boot time	28
4.6 Automatic login	28
5. Instructions for the LP-51xx	29
5.1 Basic Linux Instructions	29
5.2 General GCC Instructions	32
5.2.1 Compile without linking to the LP-51xx library	32
5.2.2 Compile by linking to the LP-51xx library (libi8k.a)	33
5.3 A Simple Example – Helloworld.c	34
5.4 i-Talk Utility	39
6. LIBI8K.A.....	40
6.1 System Information Functions	41
6.2 Watch Dog Timer Functions	63

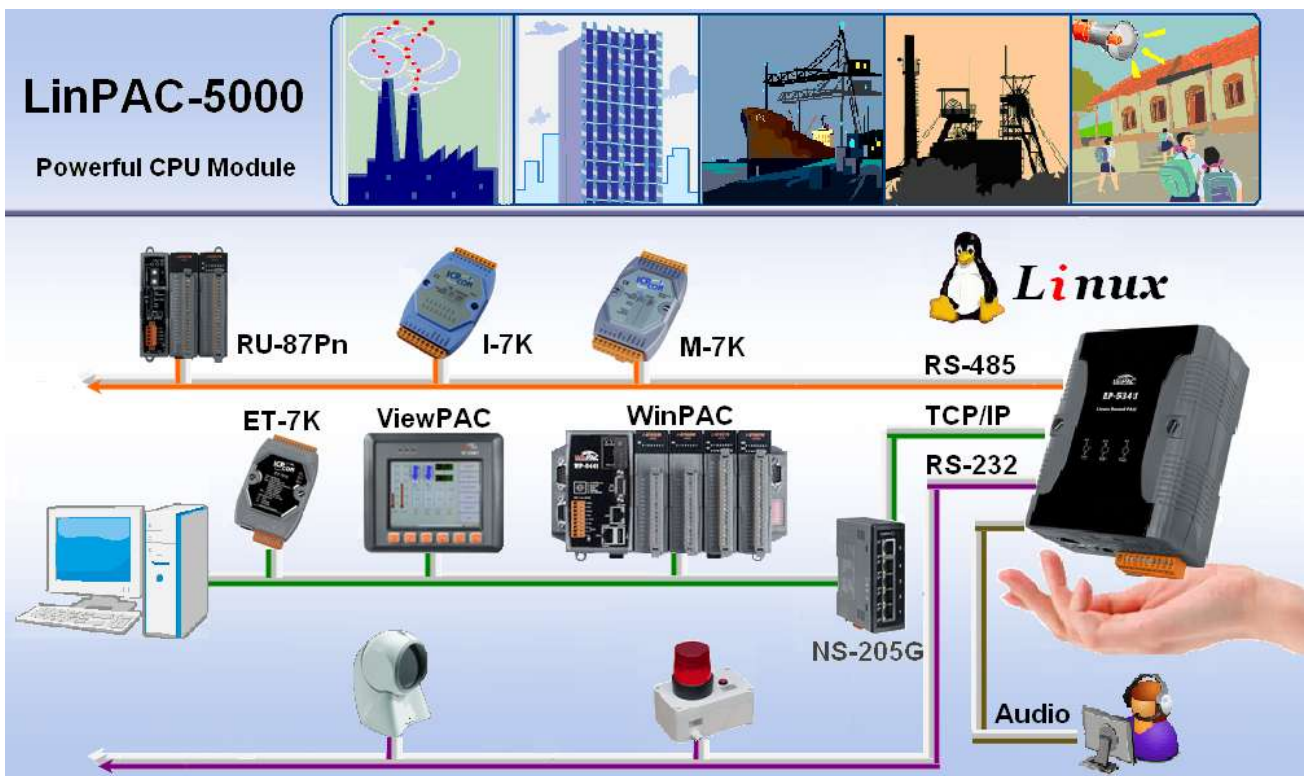
6.3 EEPROM Read/Write Functions.....	66
6.4 Digital Input/Output Functions	70
6.4.1 I-7000 series modules.....	70
6.5 Analog Input Functions	92
6.5.1 I-7000 series modules.....	92
6.6 Analog Output Functions	106
6.6.1 I-7000 series modules.....	106
6.7 Error Code Explanation	118
7. Demo for LP-51xx Modules With C Language	119
7.1 DIO Control Demo for I-7k Modules	119
7.2 AIO Control Demo for I-7k Modules	125
7.3 Overview of the Module Control Demo Program	127
7.4 Timer Function Demo.....	127
8. Overview of the Serial Ports on the LP-51xx.....	128
8.1 COM1 Port	129
8.2 COM2 Port	130
8.3 COM3 Port	131
9. LP-51xx Library Reference in C Language.....	133
9.1 List Of System Information Functions.....	133
9.2 List Of Digital Input/Output Functions	134
9.3 List Of Watch Dog Timer Functions	134
9.4 List Of EEPROM Read/Write Functions.....	134
9.5 List Of Analog Input Functions	135
9.6 List Of Analog Output Functions.....	135
10. Additional Support	136
10.1 Support for GUI Functionality.....	136
10.1.1 Booting the LP-51xx without loading the X-window environment	137
10.1.2 Enabling the X-window environment to load at boot time.....	137
10.2 Support for ScreenShot functionality	138
10.3 Support for WebCAM functionality	138
10.4 Support for Touch Screen Devices	139
10.4.1 USB Touch Screen interface.....	139
10.4.2 Serial Touch Screen interface	142
10.5 Network Support	146
10.6 Audio Function.....	153
10.7 Support for USB to RS-232 Conversion	154
Appendix A. Service Information	158
Internet Service:	158

1. Introduction

LP-51xx is the new generation Linux-based PAC from ICP DAS and is equipped with a PXA270 CPU (520 MHz) running a Linux kernel 2.6.19 operating system, variant connectivity (VGA, USB, Ethernet, RS-232/485 and audio port) and contains an optional I/O expansion board that can be used for implementing various I/O functions, such as D/I, D/O, A/D, D/A, Timer/Counter, UART, flash memory, etc.

The LP-51xx had the advantages of the good control system. These advantages include stability, small core size, I/O expansion board optional, support for Web services (Web/FTP/Telnet/SSH server), support for multiple development environments (LinPAC SDK for Linux and Windows environment using the GNU C language, JAVA, GUI software), etc. They give you all of the best features of both traditional PLCs and Linux capable PCs. That's the most powerful and flexible embedded control system.

Compared with the first generation LinCon-8000, not only has the CPU performance been improved and the OS upgraded from Linux kernel 2.4 to Linux kernel 2.6, but also many reliability features have been added, including a dual LAN, audio ports, I/O expansion board optional, etc., making the LP-51xx one of the most powerful control systems.



ICP DAS also provides a library file, **libi8k.a**, which includes all the functions from I-7000/8000/87000 series modules used in the LP-51xx Embedded Controller. The library is specially designed for I-7000/8000/87000 series modules based on the Linux platform for use with the LP-51xx controller. Custom applications can easily be developed for the LP-51xx using either C or Java and .NET applications will also be supported in the future. The various functions contained in the library are divided into sub-group functions for ease of use within the different applications. The powerful functions of the embedded controller are depicted below, including a **VGA port**, **USB ports for Card Readers, Cameras, Mouse, or Keyboard**, **microSD card**, **Series ports (RS-232/485)**, **Ethernet (Hub...)**, etc. in the picture. The LP-51xx controller contains built in HTTP, FTP, Telnet, SSH and SFTP Servers, meaning the file transfer or remote control is much more convenient with the LP-51xx. For network communication, **wireless**, **Bluetooth** transfer protocols and **Modem, GPRS, ADSL, Firewall** functions are also supported. The architecture of the LP-51xx hardware is illustrated in the figure below.

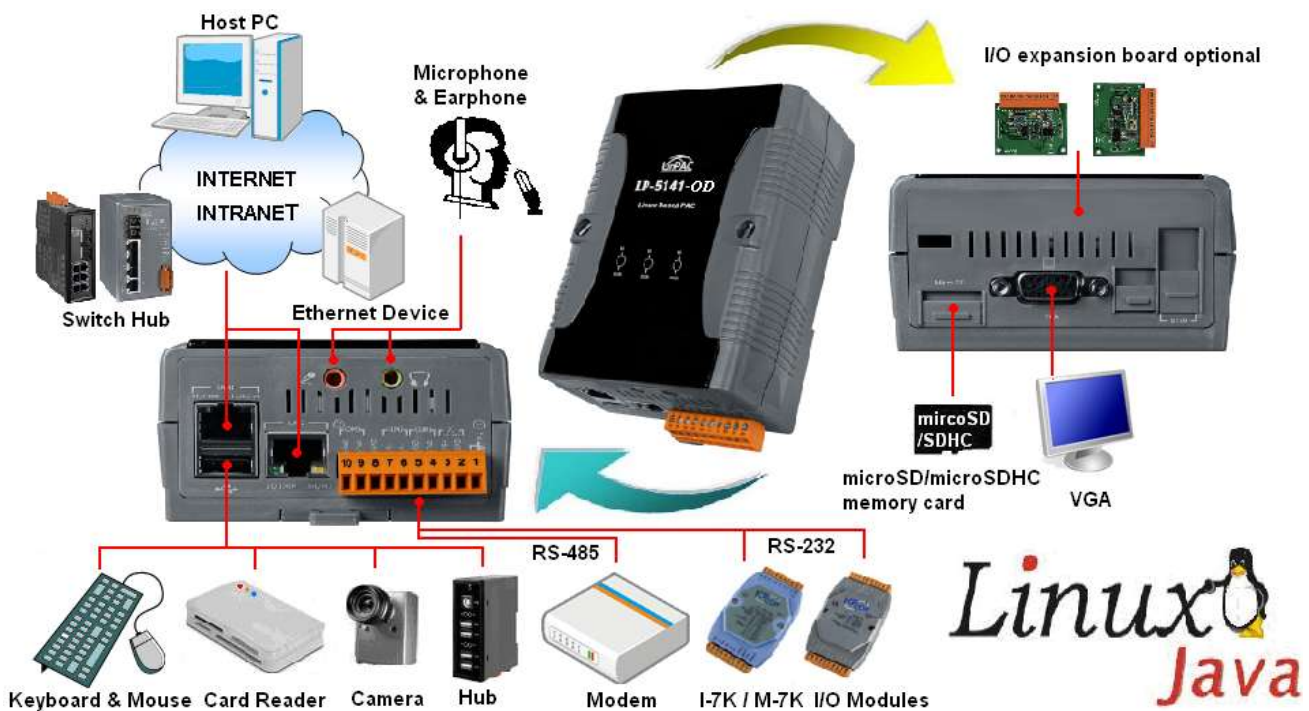


Fig. 1-1

Please note:

- ◆ The flash and microSD disk have a finite number of program-erase cycles. Important information should always be backed up on other media or storage device for long-term safekeeping.
- ◆ The Li-batterie can continually supply power to the 512 KB SRAM to retain the data for 10 years (It is recommended that batteries are changed each 5-7 year.)

2. Installation of LinPAC PXA270 SDK

The “LinPAC PXA270 SDK” is a development toolkit provided by ICP DAS, which can be used to easily develop custom applications for the LP-51xx/ 8x3x/ 8x4x embedded controller platform. The toolkit consists of the following items:

- ❑ LinPAC PXA270 SDK (GNU toolchain, Libraries, header, examples files, etc.)
- ❑ Code::Blocks project file (Windows platform only)
- ❑ Basic Linux commands (Windows platform only)

The topic provides LinPAC PXA270 SDK installation instructions for the following platforms:

- ❑ Linux
 - ◆ Download/Install LinPAC PXA270 SDK on Linux
- ❑ Windows
 - ◆ Download/Install LinPAC PXA270 SDK on Windows
 - ◆ Integrating LinPAC PXA270 SDK with Code::Blocks IDE

NOTE:

1. Start the Linux PXA270 build environment as an administrator.
2. The latest Linux PX270 SDK is integrate PXA270 series (LP-51xx/8x3x/8x4x) SDK.
3. The names of all the I/O module's API functions must begin with the prefix "I8K".
4. More detailed information, user can refer to readme.txt file here:
C:\cygwin\LinPAC_PXA270_SDK\examples\readme.txt file, or
root@LinuxPC-ICPDAS:/lincon/i8k/examples/readme.txt.

The latest version of the LinPAC PXA270 SDK (LinPAC hereinafter referred to as "LP") can be downloaded from: <ftp://ftp.icpdas.com/pub/cd/linpac/napdos/lp-8x4x/sdk/>.

Install the SDK by following the instructions below.

2.1 Quick Installation of the LinPAC PXA270 SDK

2.1.1 Download/Install SDK on Linux

1. Before installing the LinPAC PXA270 SDK, several tasks must be completed, as the root user by 'sudo' or 'su' command.
2. Insert the installation CD into your CD-ROM driver (refer to Fig.2-1 and 2-2). Locate the "linpac_pxa270_sdk_for_linux.tar.bz2" file in the \napdos\lp-8x4x\SDK\ folder, or visit the ICP DAS website to download the latest version.

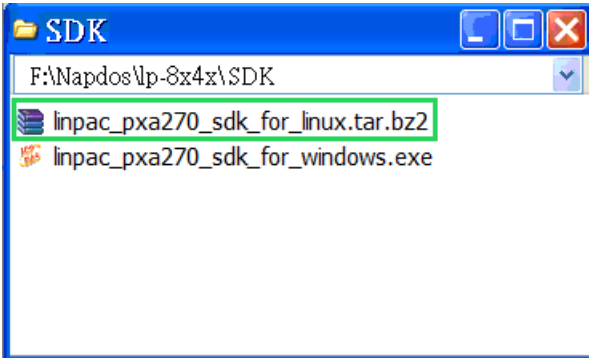


Fig. 2-1

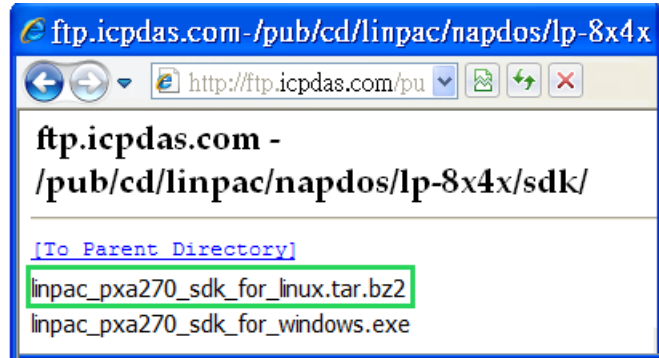


Fig. 2-2

3. Download SDK in "/" (the root directory) , and try the following command to decompress file (refer to Fig.2-3): `$ tar jxvf linpac_pxa270_sdk_for_linux.tar`

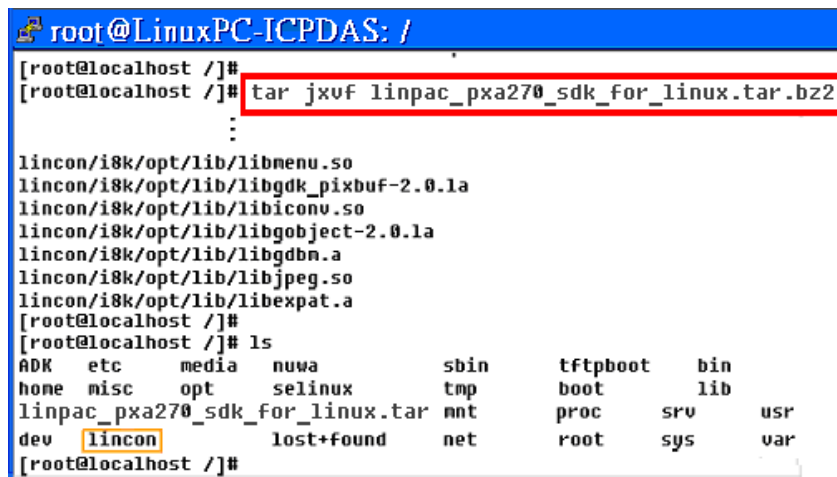


Fig. 2-3

4. Before compile the program, you need to set LinPAC PXA270 SDK path in environment variables. To execute the shell startup script and set the environment variables, enter the following command: `$./lincon/linpac.sh`

5. Type 'make' on the command line it will execute the compile command according to the Makefile (refer to Fig. 2-4).

```
root@LinuxPC-ICPDAS: /
root@LinuxPC-ICPDAS://lincon#
root@LinuxPC-ICPDAS://lincon# export |grep PATH
declare -x PATH="//lincon/tools/bin://lincon/tools/sbin:/usr/local/noweb://lincon/tools/bin://lincon/tools/sbin:/usr/local/noweb:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games"
root@LinuxPC-ICPDAS://lincon#
root@LinuxPC-ICPDAS://lincon# cd i8k/examples/
root@LinuxPC-ICPDAS://lincon/i8k/examples#
root@LinuxPC-ICPDAS://lincon/i8k/examples# ls
common  gui  i7k  i87k  i8k  java  Makefile  modbus  README  xwboard
root@LinuxPC-ICPDAS://lincon/i8k/examples#
root@LinuxPC-ICPDAS://lincon/i8k/examples# make
```

Fig. 2-4

2.1.2 Download/Install SDK on Windows

The LinPAC_PXA270_SDK_for_Windows.exe provides compilers, library, header, examples, and IDE workspace file (for Code::Blocks project). Following the step by step procedure below will help users get started.

1. Insert the installation CD into your CD-ROM driver.
2. Open the \napdos\lp-8x4x\SDK\ folder and double-click the icon for the "linpac_pxa270_sdk_for_windows.exe" file, when the Setup Wizard is displayed, click the "Next>" button to continue, refer to Fig. 2-5.
3. Click the "I accept the agreement" option and then click the "Next" button, refer to Fig. 2-6 below.

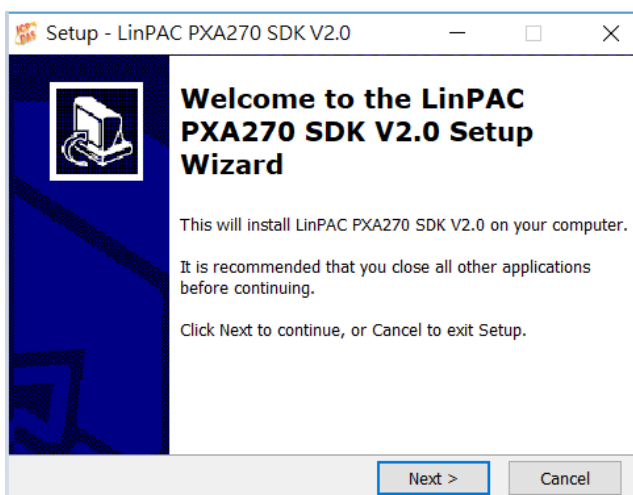


Fig. 2-5

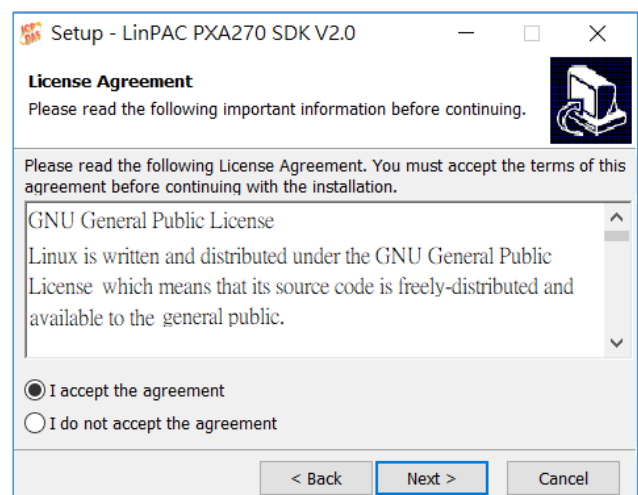


Fig. 2-6

4. The "LinPAC PXA270 SDK" files will be extracted and installed and a progress bar will be displayed to indicate the status, refer to Fig 2-7.

5. Once the software has been successfully installed, click the “Finish” button to complete the development toolkit installation, refer to Fig. 2-8.

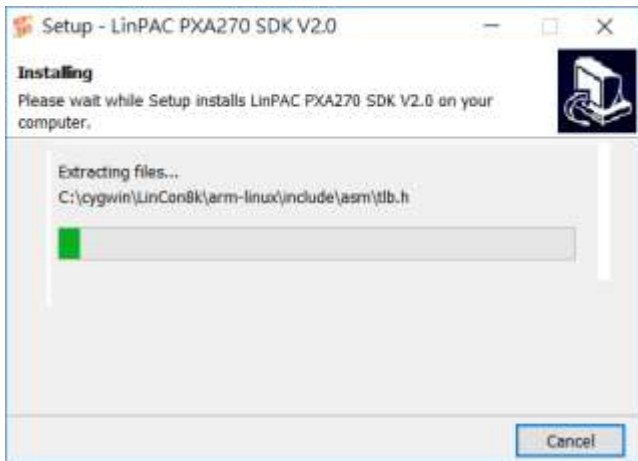


Fig. 2-7

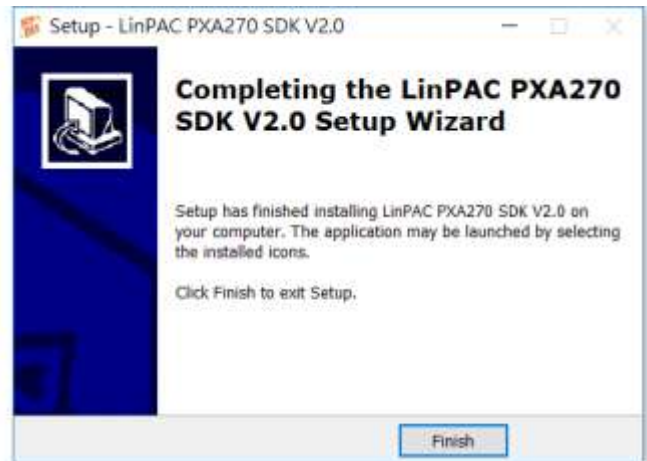


Fig. 2-8

6. Open the LinPAC PXA270 SDK installation directory, the default data directory location is “**C:\cygwin**”, user can see the contents of folder. Refer to Fig 2-9.

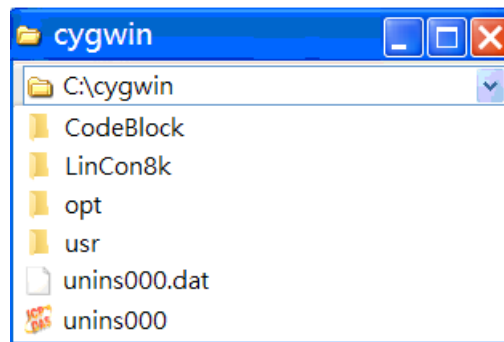


Fig. 2-9

7. Open the “**C:\cygwin\LinCon8k**” folder and see the content. Refer to Fig 2-10.

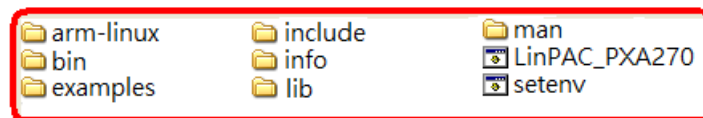


Fig. 2-10

8. From the desktop, double-click the shortcut icon for the “**LinPAC PXA270 Build Environment**” or click the “Start” > “Programs” > “ICPDAS” > “LinPAC PXA270 Build Environment”.

A Command Prompt window will then be displayed that allows applications for the LP-51xx to be compiled. Refer to Fig. 2-11.

```
LinPAC PXA270 Build Environment
C:\cygwin\LinCon8k>CMD.EXE /k c:\cygwin\LinCon8k\setenv.bat
----- LinPAC PXA270 SDK Environment Configure -----
Target      :ICPDAS LinPAC PXA270 Series (Arm based)
Work Directory :C:\Cygwin\LinCon8k
C:\cygwin\LinCon8k>
```

Fig. 2-11

9. Type “**make**” command (needs run as an administrator). A Command Prompt window will then be displayed that allows applications for the LP-51xx to be compiled. Refer to Fig. 2-12.

```
LinPAC PXA270 Build Environment
C:\cygwin\LinCon8k>CMD.EXE /k c:\cygwin\LinCon8k\setenv.bat
----- LinPAC PXA270 SDK Environment Configure -----
Target      :ICPDAS LinPAC PXA270 Series (Arm based)
Work Directory :C:\Cygwin\LinCon8k
C:\cygwin\LinCon8k>cd examples
C:\cygwin\LinCon8k\examples>ls
Makefile README common gui i7k i87k i8k java xwboard modbus
C:\cygwin\LinCon8k\examples>make
arm-linux-gcc -I. -I../include -c -o common/helloworld.o common/helloworld.c
arm-linux-gcc -I. -I../include -lm -o ./common/helloworld ./common/helloworld.o ../lib/libi8k.a
rm -f ./common/helloworld.o
arm-linux-gcc -I. -I../include -c -o common/getlist.o common/getlist.c
arm-linux-gcc -I. -I../include -lm -o ./common/getlist ./common/getlist.o ../lib/libi8k.a
rm -f ./common/getlist.o
arm-linux-gcc -I. -I../include -c -o common/read_sn.o common/read_sn.c
arm-linux-gcc -I. -I../include -lm -o ./common/read_sn ./common/read_sn.o ../lib/libi8k.a
rm -f ./common/read_sn.o
arm-linux-gcc -I. -I../include -c -o common/echosvr.o common/echosvr.c
arm-linux-gcc -I. -I../include -lm -o ./common/echosvr ./common/echosvr.o ../lib/libi8k.a
rm -f ./common/echosvr.o
arm-linux-gcc -I. -I../include -c -o common/setport.o common/setport.c
arm-linux-gcc -I. -I../include -lm -o ./common/setport ./common/setport.o ../lib/libi8k.a
```

Fig. 2-12

Once your Installation is complete, the library and demo files can be found in the following locations:

The path for the Libi8k.a and libxwboard.a file is “C:\cygwin\LinCon8k\lib”.

The path for the include files file is “C:\cygwin\LinCon8k\include”

The path for the demo file is “C:\cygwin\LinCon8k\examples”

2.1.3 Integrating SDK with Code::Blocks IDE

This tutorial gives you easy-to-follow instructions, with screenshots, for setting up a compiler (the Linaro GCC compiler), a tool that will let you turn the code that you write into programs, and Code::Blocks IDE, a free development environment. This tutorial explains how to integrate LinPAC PXA270 SDK with Code::Blocks IDE on Windows platform.

Step 1: Download Code::Blocks IDE

- ❑ Go to this website: <http://www.codeblocks.org/downloads/binaries>
- ❑ Go to the Windows 2000/ XP / Vista / 7 section, and download Windows version.

Step 2: Install Code::Block IDE

- ❑ The default install location is the C:\Program Files\CodeBlocks folder.
- ❑ A complete manual for Code::Blocks is available here:
<http://www.codeblocks.org/user-manual>

Step 3: Running in Code::Block IDE

- ❑ All files and settings that are included in a `LinPAC_PXA270_SDK` workspace file.
- ❑ Open the `C:\cygwin\CodeBlock` folder, and double click the "`LinPAC_PXA270_SDK`" as below (refer to Fig. 2-13):

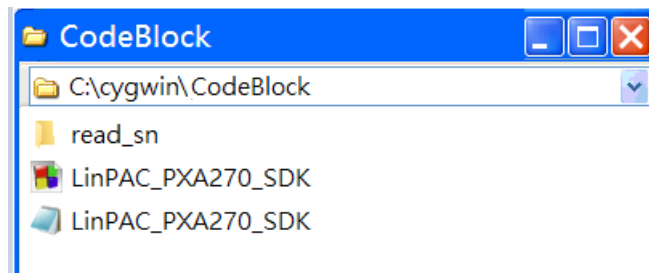


Fig. 2-13

❑ Following window will come up (refer to Fig. 2-14):

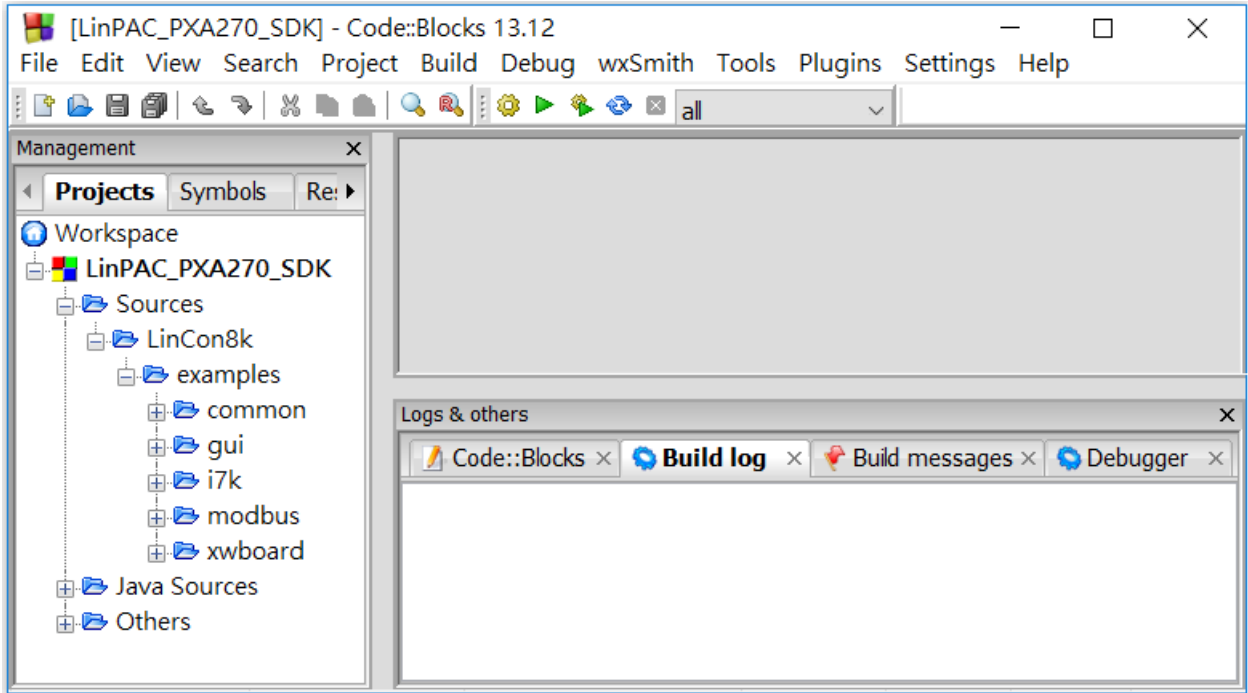


Fig. 2-14

❑ Check Compiler settings for Linaro GCC cross compiler : Click "Settings" > "Compiler" > "Toolchain executables tab" (refer to Fig. 2-15) :

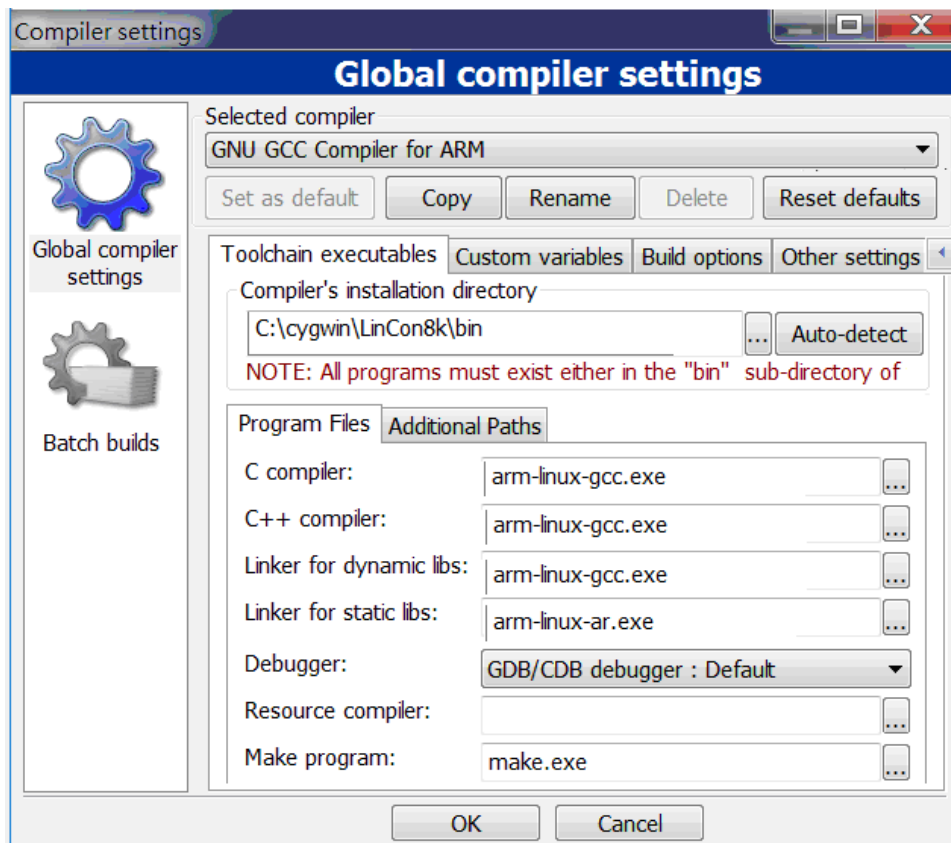


Fig. 2-15

- ❑ Check Link libraries for Linaro GCC cross compiler : Click "Settings" > "Compiler" > "Linker Settings" (refer to Fig. 2-16) :

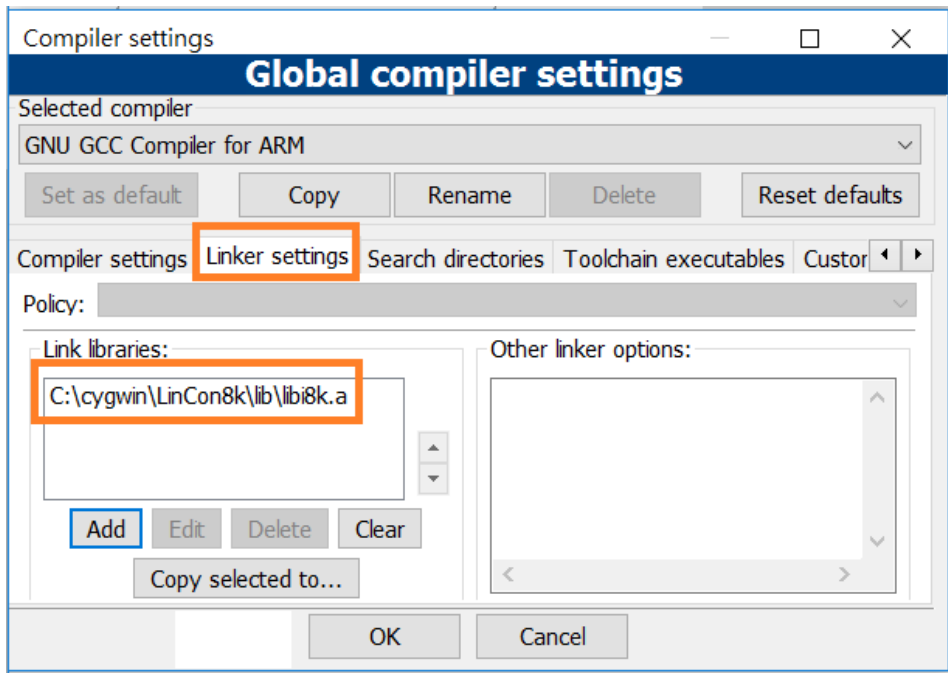


Fig. 2-16

- ❑ Check **Makefile** for Linaro GCC cross compiler : Click "Project" > "Properties" (refer to Fig. 2-17) :

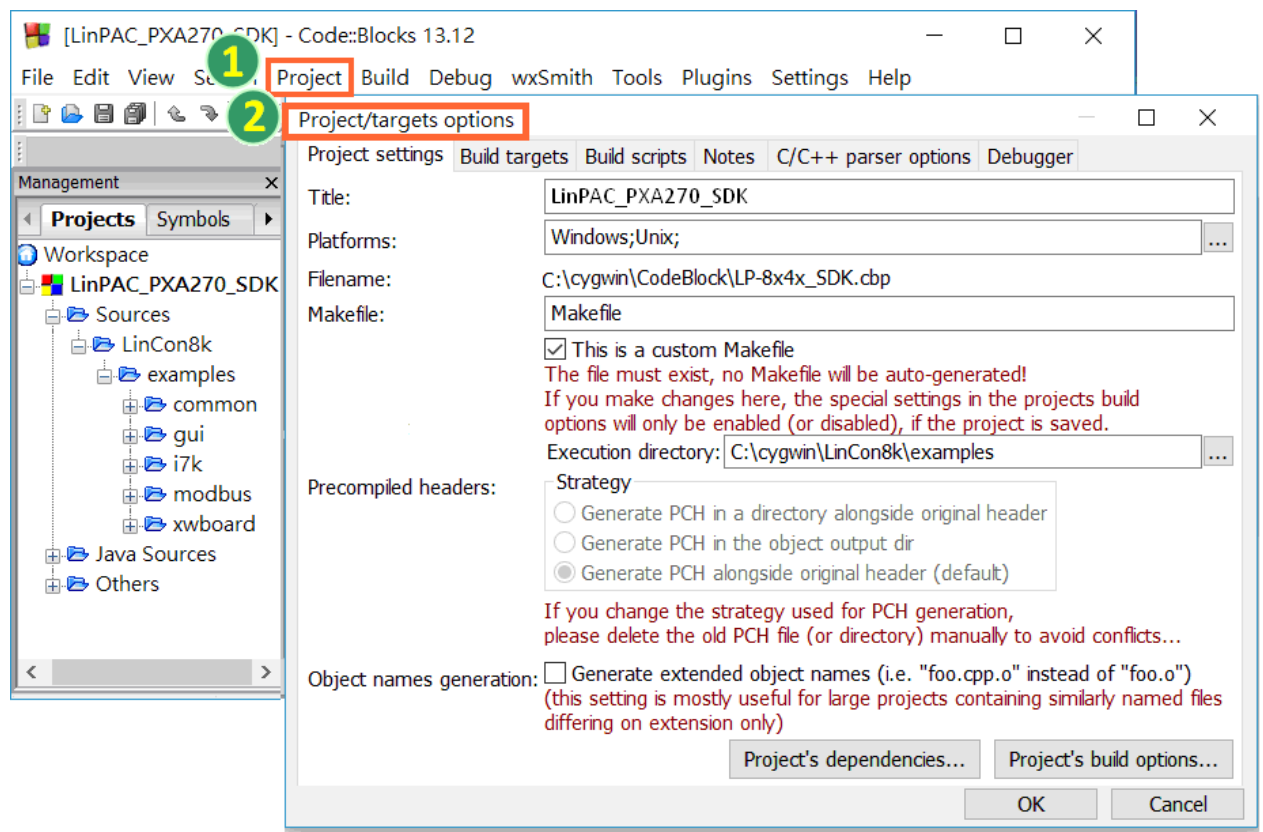


Fig. 2-17

- ❑ Click **Build** options, and it will compile the LinPAC PXA270 project completely (refer to Fig. 2-18).

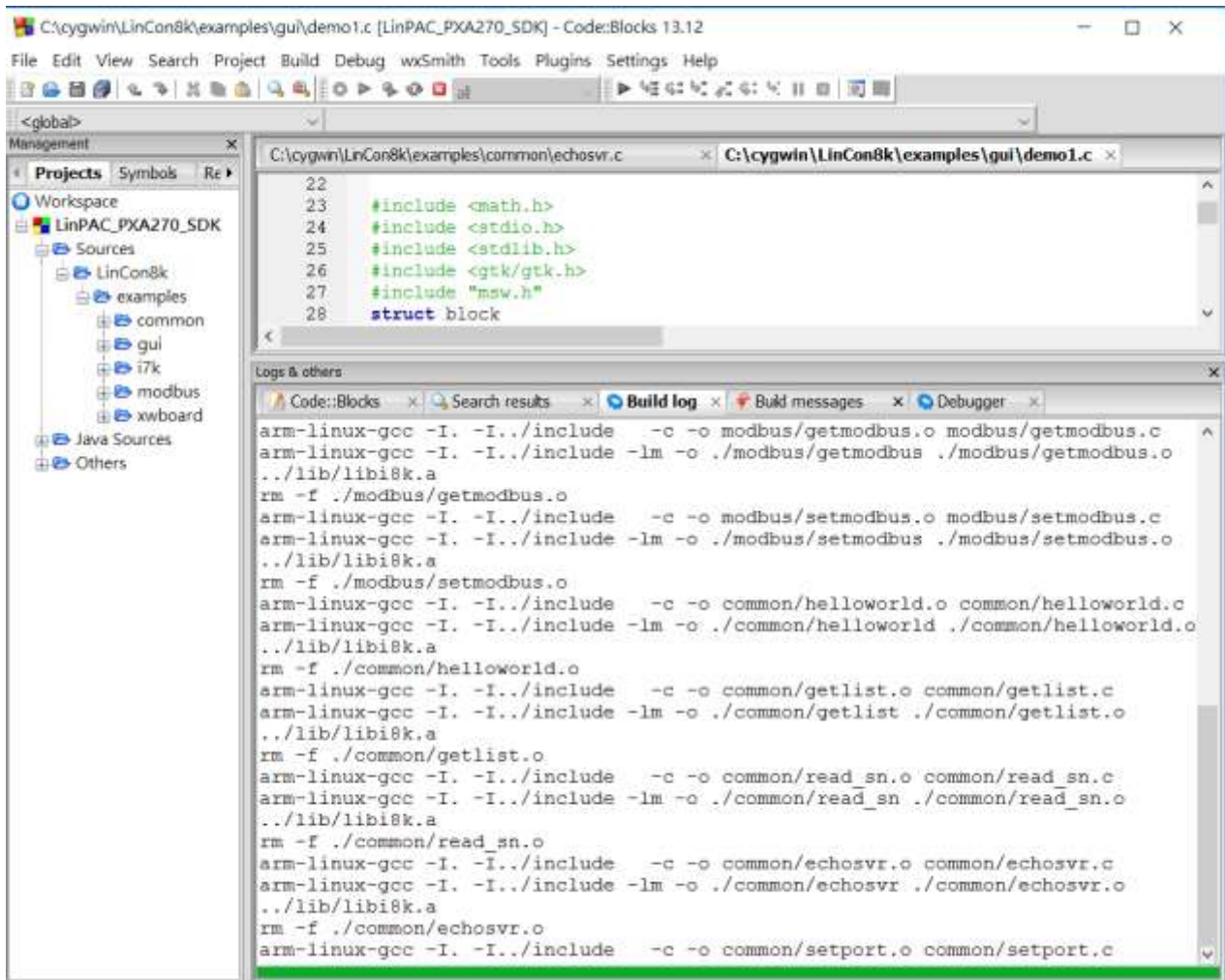


Fig. 2-18

[Note]

If you observe some characters may not display properly in cmd.exe, change the code page for the console only, do the following:

- ❑ Double-click the shortcut icon for the “**LinPAC PXA270 Build Environment**”.
- ❑ Type command: **chcp 65001** (Refer to Fig. 2-19 and Fig 2-20).



Fig. 2-19



Fig. 2-20

2.2 Introduction of the LinPAC PXA270 SDK

This section will discuss some of the techniques that are adopted in the LinPAC PXA270 SDK, including detailed explanations that describe how to easily use the SDK. The LinPAC PXA270 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 LP-51xx applications to be quickly developed. Therefore, once an application has been created, the LinPAC PXA270 SDK can be used to compile it into an executable file that can be run on the LP-51xx embedded controller.

2.2.1 Introduction to Cygwin

Cygwin is a collection of free software tools originally developed by Cygnus Solutions to allow various versions of Microsoft Windows to act somewhat like a UNIX system. Cygwin is a Linux-like environment for Windows consisting of two parts:

- (1) A DLL (cygwin1.dll) which acts as a Linux emulation layer providing substantial Linux API functionality.
- (2) A collection of tools that provide users with the Linux look and feel.

2.2.2 Introduction to Cross-Compilation

Generally, program compilation is performed by running a compiler on the build platform. The compiled program will then run on the target platform. Usually these two processes are intended for use on the same platform. However, if the intended platform is different, the process is called **cross compilation**, where source code on one platform can be compiled into executable files to be used on other platforms. For example, if the “**arm-linux-gcc**” cross-compiler is used on an x86 windows platform, the source code can be compiled into an executable file that can run on an arm-linux platform.

So why use cross compilation? In fact, cross compilation is sometimes more complicated than normal compilation, and errors are easier to make. Therefore, this method is often only employed if the program cannot be compiled on the target system, or if the program being compiled is so large that it requires more resources than the target system can provide. For many embedded systems, cross compilation is the only possible approach.

2.2.3 Download the LinPAC PXA270 SDK

- ❑ **For Windows system:** (Extract the **.exe** file into to the **C: driver**)

Download the **linpac_pxa270_sdk_for_windows.exe** file from:

<ftp://ftp.icpdas.com/pub/cd/linpac/napdos/lp-5000/lp-51xx/lp-514x/sdk/>

- ❑ **For Linux system :** (Extract the **.bz2** file into to the **root(/) directory**)

Download the **linpac_pxa270_sdk_for_linux.tar.bz2** file from:

<ftp://ftp.icpdas.com/pub/cd/linpac/napdos/lp-5000/lp-51xx/lp-514x/sdk/>

Note: We recommend user to change user ID to become **root** by 'sudo' or 'su' command.

3. The Architecture of library in the LP-51xx

The library file **libi8k.a** and **libxboard.a** are both a library file. The **libi8k.a** is designed for I7000/8000/87000 applications and **libxboard.a** is designed for I/O expansion boards. There are running on the LP-51xx Embedded Controller based on the Linux operating system and can be applied when developing custom applications **using the GNU C language**. ICP DAS provides a wide variety of demo programs that can be used to easily understand how to implement the functions and ensure that custom projects and applications can be quickly developed.

The relationships among the libi8k.a and user's applications are depicted in Fig. 3-1:

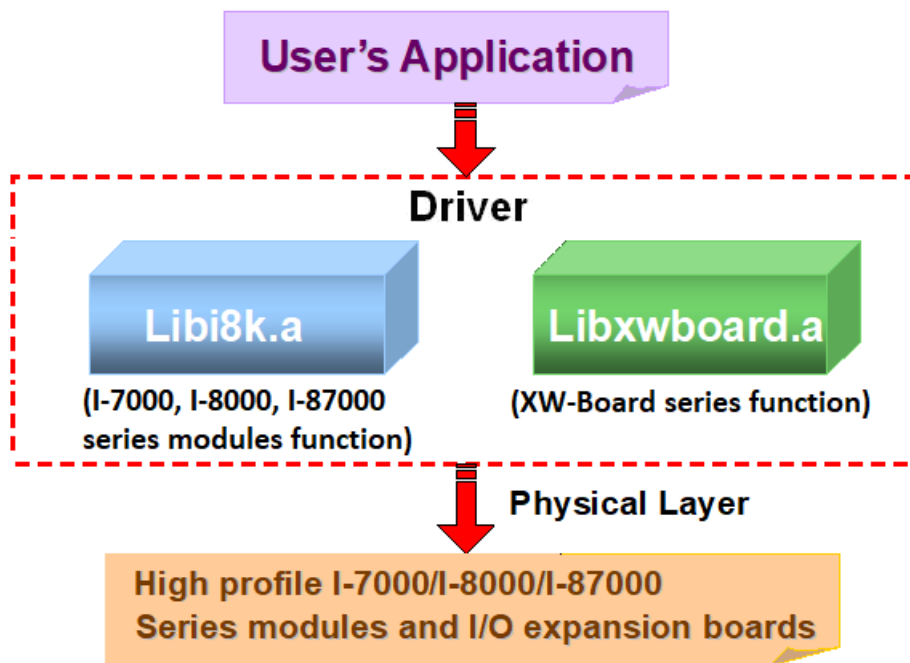


Fig. 3-1

Functions for LP-51xx Embedded Controller are divided into sub-groups for easy of use within the different applications:

1. System Information Functions
2. EEPROM Read/Write Functions
3. Watch Dog Timer Functions
4. Digital Input Functions
5. Digital Output Functions
6. Analog Input Functions
7. Analog Output Functions

The functions in the [libi8k.a](#) and [libxwboard.a](#) are specially designed for LP-51xx. For [libi8k.a](#) usage, users can easily find the functions they need for their applications from the descriptions in Chapter 6 and in the demo programs provided in Chapter 7. Another driver-[libxwboard.a](#), users can refer to [LP-51xx_xwboard_user_guide.pdf](#).

4. LP-51xx System Settings

The following is a guide to easily configuration the LP-51xx.

4.1 LP-51xx Network Settings

There are two methods of configuring the network settings for the LP-51xx. The first uses **DHCP** and the second is based on an “**Assigned IP**”. The factory default configuration is DHCP. However, if a DHCP server is not available on the network system, then the network settings need to be configured using the “Assigned IP” method.

4.1.1 Configuring the IP, Netmask and Gateway Addresses

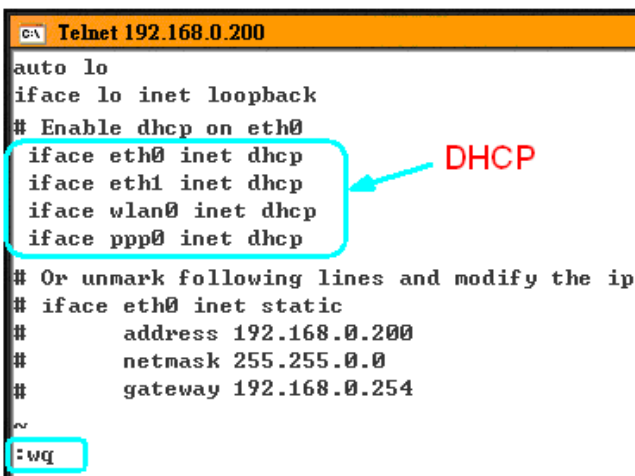
Boot up LP-51xx and click the “**start/xterm**” to open a “command line”. Type “**vi /etc/network/interfaces**” to open the network setting file.

(1) Using DHCP:

Once the network settings file is loaded, remove the “#” comment from each line in the dhcp block and add comment out the Assign IP block by adding “#” to each entry. Type “**:wq**” to save the changes, and then type “**ifup eth0**” to activate the new settings. (Refer to the Fig. 4-1 for more details)

(2) Using “Assigned IP”:

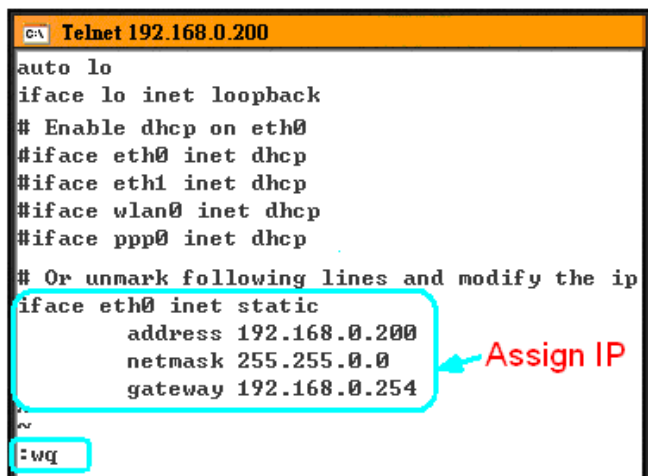
Once the network settings file is loaded, remove the “#” comment from each line in the Assign IP block and comment out the dhcp block by adding “#” to each entry. Entry the relevant IP, Netmask and Gateway details in the respective Assign IP block entries. Type “**:wq**” to save the changes, and the type “**ifup eth0**” to activate the new settings. (Refer to the Fig. 4-2)



```
C:\ Telnet 192.168.0.200
auto lo
iface lo inet loopback
# Enable dhcp on eth0
iface eth0 inet dhcp
iface eth1 inet dhcp
iface wlan0 inet dhcp
iface ppp0 inet dhcp

# Or unmark following lines and modify the ip
# iface eth0 inet static
#     address 192.168.0.200
#     netmask 255.255.0.0
#     gateway 192.168.0.254
~
:wq
```

Fig. 4-1

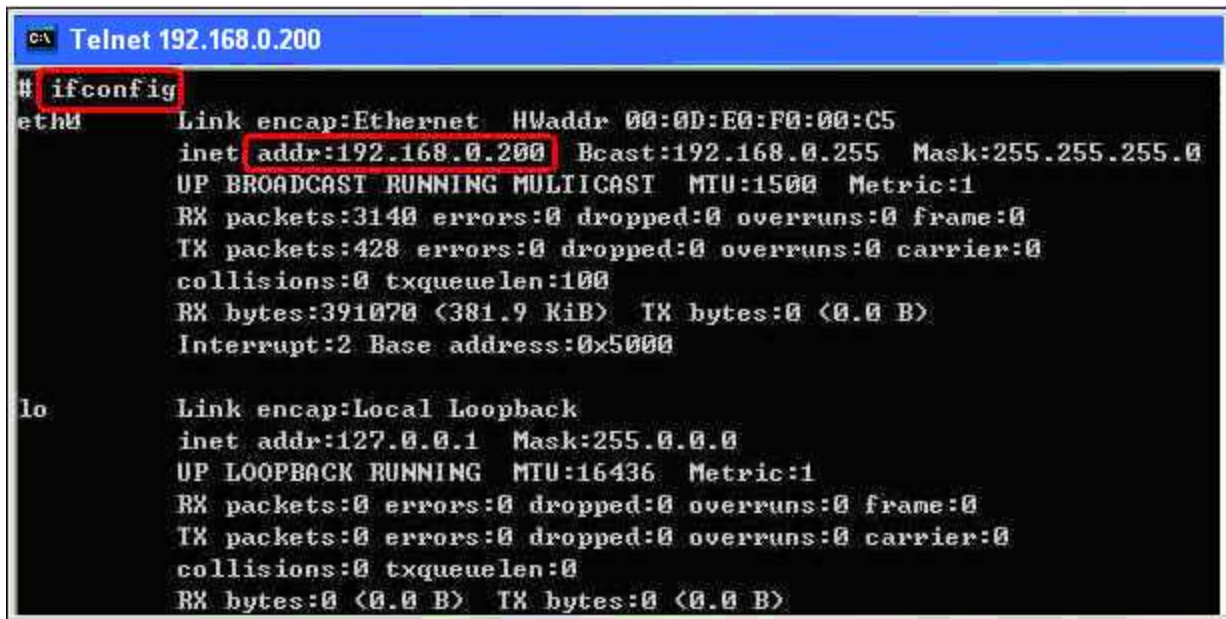


```
C:\ Telnet 192.168.0.200
auto lo
iface lo inet loopback
# Enable dhcp on eth0
#iface eth0 inet dhcp
#iface eth1 inet dhcp
#iface wlan0 inet dhcp
#iface ppp0 inet dhcp

# Or unmark following lines and modify the ip
iface eth0 inet static
    address 192.168.0.200
    netmask 255.255.0.0
    gateway 192.168.0.254
~
:wq
```

Fig. 4-2

After configuring the LinPAC network settings, type “**ifconfig**” to verify that the network settings are correct. Refer to the Fig. 4-3 for more details.



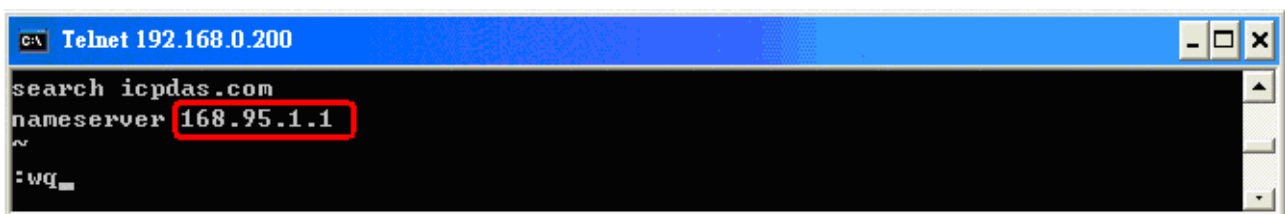
```
C:\ Telnet 192.168.0.200
# ifconfig
eth0  Link encap:Ethernet  HWaddr 00:0D:E0:F0:00:C5
      inet addr:192.168.0.200  Bcast:192.168.0.255  Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:3140 errors:0 dropped:0 overruns:0 frame:0
      TX packets:428 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:100
      RX bytes:391070 (381.9 KiB)  TX bytes:0 (0.0 B)
      Interrupt:2 Base address:0x5000

lo    Link encap:Local Loopback
      inet addr:127.0.0.1  Mask:255.0.0.0
      UP LOOPBACK RUNNING  MTU:16436  Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Fig. 4-3

4.1.2 DNS Setting

Boot the LP-51xx and click the “**start/ xterm**” to open a “command line”. Type “**vi /etc/resolv.conf**” to open the DNS setting file. Once the DNS settings file is loaded, type “DNS server” in the “**nameserver**” field. Type “**:wq**” to save the changes, and type “**reboot**” to reboot the LP-51xx so that the new settings can take effect. Refer to the Fig. 4-4 for more details.



```
C:\ Telnet 192.168.0.200
search icpdas.com
nameserver 168.95.1.1
~
:wq
```

Fig. 4-4

4.2 Usage a microSD Card

Files contained on a mounted microSD card can be accessed from the **/mnt/hda** directory (Refer to Fig. 4-5).

```
# mount
rootfs on / type rootfs (rw)
/dev/root on / type jffs2 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
tmpfs on /var type tmpfs (rw)
/dev/mmcblk0p1 on /mnt/hda type vfat (rw,mask=0022,dmask=0022,codepage=cp437,ioccharset=iso8859-1)
/dev/ram0 on /mnt/ramfs type minix (rw)
#
```

Fig. 4-5

When using the microSD card, be sure to pay attention to the following notes:

1. Umount the microSD card before removing it.
2. Do not power off or reboot the LP-51xx while data is being written to or read from the microSD card.
3. The microSD memory must be formatted in the VFAT/EXT2/EXT3 file system.

4.2.1 Mounting a microSD Card

To use a microSD card, insert the microSD card into the socket in the LP-51xx (Refer to Fig. 1-1), and it will be automatically mounted when the LP-51xx is booted. The files of microSD card can then be accessed from the **/mnt/hda** directory.

If the card is not mounted automatically, type in “**/etc/init.d/sd start**”, to mount it.

4.2.2 Unmounting the microSD Card

Before removing the microSD card from the LP-51xx, unmount the card by entering the following steps:

- (1) **/etc/init.d/apachect1 stop**
- (2) **/etc/init.d/startx stop**
- (3) **umount /mnt/hda**

The microSD card can then be safely removed to prevent damage to the card.

4.2.3 Scanning and repairing microSD Card

After the LP-51xx is booted, the microSD card will be named “/dev/mmcblk0p1“. It is recommended that the microSD card is **unmounted** first before attempting to perform a scan or repair.

- ❑ **blockdev**: this command is used to call block device ioctls from the command line.

Parameter	Description	Example
--report	print a report for device	blockdev --report /dev/mmcblk0p1
-v --getra --getbz	get readhead and blocksize	blockdev -v --getra --getbz /dev/mmcblk0p1

- ❑ **fsck.minix**: this command is used to perform a consistency check for the Linux MINIX filesystem.

Parameter	Description	Example
-r	performs interactive repairs	fsck.minix -r /dev/mmcblk0p1
-s	outputs super-block information	fsck.minix -s /dev/mmcblk0p1

- ❑ **fsck.vfat**: this command is used to check and repair MS-DOS file systems.

Parameter	Description	Example
-a	automatically repair the file system	fsck.vfat -a /dev/mmcblk0p1
-l	list path names of files being processed	fsck.vfat -l /dev/mmcblk0p1

- ❑ **mkfs**: this command is used to build a Linux file system on a device, usually a hard disk partition.

Parameter	Description	Example
-t	specifies the type of file system to be built	mkfs -t vfat /dev/mmcblk0p1
-c	check the device for bad blocks before building the file system	mkfs -c vfat /dev/mmcblk0p1

- ❑ **mkfs.minix**: this command is used to make a MINIX filesystem

Parameter	Description	Example
	create a Linux MINIX file-system	mkfs.minix /dev/mmcblk0p1
-c	check the device for bad blocks before building the file system	mkfs.minix -c /dev/mmcblk0p1

- ❑ **mkfs.vfat**: this command is used to make an MS-DOS filesystem

Parameter	Description	Example
-A	use Atari variation of the MS-DOS filesystem	mkfs.vfat -A /dev/mmcbk0p1
-v	verbose execution	mkfs.vfat -v /dev/mmcbk0p1

- ❑ **mke2fs.ext2**: this command is used to make an EXT2 filesystem

Parameter	Description	Example
-t	create a Linux EXT2 file-system	mke2fs.ext2 -t ext2 /dev/mmcbk0p1

- ❑ **mke2fs.ext3**: this command is used to make an EXT3 filesystem

Parameter	Description	Example
-t	create a Linux EXT3 file-system	mke2fs.ext2 -t ext3 /dev/mmcbk0p1

4.3 Using a USB Storage Device

USB storage devices are not automatically mounted to the LP-51xx, so it must be manually mounted before attempting to access the USB storage device.

4.3.1 Mounting a USB Storage Device

To mount a USB storage devices follow the procedure described below:

- (1) Type “**mkdir /mnt/usb**” to create a directory named “usb”.
- (2) Type “**mount /dev/sda1 /mnt/usb**” to mount the USB storage device to the usb directory and then type “**ls /mnt/usb**” to view the contents of the USB storage device. (Refer to Fig. 4-6)

```
# mkdir /mnt/usb
#
# cat /proc/diskstats | grep sda*
 8  0 sda 37 62 106 260 0 0 0 0 254 260
 8  1 sda1 98 98 0 0
#
# mount /dev/sda1 /mnt/usb
# mount
rootfs on / type rootfs (rw)
/dev/root on / type jffs2 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
tmpfs on /var type tmpfs (rw)
shmfs on /dev/shm type tmpfs (rw)
usbfs on /proc/bus/usb type usbfs (rw)
/dev/mmcblk0p1 on /mnt/hda type vfat
(rw,umask=0022,dmasks=0022,codepage=cp437,ioccharset=iso8859-1)
/dev/ram0 on /mnt/ramfs type minix (rw)
/dev/sda1 on /mnt/usb type vfat
(rw,umask=0022,dmasks=0022,codepage=cp437,ioccharset=iso8859-1)
#
# ls /mnt/usb
0429.doc  2009.avi
```

Fig. 4-6

4.3.2 Unmounting a USB Storage Device

Before removing the USB storage device from the LP-51xx, the device must be unmounted to prevent any damage to the device. To unmount the device, type the “**umount /mnt/usb**” command and then remove the USB storage device.

4.4 Adjusting the VGA Resolution

The LinPAC supports two VGA resolutions -- **640x480** and **800x600**, and the **default setting is 800x600**. To change the VGA resolution, follow the procedure described below:

- (1) Type “**vi /etc/init.d/fbman**” to open the VGA resolution configuration file.
- (2) To set the resolution to be 640x480, comment out the reference to 800x600 by adding “**#**” to the entry, and then remove the “**#**” comment from the 640x480 entry. (Refer to Fig. 4-7)

- ❑ For example, to set the resolution to 800x600, open the file “/etc/init.d/fbman”, the code should be as follows:

```
#/usr/sbin/fbset -n 640x480-60
  /usr/sbin/fbset -n 800x600-70
```


- ❑ To change the resolution to 640x480, the code should be as follows:

`/usr/sbin/fbset -n 640x480-60`

`#!/usr/sbin/fbset -n 800x600-70`

```
start)
    echo -n "Setting framebuffer ..."
    #/usr/bin/clear
    /usr/sbin/fbset -n 640x480-60
    #/usr/sbin/fbset -n 800x600-70
    EXITCODE=0
    ;;
stop)
    echo -n "Restore framebuffer ..."
    echo "done."
    EXITCODE=0
    ;;
restart)
    $0 stop
    $0 start
    EXITCODE=$?
    ;;
*)
    usage
    ;;
esac
:wq
```

Fig. 4-7

- (3) Type “**:wq**” to save the changes, and then type “**shutdown -r now**” to reboot the LP-51xx, so that the new settings can take effect. (Refer to Fig. 4-8)

```
# fbset
mode "640x480-60"
    # D: 26.000 MHz, H: 31.401 kHz, V: 59.926 Hz
    geometry 640 480 640 480 16
    timings 38461 78 46 22 10 64 12
    accel false
    rgba 5/11,6/5,5/0,0/0
endmode
#
```

Fig. 4-8

4.5 Automatically executing applications at startup

A “run level” is an operation mode that is used to determine which programs are executed during system startup. The default run level for the LP-51xx is level **2**.

The run level file is located in the `/etc/init.d` directory and contains the scripts that will be executed at boot time. These scripts are referenced by creating symbolic links in the `/etc/rc2.d` directory.

The format for the naming of these links is named S<2-digit-number><original-name>. The 2-digit number determines the order in which the scripts are executed. The valid range is from 00 to 99 and the lower numbered file will be executed earlier. Scripts prefixed with an **S** are called at startup, and those prefixed with either a **K** or an **x** are called when the system is closed or shut down.

4.5.1 Configuring a program to run at boot time

To configure a program to run at boot time, create a startup script that runs the required commands to be automatically executed then save it in the “/etc/init.d” directory. The script must then be symbolically linked to the “/etc/rc2.d” directory.

The procedure for creating a script is described below:

- (1) Create a script with the filename “hello” by typing “**vi /etc/init.d/hello**“. This will allow the script that executes the programs edit. Type “**:wq**“ to save the script and quit. (Refer to the Fig. 4-9)

Note: If necessary, the **PATH** and **LD_LIBRARY_PATH** environment variable should be included in the script. Check the “/etc/init.d/webcam” file for an example. Refer to the Fig. 4-10 for more details.

```
# cat /etc/init.d/webcam
#!/bin/sh
#
# Sample start-up script
#
export LOGNAME=root
export HOME=$LOGNAME
export LD_LIBRARY_PATH=/opt/X11R6/lib:/opt/lib:/usr/local/lib:/lib:/usr/lib:/opt/kaffe/jre/lib/arm
:/opt/php/lib:/opt/mysql/lib:/opt/apache2/lib:/etc/user/lib:/mnt/hda/opt/lib:/opt/local/lib
export PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:/opt/X11R6/bin:/opt/bin:/
opt/kaffe/bin:/opt/php/bin:/opt/mysql/bin:/opt/apache2/bin:/etc/user/bin:/etc/user/sbin:/opt/local
/bin:/opt/local/sbin:/mnt/hda/opt/bin:/opt/sbin:.
export CLASSPATH=/opt/kaffe/lib/kjc.jar:/opt/kaffe/lib/icpdas.jar:/opt/kaffe/lib/swingall.jar:.
export JAVA_HOME=/opt/kaffe
```

Fig. 4-9

- (2) Type “**chmod 755 /etc/init.d/hello**“ to change the access permissions for the file.
- (3) Type “**cd /etc/rc2.d**“ to change directory to the default run level.
- (4) Type ” **ln -s ../init.d/hello /etc/rc2.d/S85hello** “ to create a symbolic link to the script file so that it will be automatically executed at boot time. Refer to the Fig. 4-11 for more details.

```

#?/bin/sh ← For declaring
#
# ICPDAS LinCon-8000 daemon
# /etc/init.d/hello 0.1 2004/05/025 < moki matsushima >
usage()
{
    echo "Usage: $0 {start|stop|restart}"
}
EXITCODE=1
for x in "1" ; do
    if [ $# -lt 1 ] ; then usage ; break ; fi
    action=$1

    case "$action" in
    start)
        echo -n "Starting Hello services: "
        echo "Welcome to LinCon-8000!" ← Running at boot time.
        EXITCODE=0
        ;;
    stop)
        echo -n "Shutting down hello services: "
        echo "done."
        EXITCODE=0
        ;;
    restart)
        $0 stop
        $0 start
        EXITCODE=$?
        ;;
    *)
        usage
        ;;
    )
}
-wq ← Save and Quit

```

Fig. 4-10

```

# cd /etc/rc2.d ← run level
# ls
S09pppslip      S20ssh          S60snmp         S80hwclock      S99rmnologin   xS47ipsec
S10pcmcia       S40inetd        S70slot         S97fbman        xS04sd          xS72Ramdriver
S11ifupdown     S50apache       S71Serial       S98Xserver      xS20apmd
#
# ln -s ../init.d/hello /etc/rc2.d/S85hello ← Making a symbolic link
# ls -al
drwxr-xr-x 1 root root 0 Jul 23 17:36 .
drwxr-xr-x 1 root root 0 Jul 12 16:50 ..
lrwxrwxrwx 1 root root 17 Sep 12 2005 S09pppslip -> ../init.d/pppslip
lrwxrwxrwx 1 root root 16 Sep 12 2005 S10pcmcia -> ../init.d/pcmcia
lrwxrwxrwx 1 root root 18 Sep 12 2005 S11ifupdown -> ../init.d/ifupdown
lrwxrwxrwx 1 root root 13 Sep 12 2005 S20ssh -> ../init.d/ssh
lrwxrwxrwx 1 root root 15 Sep 12 2005 S40inetd -> ../init.d/inetd
lrwxrwxrwx 1 root root 19 Sep 12 2005 S50apache -> ../init.d/apachectl
lrwxrwxrwx 1 root root 14 Sep 12 2005 S60snmp -> ../init.d/snmp
lrwxrwxrwx 1 root root 14 Sep 12 2005 S70slot -> ../init.d/slot
lrwxrwxrwx 1 root root 16 Sep 12 2005 S71Serial -> ../init.d/serial
lrwxrwxrwx 1 root root 20 Sep 12 2005 S80hwclock -> ../init.d/hwclock.sh
lrwxrwxrwx 1 root root 15 Jul 23 17:36 S85hello -> ../init.d/hello ← OK
lrwxrwxrwx 1 root root 15 Sep 12 2005 S97fbman -> ../init.d/fbman
lrwxrwxrwx 1 root root 16 Jul 23 12:36 S98Xserver -> ../init.d/startx
lrwxrwxrwx 1 root root 19 Oct 30 2006 S99rmnologin -> ../init.d/rmnologin
lrwxrwxrwx 1 root root 12 Sep 12 2005 xS04sd -> ../init.d/sd
lrwxrwxrwx 1 root root 14 Sep 12 2005 xS20apmd -> ../init.d/apmd
lrwxrwxrwx 1 root root 15 Sep 12 2005 xS47ipsec -> ../init.d/ipsec
lrwxrwxrwx 1 root root 18 Sep 12 2005 xS72Ramdriver -> ../init.d/ramdrive

```

Fig. 4-11

4.5.2 Disabling a program from running at boot time

The procedure for disabling a script is described to the default run level.

- (1) Type “ **cd /etc/rc2.d** “ to change directory to the default run level.
- (2) Type “ **mv S85hello xS85hello** “ to rename the S85hello symbolic link and prevent the program from automatically executing at boot time.

4.6 Automatic login

Automatic Login allows a specified user to log into the console (normally /dev/tty1) when the system is first booted without displaying a prompt requesting a username or password. This is achieved using the **mingetty** command, and the setup procedure is as follows:

- (1) Login as root and open the “**/etc/inittab**” file for the LP-51xx.
- (2) Modify the entry for the first terminal— **tty1**, as shown in Fig. 4-12, After rebooting the LP-51xx, it will automatically login to the root account.

```
# /sbin/getty invocations for the runlevels.
#
# The "id" field MUST be the same as the last
# characters of the device (after "tty").
#
# Format:
# <id>:<runlevels>:<action>:<process>
#
1:2345:respawn:/sbin/mingetty -noclear --autologin root tty1
#1:2345:respawn:/sbin/getty 38400 tty1
2:2345:respawn:/sbin/getty 38400 tty2
3:2345:respawn:/sbin/getty 38400 tty3
4:2345:respawn:/sbin/getty 38400 tty4
5:2345:respawn:/sbin/getty 38400 tty5
6:2345:respawn:/sbin/getty 38400 tty6
```

Fig. 4-12

5. Instructions for the LP-51xx

This section provides an introduction to some of the more commonly used Linux instructions. These Linux instructions are similar to those used in DOS and are generally expressed in **lower case** letters.

5.1 Basic Linux Instructions

5.1.1 ls: lists the file information (Equivalent DOS Command: dir)

Parameter	Description	Example
-l	Lists detailed information related to the files	ls -l
-a	Lists all files, including hidden files	ls -a
-t	Lists the files arranged in date/time order	ls -t

5.1.2 cd directory: Changes directory (Equivalent DOS Command: cd)

Parameter	Description	Example
..	Move to the parent directory	cd ..
~	Move back to the root directory	cd ~
/	Path component separator	cd /root/i8k

5.1.3 mkdir: creates a subdirectory (Equivalent DOS Command: md)

Parameter	Description	Example
-p	No error if the subdirectory exists, and creates the parent directories as needed	mkdir -p directory

5.1.4 rmdir: deletes the subdirectory which must be empty (Equivalent DOS Command: rd)

Parameter	Description	Example
-p	Removes the specified DIRECTORY, then attempts to remove each parent directory component with the same path name	rmdir -p directory

5.1.5 passwd: used to change the password

5.1.6 reboot: reboots the LinPAC (or use 'shutdown -r now')

5.1.7 rm: deletes (removes) the file or directory (Equivalent DOS Command: delete)

Parameter	Description	Example
-i	Displays a warning message before deleting	rm -i test.exe
-r	Deletes the directory even if it isn't empty	rm -r test.exe
-f	No warning message displayed when deleting	rm -f test.exe

5.1.8 cp: copies one or more files (Equivalent DOS Command: copy)

Parameter	Description	Example
-R	Performs a recursive copy	cp -R test bak
-i	Displays a confirmation prompt before overwriting	cp -i test bak
-l	Links the files instead of copying them	cp -l test bak

5.1.9 mv: moves or renames a file or directory (Equivalent DOS Command: move)

Parameter	Description	Example
-f	Does not display a confirmation prompt before overwriting	cp -f sour des
-i	Displays a confirmation prompt before overwriting	cp -i /sour /des

5.1.10 pwd: displays the full path of the current working directory

5.1.11 who: displays a list of the users current logged on

5.1.12 chmod: changes the access permissions for a file

Syntax → chmod ??? file → ??? means owner: group: all users

For example: chmod 754 test.exe

7 5 4 → 111(read, write, execute)

101(read, write, execute)

100(read, write, execute)

The first number 7: the **owner** can read and write and execute files

The second number 5: the **group** can only read and execute files

The third number 4: **all users** can only read files

5.1.13 uname: displays the Linux version information

5.1.14 ps: displays a list of the currently active procedures

5.1.15 ftp: transfers a file using the file transfer protocol (FTP)

ftp IPAddress (Example: ftp 192.168.0.200 → connect to ftp server)

! : temporarily exits the FTP

exit : back to the ftp

bin : transfers files in “binary” mode

get : downloads a file from the LinPAC to the Host (For example: get /mnt/hda/test.exe
c:/test.exe)

put : uploads a file from the host to the LinPAC (For example: put c:/test.exe
/mnt/hda/test.exe)

bye : exits FTP

5.1.16 telnet: establishes a connection to another PC via Telnet terminal

Syntax: telnet IPAddress

For example telnet 192.168.0.200 (will allow remote control of the LP-51xx)

5.1.17 date: prints or sets the system date and time

5.1.18 hwclock: queries and sets the hardware clock (RTC)

Parameter	Description	Example
-r	Reads the hardware clock and prints the time on a standard output.	hwclock -r
-w	Sets the hardware clock to the current system time	hwclock -w

5.1.19 netstat: displays the current state of the network

Parameters: [-a]: list all states (For example: netstat -a)

5.1.20 ifconfig: displays the ip and network mask information (Equivalent DOS

Command: ipconfig)

5.1.21 ping: used to test whether the host in a network is reachable

Syntax: ping IPAddress

For example ping 192.168.0.1

5.1.22 clear: clears the screen

5.2 General GCC Instructions

The GNU Compiler Collection (GCC) is a compiler system developed by the collaborative GNU Project that can be used to compile source code written using either ANSIC or Traditional C into executable files. Executable files compiled using GCC can be executed within the different operating system and different hardware systems. The GCC is distributed by the Free Software Foundation and is free of charge. Consequently, it has become very popular with users of Unix-based systems.

Fig. 5-1 below provides an illustration of the compilation procedure within Linux.

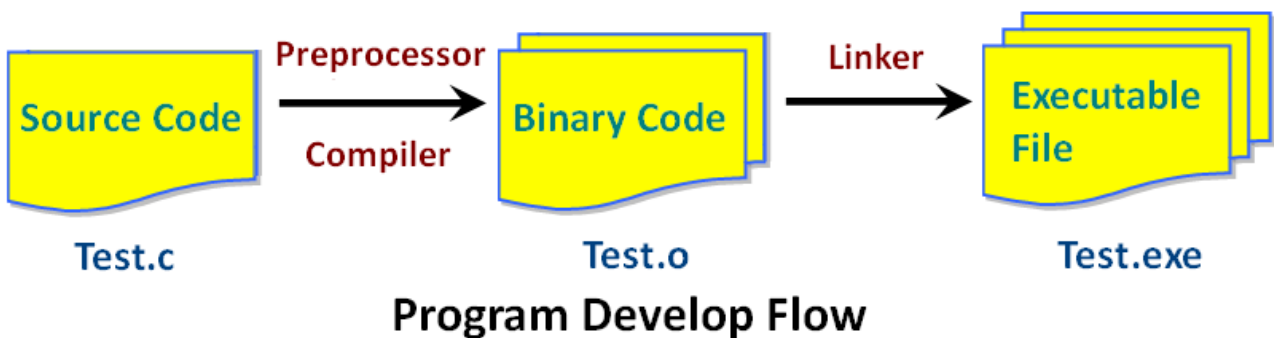


Fig. 5-1

The following is an overview of some of the common GCC instructions that will provide a guide for compiling *.c files to *.exe applications together with an explanation of the parameters used by the GCC during the compilation process.

5.2.1 Compile without linking to the LP-51xx library

❑ **Purpose:** *.c to *.exe

Command: arm-linux-gcc -o target source.c

Parameter:

-o target: assigns the name of output file

source.c: the C source code file

Example: arm-linux-gcc -o helloworld.exe helloworld.c

Output File: helloworld.exe

```
LinPAC-5000 Build Environment
C:\cygwin\LinCon8k\examples\common>arm-linux-gcc helloworld.c -o helloworld.exe
C:\cygwin\LinCon8k\examples\common>
```

Fig. 5-2

5.2.2 Compile by linking to the LP-51xx library (libi8k.a)

(1) Purpose: *.c to *.o

Command: arm-linux-gcc -lincludeDIR -lm -c -o target source.c library

Parameter:

-lincludeDir: the path to the include files

-lm: includes the math library (libm.a)

-c: only compile *.c to *.o (object file)

-o target: assigns the name of output file

source.c: the C source code file

library: the path of library

Example: arm-linux-gcc -I. -I./include -lm -c -o test.o test.c ../lib/libi8k.a

Output File: test.o

(2) Purpose: *.o to *.exe

Command: arm-linux-gcc -lincludeDIR -lm -o target source.o library

Parameter:

-lincludeDir: the path of include files

-lm: includes the math library (libm.a)

-o target: assigns the name of the output file

source.o: the object file

library: the path to the library

Example: arm-linux-gcc -I. -I./include -lm -o test.exe test.o ../lib/libi8k.a

Output File: test.exe

(3) Purpose: *.c to *.exe

Command: arm-linux-gcc -lincludeDIR -lm -o target source.c library

Parameter:

-lincludeDir: the path of include files

-lm: include math library (libm.a)

-o target: assign the name of output file

source.c: source code of C

library: the path of library

Example: arm-linux-gcc -I. -I./include -lm -o test.exe test.c ../lib/libi8k.a

Output File: test.exe

5.3 A Simple Example – Helloworld.c



This section describes how to 1) compile the helloworld.c file to the helloworld.exe executable file, 2) transfer the executable file (helloworld.exe) to the LP-51xx using FTP, and 3) execute this file via the Telnet Server on the LP-51xx.

This process can be accomplished using a single PC without requiring an additional monitor for the LP-51xx. No ICP DAS modules are used in this example. However to use ICP DAS modules to control the system, refer to the demo described in chapter 7.

The process can be divided into three steps, which are described below:

Step 1: Compile helloworld.c to helloworld.exe

(1) Open LP-51xx SDK Build environment (refer to step 8 in section 2.1) and then type "**cd examples/common**" to change the path to **C:/cygwin/LinCon8k/examples/common**. Type "**dir/w**" and to display the contents of the directory and confirm that the helloworld.c file is present. Refer to Fig. 5-3 for more details.

```
LinPAC-51XX Build Environment
C:\cygwin\LinCon8k> cd examples\common
C:\cygwin\LinCon8k\examples\common> dir/w
Volume in drive C has no label.
Volume Serial Number is 649C-FEE2

C:\cygwin\LinCon8k\examples\common
[.]
back_plane_id.exe      dip_switch.c          dip_switch.exe        back_plane_id.c
echosvr.exe            eeprom.c              eeprom.exe            echosvr.c
getdo_rb.exe           [getexai              getexai.c             getdo_rb.c
getexai.rar            getexdi.c             getexdi.exe           getexai.exe
getlist.exe            getport.c             getport.exe           getlist.c
getreceive.exe         getsendreceive.c     getsendreceive.exe    getreceive.c
getsendreceive_bin.exe helloworld.c          port.exe              getsendreceive_bin.c
led.exe                port.c                read_sn.exe           led.c
read_sn-test.exe      read_sn.c             read_sn.exe           read_sn-test.c
rotary_id.exe          send_receive.c        send_receive.exe     rotary_id.c
serial_test.exe        setdo_bw-ok.c         setdo_bw.c            serial_test.c
setexao.c              setexao.exe           setexdo.c             setdo_bw.exe
setport.c              setport.exe           setsend.c             setexdo.exe
slot_count.c           slot_count.exe        wdt_safe_value.c     setsend.exe
sramok.exe             timer.c               wdt.c                 wdt_safe_value.exe
timer2.exe             uart.c                sram-read.c          wdt.exe
timer.exe              timer2.c              sram.c               sram-read.exe
                      sram.c               sram.exe
```

Fig. 5-3

- (2) Type the command “**arm-linux-gcc -o helloworld.exe helloworld.c**” to compile helloworld.c into helloworld.exe, then type “**dir/w**” to display the contents of the directory and confirm that the helloworld.exe file has been created. (Refer to Fig. 5-4)

```

C:\cygwin\LinCon8k\examples\common> arm-linux-gcc -o helloworld.exe helloworld.c
C:\cygwin\LinCon8k\examples\common> dir/w
Volume in drive C has no label.
Volume Serial Number is 649C-FEE2
C:\cygwin\LinCon8k\examples\common

[.]                [..]                back_plane_id.c      back_plane_id.exe
echosvr.c          echosvr.exe          eeprom.c             eeprom.exe
getdo_rb.c         getdo_rb.exe         getexai.c            getexai.exe
getexdi.c         getexdi.exe         getlist.c            getlist.exe
getport.c         getport.exe         getreceive.c         getreceive.exe
getsendreceive.c  getsendreceive.exe  getsendreceive_bin.c getsendreceive_bin.exe
helloworld.c      helloworld.exe      led.c                led.exe
port.c            port.exe            read_sn.c            read_sn.exe
rotary_id.c       rotary_id.exe       send_receive.c       send_receive.exe
setdo_hw.c        setdo_hw.exe        setexao.c            setexao.exe
setexdo.c         setexdo.exe        setport.c            setport.exe
setsend.c         setsend.exe         sram.c               sram.exe
timer.c           timer.exe           timer2.c             timer2.exe
uart.c            uart.exe            wdt.c                wdt.exe
wdt_safe_value.c  wdt_safe_value.exe

                    56 File(s)          2,257,242 bytes
                    2 Dir(s)  98,268,422,144 bytes free

```

Fig. 5-4

Step 2: Transfer helloworld.exe to the LP-51xx

Two methods can be used to transfer files to the LP-51xx:

<Method one> Using the “DOS Command Prompt”

- (1) Open a “DOS Command Prompt” and type the ftp IP Address of the LP-51xx, for example, **ftp 192.168.0.200** to establish a connection to the FTP Server on the LP-51xx. When prompted, type the **User_Name** and Password (“**root**” is the default value for **both**) to establish a connection to the LP-51xx.
- (2) Before transferring the files to the LP-51xx, type in the “**bin**” command to ensure that the file transferred to the LP-51xx **in binary mode**. (refer to Fig.5-5)

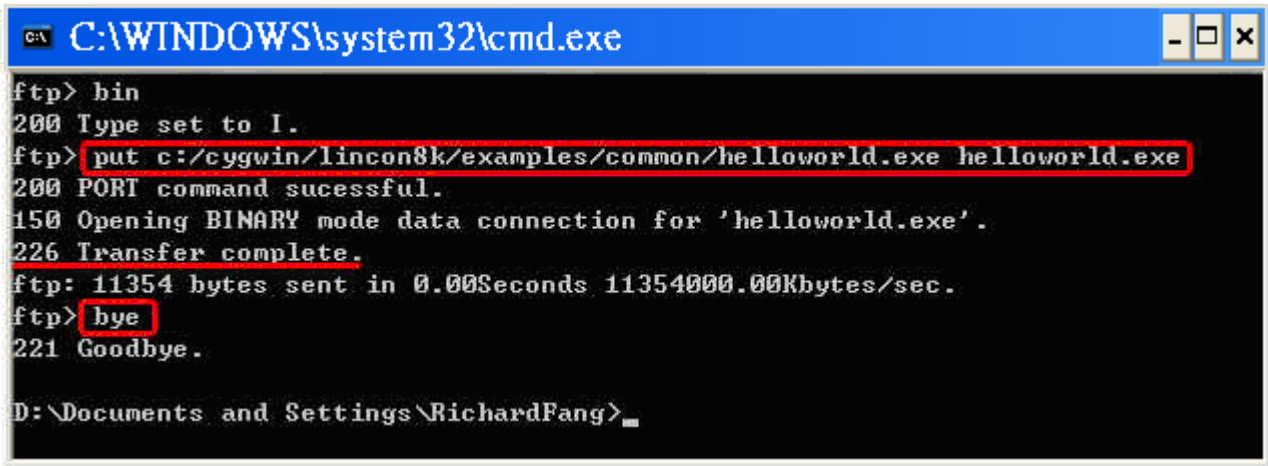
```

C:\WINDOWS\system32\cmd.exe - ftp 192.168.0.200
C:\>ftp 192.168.0.200
Connected to 192.168.0.200.
220 localhost FTP server (GNU inetutils 1.4.2) ready.
User (192.168.0.200:(none)): root
331 Password required for root.
Password:
230- MOKI 0.90
230 User root logged in.
ftp: bin
200 Type set to I.
ftp>

```

Fig.5-5

- (3) Type the command “**put C:/cygwin/LinCon8k/examples/common/helloworld.exe helloworld.exe**” to transfer the helloworld.exe file to the LP-51xx. Once the message “**Transfer complete**” is displayed, then transfer process has been completed. To disconnect from the LP-51xx, type the “**bye**” command to return to the PC console. (refer to Fig. 5-6).



```
C:\WINDOWS\system32\cmd.exe
ftp> bin
200 Type set to I.
ftp> put c:/cygwin/lincon8k/examples/common/helloworld.exe helloworld.exe
200 PORT command successful.
150 Opening BINARY mode data connection for 'helloworld.exe'.
226 Transfer complete.
ftp: 11354 bytes sent in 0.00Seconds 11354000.00Kbytes/sec.
ftp> bye
221 Goodbye.

D:\Documents and Settings\RichardFang>
```

Fig.5-6

<Method two> Using an FTP Client:

- (1) Open the FTP Software and add an FTP Host to the LP-51xx. The default value for both the **User_Name** and **Password** is “**root**”.
- (2) Then click the “**Quickconnect**” button to establish a connection to the FTP server on the LP-51xx. (refer to Fig. 5-7).



Fig.5-7

(3) Upload the “**Helloworld.exe**” file to the LP-51xx. (refer to Fig.5-8).

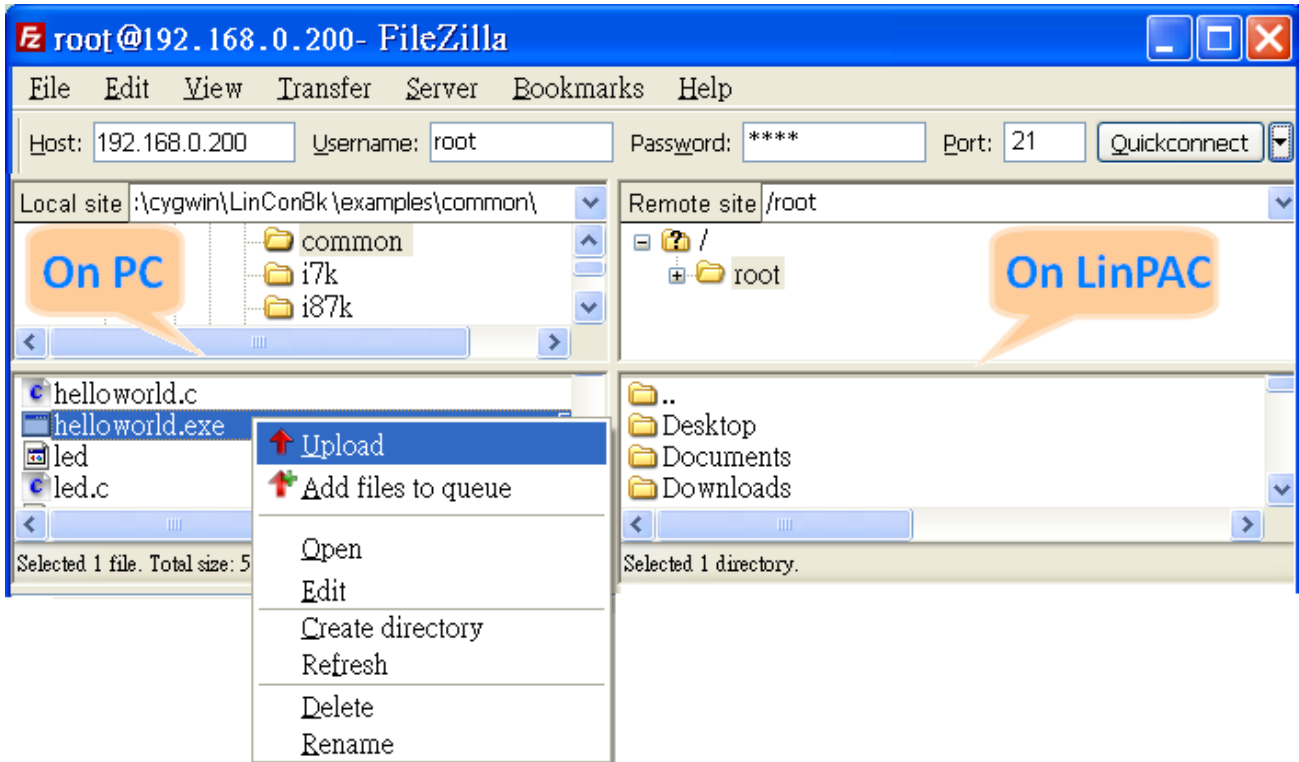


Fig.5-8

(4) Click the `helloworld.exe` file in the LP-51xx to select it and then right click the file icon and click the “**File Permissions**” option. In the Properties dialog box, type 777 into the Numeric textbox, and then click the OK button. Refer to Fig. 5-9 and Fig. 5-10 for more details.

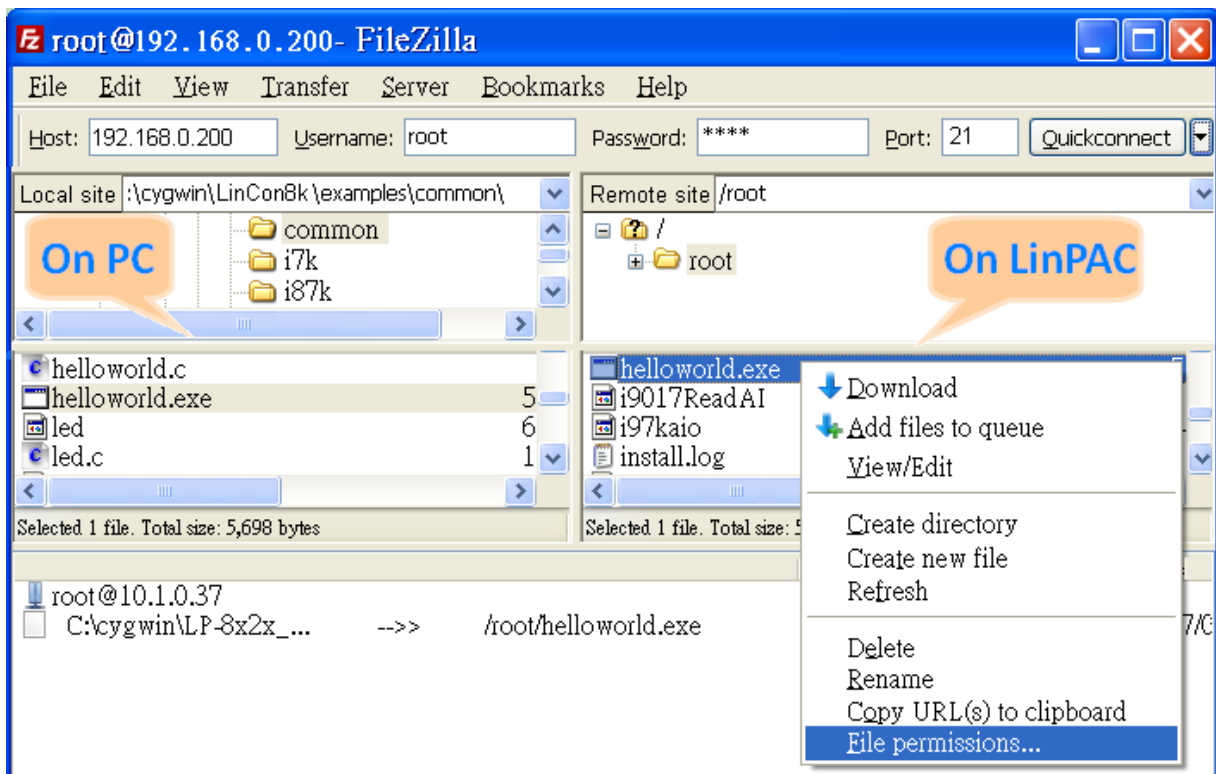


Fig.5-9

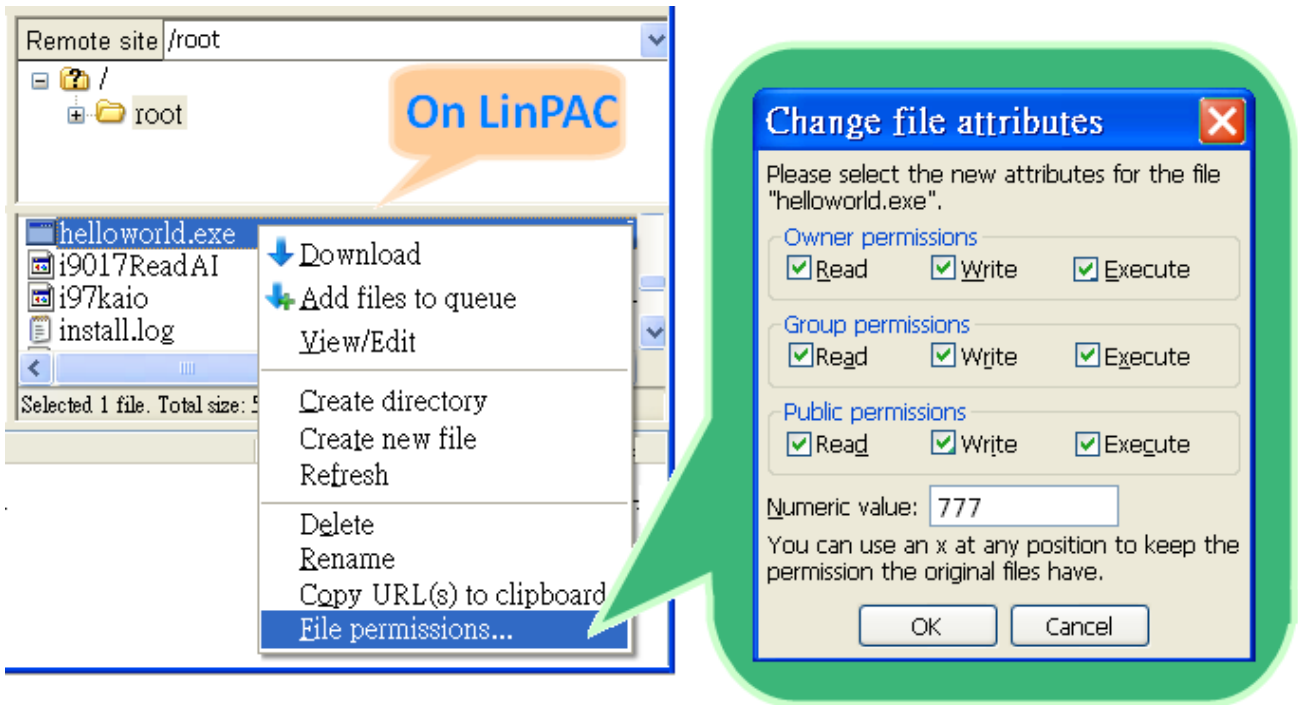


Fig.5-10

Step 3: Use telnet to access the LP-51xx and execute the program

(1) Open a “DOS Command Prompt” and type the IP Address of the LP-51xx, for example, **telnet 192.168.0.200**, to establish a connection to the Telnet Server on the LP-51xx. When prompted, type the **User_Name** and **Password (“root” is the default value for both)** to establish a connection to the LP-51xx. If the “#” prompt character is displayed, it signifies that a connection to the telnet server on the LP-51xx has been successfully established. (refer to Fig. 5-11)

```

C:\> Telnet 192.168.0.200
.LN      _NNNNN  .NNNNN_  .NNNNN_  (L      .JNNNNNN
(NN      .NNNF"4F (NN""NNL (NN"4NNN.  .NN      .NNN"4F4F
JN) (NN`      NN) `NN JN) 4NN      .NNN) (N)
NN) .NN)      NN) (NN NN)  NN      .NN4N) (NNL
NN` (NN      (NN_ANN) NN      (NN      NN` NN      `NNNL .
(NN (N)      JNNNNNF` (NN      JNF NN) NN.      4NN)N)
(NF (NN      . NN)      (N)      JNN` JNNNNNNN) (N)N)
NN)  NNNL_JN) (NN      NNL_NNNN` JNF      (NN .NL_NNN)N)
NN` "NNNF` (NF      NNNNN" (NN      NN `NNNF` F
LinPAC-5000 series
Linux embedded controller
linpac-5000 login: root
Password:
Distributor ID:      ICP DAS
Description:         LinPAC-5x4x
Release OS:          1.2
Release bootloader:  1.2
Codename:            PACLNX 0.90
installed XW-boards list
slot 1 ... XW-107
#
  
```

Fig.5-11

(2) Type the “**ls -l**” command to list all the files in the /root directory and verify that the helloworld.exe file is present. Type the “**chmod 777 helloworld.exe**” command to change the permissions for the helloworld.exe file if necessary. This means that the file is executable. Execute the “**./helloworld.exe**” file by typing and the message “Welcome to LinPAC-5000” will be displayed.

The compile, transfer and execution processes are now complete. (refer to Fig. 5-12)

```

Telnet 192.168.0.200
#
# chmod 777 helloworld.exe
# ls -l
-rwxrwxrwx 1 root root 5243 Jun 30 14:52
# ./helloworld.exe
Hi ~ Welcome to LinPAC-5000
#
  
```

Fig.5-12

5.4 i-Talk Utility

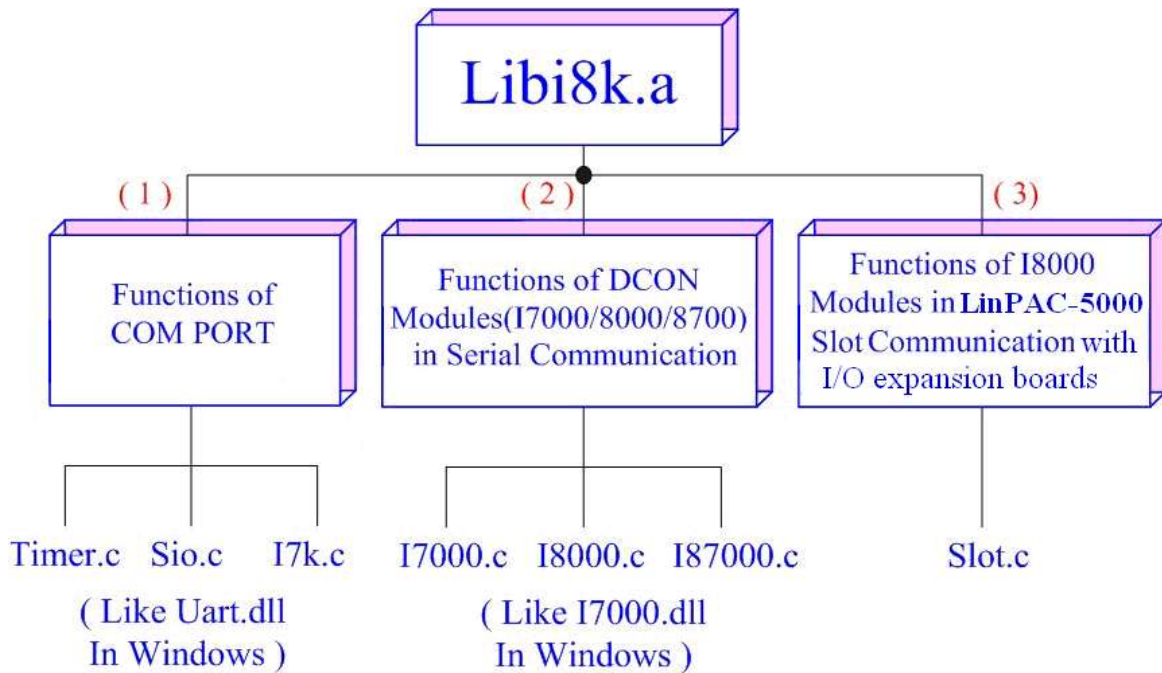
The **i-Talk utility** provides **eleven instructions** that make it convenient for users to access the modules and hardware in the LP-51xx and can be found in the path – **/usr/local/bin**. An overview of the i-Talk utility functions is given below (Refer to Fig. 5-13) Typing the name of the instruction will display usage details for the instruction.

Instruction	Description
getport	Get port value by offset from a module
setport	Set port value by offset to a module
setsend	Send string from LP-51xx COM port
getreceive	Receive string from LP-51xx COM port
getsendreceive	Send/Receive string from LP-51xx COM port
read_sn	Get Hardware Serial Number of LP-51xx
getlist_xw	Get what's the xwboard plug-in LP-51xx
setexdo	Set digital output value to xw-board and i-7k module
setexao	Set analog output value to xw-board and i-7k module
getexdi	Set digital input value to xw-board and i-7k module
getexai	Get analog input value to xw-board and i-7k module

Fig.5-13

6. LIBI8K.A

This section provides examples that focus on the description of and application of the functions found in the Libi8k.a library. The functions contained in the Libi8k.a library can be classified into three groups, as illustrated in Fig. 6-1.



Structure of Libi8k.a

Fig. 6-1

The functions node (1) and (2) in the diagram for the Libi8k.a library is the same as those of the DCON.DLL Driver (including Uart.dll and I7000.dll) as used in the DCON modules (I-7000/ I-8000/ I-87000 for serial communication). For more details, refer to the DCON.DLL Driver manual which includes a description of how to use the functions on DCON modules. The DCON.DLL Driver has now been included in the Libi8k.a library.

Functions (3) in the diagram contains the most important functions, as they are especially designed for I-8000 modules inserted in the LP-51xx slots. They are different from functions (1) and (2) because the communication method for I-8000 modules inserted in the LP-51xx is based on parallel mode rather than serial mode. Accordingly, the I8000.c library has been completely rewritten especially for I-8000 modules inserted in LP-51xx slots and has been rename as slot.c. The following is an introduction to the functions for slot.c, which can be arranged into six main categories:

1. System Information Functions;
2. Watch Dog Timer Functions;
3. EEPROM Read/Write Functions;
4. Digital Input/Output Functions;
5. Analog Input Functions;
6. Analog Output Functions;

Note that when using a development tool to create develop applications, the **msw.h** file must be included in the header of the source program, and the **Libi8k.a** library file must also be linked. To control ICP DAS remote I/O modules such as the I-7K, I-8K and I-87K series modules **via the COM1 or COM2 or COM3 ports on the LP-51xx**, the functions are the same as those included in the DCON DLL. To control **I-8K series modules** that are inserted in the slots of the LP-51xx, the functions are different and they are described in more detail below:

6.1 System Information Functions

■ Open_Slot

Description:

This function is used to open and initialize a specific slot on the LP-51xx, the I/O expansion board(http://www.icpdas.com/products/PAC/up-5000/XW-board_Selection_Guide.htm) in the LP-51xx will use this function. For example, to send or receive data from a specific slot, this function must be called first before any other functions can be used.

Syntax:

[C]
<code>int Open_Slot(int slot)</code>

Parameter:

slot: [Input] Specifies the slot where the I/O module is inserted.

Return Value:

0: The slot was successfully initialized.

Other: The initialization failed.

Refer to Chapter 6.7: "Error Code Definitions" for details of other returned values.

Example:

```
Int slot=1;
Open_Slot(slot);
// The first slot in the LP-51xx will be open and initiated, and only for XW-board.
```

■ Close_Slot

Description:

After using the of `Open_Slot()` function to open and initialize a specific slot on the LP-51xx, the `Close_Slot()` function must also be used to close the slot. This function will be used by The I/O expansion board in the LP-51xx. For example, the `Close_Slot()` function should be called after sending or receiving data from the specified slot.

Syntax:

```
void Close_Slot(int slot) [ C ]
```

Parameter:

slot: [Input] Specifies the slot where the I/O module is inserted.

Return Value:

None

Example:

```
int slot=1;
Close_Slot(slot);
// The first slot in the LP-51xx will be closed, and only for XW-board.
```

Remark:

■ Open_SlotAll

Description:

This function is used to open and initialize all slots on the LP-51xx. For example, to send or receive data from multiple slots, this function can be used to simplify the program, and other functions can be used.

Syntax:

[C]
<code>int Open_SlotAll(void)</code>

Parameter:

None

Return Value:

0: The slot was successfully initialized.

Other: The initialization failed.

Refer to Chapter 6.7: "Error Code Definitions" for details of other returned values.

Example:

```
Open_SlotAll();  
// All slots in the LP-51xx will be open and initiated.
```

Remark:

■ Close_SlotAll

Description:

If you the Open_SlotAll() function was used to open and initialize all the slots on the LP-51xx, the Close_SlotAll() function can be used to quickly close them simultaneously. For example, the Close_SlotAll() function can be called after sending or receiving data from multiple slots to close all the slots at the same time.

Syntax:

[C]
<code>void Close_SlotAll(void)</code>

Parameter:

None

Return Value:

None

Example:

```
Close_SlotAll();  
// All slots in the LP-51xx will be closed.
```

Remark:

■ Open_Com

Description:

This function is used to open and configure the COM port and must be **called at least once before** sending/receiving a command via the COM port. For example, to send or receive data from a specified COM port, this function should be called first, and then other series functions can be used.

Syntax:

[C]
<code>WORD Open_Com(char port, DWORD baudrate, char cData, char cParity, char cStop)</code>

Parameter:

port: [Input] COM1, COM2, COM3..., COM255.
baudrate: [Input] 1200/2400/4800/9600/19200/38400/57600/115200
cDate: [Input] Data5Bit, Data6Bit, Dat7Bit, Data8Bit
cParity: [Input] NonParity, OddParity, EvenParity
cStop: [Input] OneStopBit, TwoStopBit

Return Value:

0: The com port was successfully initialized.
Other: The initialization failed.
Refer to Chapter 6.7: "Error Code Definitions" for details of other returned values.

Example:

```
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);
```

Remark:

■ Close_Com

Description:

This function is used to closes and releases the resources of the COM port computer resources. And it must be **called before exiting the application program**. The Open_Com will return an error message if the program exit without calling Close_Com function.

Syntax:

[C]
BOOL Close_Com(char port)

Parameter:

port : [Input] COM1,COM2, COM3...COM255.

Return Value:

None

Example:

```
Close_Com (COM3);
```

Remark:

■ Send_Receive_Cmd

Description:

This function is used to send a command string to RS-485 network and receive the response from the RS-485 network. If the `wChkSum=1`, this function automatically adds the two checksum bytes into the command string and also check the checksum status when receiving the response from the modules. Note that the end of sending string is added `[0x0D]` to mean the termination of every command.

Syntax:

[C]

```
WORD Send_Receive_Cmd (char port, char szCmd[ ], char szResult[ ],  
WORD wTimeOut, WORD wChksum, WORD *wT)
```

Parameter:

port:	[Input] 1=COM1, 2=COM2, 3=COM3..., 255=COM255.
szCmd:	[Input] Sending command string
szResult:	[Input] Receiving the response string from the modules
wTimeOut:	[Input] Communicating timeout setting, the unit=1ms
wChkSum:	[Input] 0=Disable, 1=Enable
*wT:	[Output] Total time of send/receive interval, unit=1 ms

Return Value:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 6.7: "Error Code Definitions" for details of other returned values.

Example:

```
char m_port =1;  
DWORD m_baudrate=115200;  
WORD m_timeout=100;  
WORD m_chksum=0;  
WORD m_wT;  
char m_szSend[40], m_szReceive[40];  
int RetVal;  
m_szSend[0] = '$';  
m_szSend[1] = '0';  
m_szSend[2] = '0';
```

```

m_szSend[3] = 'M';
m_szSend[4] = 0;
/* open device file */
Open_Slot(1);
RetVal = Open_Com(m_port, m_baudrate, Data8Bit, NonParity, OneStopBit);
if (RetVal >0)
{
    printf("Open COM%d failed!\n", m_port);
    return FAILURE;
}
RetVal = Send_Receive_Cmd(m_port, m_szSend, m_szReceive, m_timeout,
                        m_chksum, &m_wT);
if (RetVal)
{
    printf("Module at COM%d Address %d error !!!\n", m_port, m_szSend[2] );
    return FAILURE;
}
Close_Com (m_port);

```

Remark:

■ Send_Cmd

Description:

This function only sends a command string to DCON series modules. If the wChkSum=1, it automatically **adds the two checksum bytes to the command string**. And then the end of sending string is further added [0x0D] to mean the termination of the command (szCmd). And this command string cannot include space char within the command string. For example: "\$01M 02 03" is user's command string. However, the actual command send out is "\$01M".

Syntax:

[C]
WORD Send_Cmd (char port, char szCmd[], WORD wTimeOut, WORD wChksum)

Parameter:

port: [Input] 1=COM1, 2=COM2, 3=COM3..., 255=COM255.
szCmd: [Input] Sending command string
wTimeOut: [Input] Communicating timeout setting, the unit=1ms
wChkSum: [Input] 0=Disable, 1=Enable

Return Value:

None

Example:

```
char m_port=1, m_szSend[40];  
DWORD m_baudrate=115200;  
WORD m_timeout=100, m_chksum=0;  
m_szSend[0] = '$';  
m_szSend[1] = '0';  
m_szSend[2] = '0';  
m_szSend[3] = 'M';  
Open_Slot(2); // The module is inserted in slot 2 and address is 0.  
Open_Com(m_port, m_baudrate, Data8Bit, NonParity, OneStopBit);  
Send_Cmd(m_port, m_szSend, m_timeout, m_chksum);  
Close_Com (m_port);
```

Remark:

■ Receive_Cmd

Description:

This function is used to obtain the responses string from the modules in the RS-485 network. And this function provides a response string without the last byte [0x0D].

Syntax:

```
[ C ]  
WORD Receive_Cmd (char port, char szResult[ ], WORD wTimeOut,  
WORD wChecksum)
```

Parameter:

port: [Input] 1=COM1, 2=COM2, 3=COM3..., 255=COM255.
szResult: [Output] Sending command string
wTimeOut: [Input] Communicating timeout setting, the unit=1ms
wChkSum: [Input] 0=Disable, 1=Enable

Return Value:

None

Example:

```
char m_port=3;  
char m_Send[40], m_szResult[40] ;  
DWORD m_baudrate=115200;  
WORD m_timeout=100, m_checksum=0;  
m_szSend[0] = '$';  
m_szSend[1] = '0';  
m_szSend[2] = '1';  
m_szSend[3] = 'M';  
m_szSend[4] = 0;  
Open_Com (m_port, m_baudrate, Data8Bit, NonParity, OneStopBit);  
Send_Cmd (m_port, m_szSend, m_timeout, m_checksum);  
Receive_Cmd (m_port, m_szResult, m_timeout, m_checksum);  
Close_Com (m_port);  
// Read the remote module:I-7016D , m_szResult : "!017016D"
```

Remark:

■ Send_Binary

Description:

Send out the command string by fix length, which is controlled by the parameter "iLen". The difference between this function and Send_cmd is that Send_Binary terminates the sending process by the string length "iLen" instead of the character "CR"(Carry return). Therefore, this function can send out command string with or without null character under the consideration of the command length. Besides, because of this function without any error checking mechanism (Checksum, CRC, LRC... etc.), users have to add the error checking information to the raw data by themselves if communication checking system is required. Note that this function is usually applied to communicate with the other device, but not for ICP DAS DCON (I-7000/8000/87K) series modules.

Syntax:

[C]
WORD Send_Binary (char port, char szCmd[], int iLen)

Parameter:

port: [Input] 1=COM1, 2=COM2, 3=COM3..., 255=COM255.
szCmd: [Input] Sending command string
iLen: [Input] The length of command string.

Return Value:

None

Example:

```
int m_length=4;
char m_port=3, char m_szSend[40];
DWORD m_baudrate=115200;
m_szSend[0] = '0';
m_szSend[1] = '1';
m_szSend[2] = '2';
m_szSend[3] = '3';
Open_Com(m_port, m_baudrate, Data8Bit, NonParity, OneStopBit);
Send_Binary(m_port, m_szSend, m_length);
Close_Com (m_port);
```

Remark:

■ Receive_Binary

Description:

This function is applied to receive the fix length response. The length of the receiving response is controlled by the parameter "iLen". The difference between this function and Receive_cmd is that Receive_Binary terminates the receiving process by the string length "iLen" instead of the character "CR"(Carry return). Therefore, this function can be used to receive the response string data with or without null character under the consideration of receiving length. Besides, because of this function without any error checking mechanism (checksum, CRC, LRC... etc.), users have to remove the error checking information from the raw data by themselves if communication checking system is used. Note that this function is usually applied to communicate with the other device, but not for ICP DAS DCON (I-7000/8000/87K) series modules.

Syntax:

```
[ C ]  
WORD Receive_Binary (char cPort, char szResult[], WORD wTimeOut,  
                    WORD wLen, WORD *wT)
```

Parameter:

port: [Input] 1=COM1, 2=COM2, 3=COM3..., 255=COM255.
szResult: [Input] Receiving the response string from the modules
wTimeOut: [Input] Communicating timeout setting, the unit=1ms
wLen: [Input] The length of command string.
*wT: [Output] Total time of send/receive interval, unit=1 ms

Return Value:

None

Example:

```
int m_length=10;  
char m_port=3;  
char m_szSend[40];  
char m_szReceive[40];  
DWORD m_baudrate=115200;  
WORD m_wt;  
WORD m_timeout=10;  
WORD m_wlength=10;
```

```
m_szSend[0] = '0';
m_szSend[1] = '1';
m_szSend[2] = '2';
m_szSend[3] = '3';
m_szSend[4] = '4';
m_szSend[5] = '5';
m_szSend[6] = '6';
m_szSend[7] = '7';
m_szSend[8] = '8';
m_szSend[9] = '9';
Open_Com(m_port, m_baudrate, Data8Bit, NonParity, OneStopBit);
// send 10 character
Send_Binary(m_port, m_szSend, m_length);
// receive 10 character
Receive_Binary(m_port, m_szResult, m_timeout, m_wlength, &m_wt);
Close_Com (m_port);
```

Remark:

■ sio_open

Description:

This function is used to open and initiate a specified serial port in the LP-51xx. The n-port modules in the LP-51xx will use this function. For example, if you want to send or receive data from a specified serial port, this function must be called first. Then the other functions can be used later.

Syntax:

```
                                [ C ]  
  
int sio_open(const char *port, speed_t baudrate, tcflag_t data, tcflag_t parity,  
            tcflag_t stop)
```

Parameter:

port: [Input] device name: /dev/ttyS0, /dev/ttyS1.../dev/ttyS34
baudrate: [Input] B1200/ B2400/ B4800/ B9600/ B19200/ B38400/ B57600/
B115200
date: [Input] DATA_BITS_5/ DATA_BITS_6/ DATA_BITS_7/ DATA_BITS_8
parity: [Input] NO_PARITY / ODD_PARITY / EVEN_PARITY
stop: [Input] ONE_STOP_BIT / TWO_STOP_BITS

Return Value:

This function returns int port descriptor for the port opened successfully.

ERR_PORT_OPEN is for Failure

Example:

```
#define COM_M1 "/dev/ttyS0" // Defined the first port for COM2(RS-485)  
char fd[5];  
fd[0]=sio_open(COM_M1, B9600, DATA_BITS_8, NO_PARITY,ONE_STOP_BIT);  
if (fd[0] == ERR_PORT_OPEN) {  
    printf("open port_m failed!\n");  
    return (-1);  
}  
// The first port will be opened.
```

Remark:

■ **sio_close**

Description:

If you have used the function of `sio_open()` to open the specified serial port in the LP-51xx, you need to use the `sio_close()` function to close the specified serial port in the LP-51xx. For example, once you have finished sending or receiving data from a specified serial port, this function would then need to be called.

Syntax:

[C]
<code>int sio_close(int port)</code>

Parameter:

port : [Input] device name: /dev/ttyS0, /dev/ttyS1.../dev/ttyS34

Return Value:

None

Example:

```
#define COM_M2 "/dev/ttyS1" // Defined the second port for COM3(RS-232)
char fd[5];
fd[0]=sio_open(COM_M2, B9600, DATA_BITS_8, NO_PARITY, ONE_STOP_BIT);
sio_close (fd[0]);
// The second port will be closed.
```

Remark:

■ `sio_set_noncan`

Description:

If you have used the function of `sio_open()` to open the specified serial port in the LP-51xx, you need to use the `sio_close()` function to close the specified serial port in the LP-51xx. For example, once you have finished sending or receiving data from a specified serial port, this function would then need to be called set a opened serial port to non-canonical mode.

Syntax:

```
[ C ]  
  
int sio_set_noncan (int port)
```

Parameter:

port : [Input] device name: /dev/ttyS0, /dev/ttyS1.../dev/ttyS34

Return Value:

None

Example:

```
#define COM_M2 "/dev/ttyS1" // Defined the second port for COM3(RS-232)  
char fd[5];  
fd[0]=sio_open(COM_M2, B9600, DATA_BITS_8, NO_PARITY, ONE_STOP_BIT);  
sio_close (fd[0]);  
// The second port will be closed.
```

Remark:

■ Read_SN

Description:

This function is used to retrieve the hardware serial identification number on the LP-51xx main controller. This function supports the control of hardware versions by reading the serial ID chip.

Syntax:

```
[ C ]  
void Read_SN(unsigned char serial_num[])
```

Parameter:

serial_num: [Output] Receive the serial ID number.

Return Value:

None

Example:

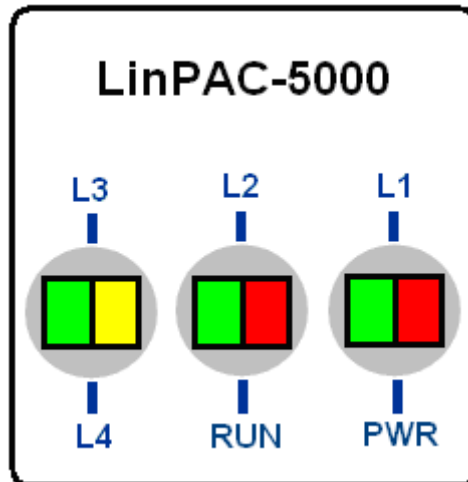
```
int slot;  
unsigned char serial_num[8];  
Open_Slot(0);  
Read_SN(serial_num);  
printf("SN=%x%x%x%x%x%x%x%x\n",serial_num[0],serial_num[1], serial_num[2]  
      ,serial_num[3],serial_num[4],serial_num[5],serial_num[6],serial_num[7]);
```

Remark:

■ SetLED

Description:

This function is used to turn the LP-51xx LED's on/off.



Address	L4	L3	L2	RUN/L5	PWR	L1
Color	Green	Yellow	Green	Red	Green	Red
Programmable	Yes	Yes	Yes	Yes	No	Yes
Function	None	None	None	Start	Power	None

Syntax:

```
[ C ]  
void SetLED(unsigned int addr, unsigned int value)
```

Parameter:

addr: [Input] Range of programmable LED display is 1 to 5
value: [Input] **1** : Turn on the LED
0 : Turn off the LED

Return Value:

None

Example:

```
unsigned int addr,value;  
addr=4;  
value=1;  
SetLED(addr, value);  
// Turn on the LED4.
```

Remark:

■ GetBackPlaneID

Description:

This function is used to retrieve the back plane ID number in the LP-51xx.

Syntax:

[C]
<code>int GetBackPlaneID()</code>

Parameter:

None

Return Value:

Backplane ID number.

Example:

```
int id;
id=GetBackPlaneID();
printf("GetBackPlaneID =%d \n", id);
// Get the LP-51xx backplane id . Returned Value: GetBackPlaneID = 2
```

Remark:

■ GetRotaryID

Description:

This function is used to retrieve the rotary ID number in the LP-51xx.

Syntax:

[C]
<code>int GetRotaryID(int type, &id)</code>

Parameter:

slot: [input] number of slot.
id: [Output] Rotary ID number

Return Value:

0: The slot was successfully initialized.

Other: The initialization failed.

Refer to Chapter 6.7: "Error Code Definitions" for details of other returned values.

Example:

```
int id, slot, type, wRetVal;  
type = 2; // for LP-5000  
wRetVal = Open_Slot(slot);  
if (wRetVal > 0) {  
    printf("open Slot%d failed!\n",slot);  
    return (-1);  
}  
id= GetRotaryID(type, &id);  
printf("GetRotaryID =%d \n",id); // Get the LP-51xx rotary id. If user turn the rotary  
switch to the 1 position, would get the returned value: GetRotaryID = 78
```

Remark:

SW	0	1	2	3	4	5	6	7	8	9
ID	47	46	45	44	43	42	41	40	39	38

■ GetSDKversion

Description:

This function is used to retrieve the version of LP-51xx SDK.

Syntax:

```
float GetSDKversion(void) [ C ]
```

Parameter:

None

Return Value:

Version number.

Example:

```
printf(" GetSDKversion = %4.2f \n ", GetSDKversion());  
// Get the LP-51xx SDK version number.  
// Returned Value: GetSDKversion = 1.0
```

Remark:

■ GetNameOfModule

Description:

This function is used to retrieve the name of an XWboard series I/O module, which is plugged into a slot in the LP-51xx. This function supports the collection of system hardware configurations.

Syntax:

```
int GetNameOfModule_xw() [ C ]
```

Parameter:

slot: [Input] Specifies the slot where the I/O module is inserted.

Return Value:

I/O module ID. For Example, the XW107 module will return XW107.

Example:

```
int slot;  
int moduleID;  
Open_Slot(1);  
moduleID=GetNameOfModule_xw();  
Close_Slot(1);  
// The XW107 card plugged in slot 1 of LP-51xx  
// Returned Value: moduleName=" XW107 "
```

Remark:

One LP-51xx can only plug only one XW-board.



6.2 Watch Dog Timer Functions

- EnableWDT
- DisableWDT

Description:

This function can be used to enable the watch dog timer (WDT) and users need to reset WDT in the assigned time set by users. Or LinPAC will reset automatically.

Syntax:

```
[C]
void EnableWDT(unsigned int msecond)
void DisableWDT(void)
```

Parameter:

milliseconds: LinPAC will reset in the assigned time if users don't reset WDT.
The unit is mini-second.

Return Value:

None

Example:

```
EnableWDT(10000); //Enable WDT interval 10000ms=10s
while (getchar()==10)
{
    printf("Refresh WDT\n");
    EnableWDT(10000); //Refresh WDT 10s
}
printf("Disable WDT\n");
DisableWDT();
```

Remark:

■ WatchDogSWEven

Description:

This function is used to read the LinPAC Reset Condition and users can reinstall the initial value according to the Reset Condition.

Syntax:

```
[C]  
unsigned int WatchDogSWEven (void)
```

Parameter:

None

Return Value:

Just see the last number of the return value – **RCSR** (Reset Controller Status Register). For example, RCSR is “20009a4”, so just see the last number “4”. 4 is **0100** in bits and it means:

Bit 0: [Hardware Reset](#) (Like Power Off, Reset Button)

Bit 1: [Software Reset](#) (Like Type “shutdown -r now” or “reboot” in command prompt)

Bit 2: [WDT Reset](#) (Like Use “EnableWDT(1000)”)

Bit 3: [Sleep Mode Reset](#) (Not supported in the LinPAC)

Example:

```
printf("RCRS = %x\n", WatchDogSWEven() );
```

Remark:

■ ClearWDTSEven

Description:

This function is used to clear RCSR value.

Syntax:

```
[C]  
void ClearWDTSEven (unsigned int rcsr)
```

Parameter:

rcsr : Clear bits of RCSR. Refer to the following parameter setting :

- 1 : clear bit 0
- 2 : clear bit 1
- 4 : clear bit 2
- 8 : clear bit 3
- F : clear bit 0 to bit 3

Return Value:

None

Example:

```
ClearWDTSEven(0xF) ; // Used to clear bit 0 to bit 3 of RCRS to be zero.
```

Remark:

6.3 EEPROM Read/Write Functions

■ Enable_EEP

Description:

This function is used to make EEPROM able to read or write. It must be used before using Read_EEP or Write_EEP. This total size for the EEPROM is 0x0 ~ 0x3FFF (16,384 bytes; 16K) capacity.

Syntax:

[C]
<code>void Enable_EEP(void)</code>

Parameter:

None

Return Value:

None

Example:

```
Enable_EEP();  
// After using this function, you can use Write_EEP or Read_EEP to write or read  
// data of EEPROM.
```

Remark:

■ Disable_EEP

Description:

This function is used to make EEPROM unable to read or write. You need to use this function after using Read_EEP or Write_EEP. Then it will protect you from modifying your EEPROM data carelessly.

Syntax:

[C]
<code>void Disable_EEP(void)</code>

Parameter:

None

Return Value:

None

Example:

```
Disable_EEP();
```

// After using this function, you will not use Write_EEP or Read_EEP to write or read data of EEPROM.

Remark:

■ Read_EEP

Description:

This function will read one byte data from the EEPROM. There is a 16K-byte (0~0x3fff) EEPROM in the main control unit in the LP-51xx system. This EEPROM with its accessing APIs provides another mechanism for storing critical data inside non-volatile memory.

Syntax:

```
[ C ]  
unsigned char Read_EEP(int block, int offset)
```

Parameter:

- block: [Input] this is reserved for the system, and the default value is 0.
- offset: [Input] specifies the memory address where read from.

Return Value:

Data read from the EEPROM.

Example:

```
int block=0, offset;  
unsigned char data;  
Enable_EEP();  
data= ReadEEP(block, offset);  
Disable_EEP();  
// Returned value: data= read an 8-bit value from the EEPROM (block & offset)
```

Remark:

■ Write_EEP

Description:

To write one byte of data to the EEPROM. There is a 16K-byte (0~0x3fff) EEPROM in the main control unit of the LP-51xx system. This EEPROM is divided into 64 blocks (block 0 to 63), and each block has 256 bytes in length from the offset of 0 to 255. This EEPROM with its accessing APIs provides another mechanism for storing critical data inside non-volatile memory.

Syntax:

```
[ C ]  
void Write_EEP(int block, int offset, unsigned char data)
```

Parameter:

block: [Input] this is reserved for the system, and the default value is 0.
offset: [Input] specifies the memory address where write from.
Data: [Input] data to write to EEPROM.

Return Value:

None

Example:

```
int block, offset;  
unsigned char data=10;  
Enable_EEP()  
WriteEEP(block, offset, data);  
Disable_EEP();  
// Writes a 10 value output to the EEPROM (block & offset) location
```

Remark:

6.4 Digital Input/Output Functions

6.4.1 I-7000 series modules

■ DigitalOut

Description:

This function is used to output the value of the digital output module for I-7000 series modules.

Syntax:

```
[ C ]  
WORD DigitalOut(WORD wBuf[], float fBuf[], char szSend[], char szReceive[])
```

Parameter:

wBuf: WORD Input/Output argument table
wBuf[0]: [Input] COM port number, from 1 to 255
wBuf[1]: [Input] Module address, form 0x00 to 0xFF
wBuf[2]: [Input] Module ID, 0x7011/12/14/42/43/44/50/60/63/65/66/67/80
wBuf[3]: [Input] 0= Checksum disable; 1= Checksum enable
wBuf[4]: [Input] Timeout setting , normal=100 msecond
wBuf[5]: [Input] 16-bit digital output data
wBuf[6]: [Input] 0 → no save to szSend &szReceive
 1 → Save to szSend &szReceive
fBuf: Not used.
szSend: [Input] Command string to be sent to I-7000 series modules.
szReceive: [Output] Result string receiving from I-7000 series modules.

Return Value:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 6.7: "Error Code Definitions" for details of other returned values.

Example:

```
char szSend[80];  
char szReceive[80];  
float fBuf[12];  
WORD wBuf[12];  
WORD m_port=3;  
WORD m_address=1;  
WORD m_timeout=100;
```

```
WORD m_checksum=0;
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);
wBuf[0] = m_port;
wBuf[1] = m_address;
wBuf[2] = 0x7050;
wBuf[3] = m_checksum;
wBuf[4] = m_timeout;
wBuf[5] = 0x0f;           // 8 DO Channels On
wBuf[6] = 0;
DigitalOut(wBuf, fBuf, szSend, szReceive);
Close_Com(COM3);
```

Remark:

■ DigitalBitOut

Description:

This function is used to set the digital output value of the channel No. of I-7000 series modules. The output value is “0” or “1”.

Syntax:

```
[ C ]  
WORD DigitalBitOut(WORD wBuf[ ], float fBuf[ ], char szSend[ ], char szReceive[ ])
```

Parameter:

wBuf: WORD Input/Output argument talbe
wBuf[0]: [Input] COM port number, from 1 to 255
wBuf[1]: [Input] Module address, form 0x00 to 0xFF
wBuf[2]: [Input] Module ID, 0x7042/43/44/50/60/63/65/66/67
wBuf[3]: [Input] 0= Checksum disable; 1= Checksum enable
wBuf[4]: [Input] Timeout setting , normal=100 msecond
wBuf[5]: Not used
wBuf[6]: [Input] 0 → no save to szSend &szReceive
 1 → Save to szSend &szReceive
wBuf[7]: [Input] The digital output channel No.
wBuf[8]: [Input] Logic value(0 or 1)
fBuf: Not used.
szSend: [Input] Command string to be sent to I-7000 series modules.
szReceive: [Output] Result string receiving from I-7000 series modules.

Return Value:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 6.7: “Error Code Definitions” for details of other returned values.

Example:

```
char szSend[80];  
char szReceive[80];  
float fBuf[12];  
WORD wBuf[12];  
WORD m_port=3;  
WORD m_address=1;  
WORD m_timeout=100;  
WORD m_checksum=0;
```



```
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);
wBuf[0] = m_port;
wBuf[1] = m_address;
wBuf[2] = 0x7065;
wBuf[3] = m_checksum;
wBuf[4] = m_timeout;
wBuf[6] = 0;
wBuf[7] = 0x08;           //RL4 relay On
wBuf[8] = 1;
DigitalBitOut (wBuf, fBuf, szSend, szReceive);
Close_Com(COM3);
```

Remark:

■ DigitalOutReadBack

Description:

This function is used to read back the digital output value of I-7000 series modules.

Syntax:

```
[ C ]  
WORD DigitalOutReadBack(WORD wBuf[ ], float fBuf[ ],char szSend[ ],  
                        char szReceive[ ])
```

Parameter:

wBuf: WORD Input/Output argument talbe
wBuf[0]: [Input] COM port number, from 1 to 255
wBuf[1]: [Input] Module address, form 0x00 to 0xFF
wBuf[2]: [Input] Module ID, 0x7042/43/44/50/60/63/65/66/67/80
wBuf[3]: [Input] 0=Checksum disable; 1=Checksum enable
wBuf[4]: [Input] Timeout setting , normal=100 msecond
wBuf[5]: [Output] 16-bit digital output data read back
wBuf[6]: [Input] 0 → no save to szSend &szReceive
 1 → Save to szSend & szReceive
fBuf: Not used.
szSend: [Input] Command string to be sent to I-7000 series modules.
szReceive: [Output] Result string receiving from I-7000 series modules.

Return Value:

0 is for Success
Not 0 is for Failure

Example:

```
char szSend[80];  
char szReceive[80];  
float fBuf[12];  
WORD DO;  
WORD wBuf[12];  
WORD m_port=3;  
WORD m_address=1;  
WORD m_timeout=100;  
WORD m_checksum=0;  
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);  
wBuf[0] = m_port;
```

```
wBuf[1] = m_address;  
wBuf[2] = 0x7050;  
wBuf[3] = m_checksum;  
wBuf[4] = m_timeout;  
wBuf[6] = 0;  
DigitalOutReadBack (wBuf, fBuf, szSend, szReceive);  
DO=wBuf[5];  
Close_Com(COM3);
```

Remark:

■ DigitalOut_7016

Description:

This function is used to set the digital output value of the specified channel No. of I-7016 module. If the parameter of wBuf[7] is "0", it means to output the digital value through Bit0 and Bit1 digital output channels. If wBuf[7] is "1", it means to output the digital value through Bit2 and Bit3 digital output channels.

Syntax:

```
[ C ]  
WORD DigitalOut_7016(WORD wBuf[], float fBuf[], char szSend[], char szReceive[])
```

Parameter:

wBuf: WORD Input/Output argument talbe
wBuf[0]: [Input] COM port number, from 1 to 255
wBuf[1]: [Input] Module address, form 0x00 to 0xFF
wBuf[2]: [Input] Module ID, 0x7016
wBuf[3]: [Input] 0= Checksum disable; 1= Checksum enable
wBuf[4]: [Input] Timeout setting , normal=100 msecond
wBuf[5]: [Input] 2-bit digital output data in decimal format
wBuf[6]: [Input] 0 → no save to szSend &szReceive
 1 → Save to szSend &szReceive
wBuf[7]: [Input] 0 : Bit0, Bit1 output
 1 : Bit2, Bit3 output
fBuf: Not used.
szSend: [Input] Command string to be sent to I-7000 series modules.
szReceive: [Output] Result string receiving from I-7000 series modules.

Return Value:

0 is for Success
Not 0 is for Failure

Example:

```
char szSend[80];  
char szReceive[80];  
float fBuf[12];  
WORD wBuf[12];  
WORD m_port=3;  
WORD m_address=1;  
WORD m_timeout=100;
```

```
WORD m_checksum=0;
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);
wBuf[0] = m_port;
wBuf[1] = m_address;
wBuf[2] = 0x7016;
wBuf[3] = m_checksum;
wBuf[4] = m_timeout;
wBuf[5] = 1;
wBuf[6] = 0;
wBuf[7] = 1;    // Set the Bit2, Bit3 digital output
DigitalOut_7016(wBuf, fBuf, szSend, szReceive);
Close_Com(COM3);
```

Remark:

■ DigitalIn

Description:

This function is used to obtain the digital input value from I-7000 series modules.

Syntax:

```
[ C ]  
WORD DigitalIn(WORD wBuf[], float fBuf[], char szSend[], char szReceive[])
```

Parameter:

wBuf: WORD Input/Output argument talbe
wBuf[0]: [Input] COM port number, from 1 to 255
wBuf[1]: [Input] Module address, form 0x00 to 0xFF
wBuf[2]: [Input] Module ID, 0x7041/44/50/52/53/55/58/60/63/65
wBuf[3]: [Input] 0= Checksum disable; 1= Checksum enable
wBuf[4]: [Input] Timeout setting , normal=100 msecond
wBuf[5]: [Output] 16-bit digital output data
wBuf[6]: [Input] 0 → no save to szSend &szReceive
 1 → Save to szSend &szReceive
fBuf: Not used.
szSend: [Input] Command string to be sent to I-7000 series modules.
szReceive: [Output] Result string receiving from I-7000 series modules.

Return Value:

0 is for Success
Not 0 is for Failure

Example:

```
char szSend[80];  
char szReceive[80];  
float fBuf[12];  
WORD DI;  
WORD wBuf[12];  
WORD m_port=3;  
WORD m_address=1;  
WORD m_timeout=100;  
WORD m_checksum=0;  
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);  
wBuf[0] = m_port;  
wBuf[1] = m_address;
```

```
wBuf[2] = 0x7050;  
wBuf[3] = m_checksum;  
wBuf[4] = m_timeout;  
wBuf[6] = 0;  
DigitalIn(wBuf, fBuf, szSend, szReceive);  
DI=wBuf[5];  
Close_Com(COM3);
```

Remark:

■ DigitalInLatch

Description:

This function is used to obtain the latch value of the high or low latch mode of the digital input module.

Syntax:

```
[ C ]  
WORD DigitalInLatch(WORD wBuf[], float fBuf[], char szSend[], char szReceive[])
```

Parameter:

wBuf: WORD Input/Output argument talbe
wBuf[0]: [Input] COM port number, from 1 to 255
wBuf[1]: [Input] Module address, form 0x00 to 0xFF
wBuf[2]: [Input] Module ID, 0x7041/44/50/52/53/55/58/60/63/65/66
wBuf[3]: [Input] 0= Checksum disable; 1= Checksum enable
wBuf[4]: [Input] Timeout setting , normal=100 msecond
wBuf[5]: [Input] 0: low Latch mode ; 1:high Latch mode
wBuf[6]: [Input] 0 → no save to szSend &szReceive
 1 → Save to szSend &szReceive
wBuf[7]: [Output] Latch value
fBuf: Not used.
szSend: [Input] Command string to be sent to I-7000 series modules.
szReceive: [Output] Result string receiving from I-7000 series modules.

Return Value:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 6.7: “Error Code Definitions” for details of other returned values.

Example:

```
char szSend[80];  
char szReceive[80];  
float fBuf[12];  
WORD wBuf[12];  
WORD m_port=3;  
WORD m_address=1;  
WORD m_timeout=100;  
WORD m_checksum=0;  
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);
```



```
wBuf[0] = m_port ;  
wBuf[1] = m_address ;  
wBuf[2] = 0x7050;  
wBuf[3] = m_checksum ;  
wBuf[4] = m_timeout ;  
wBuf[5] = 1;      // Set the high Latch mode  
wBuf[6] = 0;  
wBuf[7] = 0x03;  // Set the Latch value  
DigitalInLatch(wBuf, fBuf, szSend, szReceive);  
Close_Com(COM3);
```

Remark:

■ ClearDigitalInLatch

Description:

This function is used to clear the latch status of digital input module when latch function has been enabling.

Syntax:

```
[ C ]  
WORD ClearDigitalInLatch(WORD wBuf[], float fBuf[],char szSend[],  
char szReceive[])
```

Parameter:

wBuf: WORD Input/Output argument talbe
wBuf[0]: [Input] COM port number, from 1 to 255
wBuf[1]: [Input] Module address, form 0x00 to 0xFF
wBuf[2]: [Input] Module ID, 0x7011/12/14/42/43/44/50/55/58/60/63/65/66/67
wBuf[3]: [Input] 0= Checksum disable; 1= Checksum enable
wBuf[4]: [Input] Timeout setting , normal=100 msecond
wBuf[5]: Not used.
wBuf[6]: [Input] 0 → no save to szSend &szReceive
1 → Save to szSend &szReceive
fBuf: Not used.
szSend: [Input] Command string to be sent to I-7000 series modules.
szReceive: [Output] Result string receiving from I-7000 series modules .

Return Value:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 6.7: "Error Code Definitions" for details of other returned values.

Example:

```
char szSend[80];  
char szReceive[80];  
float fBuf[12];  
WORD wBuf[12];  
WORD m_port=3;  
WORD m_address=1;  
WORD m_timeout=100;  
WORD m_checksum=0;  
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);
```

```
wBuf[0] = m_port;  
wBuf[1] = m_address;  
wBuf[2] = 0x7050;  
wBuf[3] = m_checksum;  
wBuf[4] = m_timeout;  
wBuf[6] = 0;  
ClearDigitalInLatch(wBuf, fBuf, szSend, szReceive);  
Close_Com(COM3);
```

Remark:

■ DigitalInCounterRead

Description:

This function is used to obtain the counter event value of the channel number of the digital input module.

Syntax:

```
[ C ]  
  
WORD DigitalInCounterRead(WORD wBuf[], float fBuf[], char szSend[],  
                           char szReceive[])
```

Parameter:

wBuf: WORD Input/Output argument talbe
wBuf[0]: [Input] COM port number, from 1 to 255
wBuf[1]: [Input] Module address, form 0x00 to 0xFF
wBuf[2]: [Input] Module ID, 0x7041/44/50/51/52/53/55/58/60/63/65
wBuf[3]: [Input] 0= Checksum disable; 1= Checksum enable
wBuf[4]: [Input] Timeout setting , normal=100 msecond
wBuf[5]: [Input] The digital input Channel No.
wBuf[6]: [Input] 0 → no save to szSend &szReceive
 1 → Save to szSend &szReceive
wBuf[7]: [Output] Counter value of the digital input channel No.
fBuf: Not used.
szSend: [Input] Command string to be sent to I-7000 series modules.
szReceive: [Output] Result string receiving from I-7000 series modules.

Return Value:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 6.7: "Error Code Definitions" for details of other returned values.

Example:

```
char szSend[80];  
char szReceive[80];  
float fBuf[12];  
WORD DI_counter;  
WORD wBuf[12];  
WORD m_port=3;  
WORD m_address=1;  
WORD m_timeout=100;
```

```
WORD m_checksum=0;
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);
wBuf[0] = m_port;
wBuf[1] = m_address;
wBuf[2] = 0x7050;
wBuf[3] = m_checksum;
wBuf[4] = 100;
wBuf[5] = 0;           // Set the digital input channel No.
wBuf[6] = 0;
DigitalInCounterRead(wBuf, fBuf, szSend, szReceive);
DI_counter=wBuf[7];
Close_Com(COM3);
```

Remark:

■ ClearDigitalInCounter

Description:

This function is used to clear the counter value of the channel number of the digital input module.

Syntax:

```
[ C ]  
WORD ClearDigitalInCounter(WORD wBuf[], float fBuf[],char szSend[],  
                           char szReceive[])
```

Parameter:

wBuf: WORD Input/Output argument talbe
wBuf[0]: [Input] COM port number, from 1 to 255
wBuf[1]: [Input] Module address, form 0x00 to 0xFF
wBuf[2]: [Input] Module ID, 0x7041/44/50/51/52/53/55/58/60/63/65
wBuf[3]: [Input] 0= Checksum disable; 1= Checksum enable
wBuf[4]: [Input] Timeout setting , normal=100 msecond
wBuf[5]: [Input] The digital input channel No.
wBuf[6]: [Input] 0 → no save to szSend &szReceive
 1 → Save to szSend &szReceive
fBuf: Not used.
szSend: [Input] Command string to be sent to I-7000 series modules.
szReceive: [Output] Result string receiving from I-7000 series modules.

Return Value:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 6.7: "Error Code Definitions" for details of other returned values.

Example:

```
char szSend[80];  
char szReceive[80];  
float fBuf[12];  
WORD wBuf[12];  
WORD m_port=3;  
WORD m_address=1;  
WORD m_timeout=100;  
WORD m_checksum=0;  
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);
```

```
wBuf[0] = m_port;  
wBuf[1] = m_address;  
wBuf[2] = 0x7050;  
wBuf[3] = m_checksum;  
wBuf[4] = m_timeout;  
wBuf[5] = 0;           // Set the digital input channel No.  
wBuf[6] = 0;  
ClearDigitalInCounter(wBuf, fBuf, szSend, szReceive);  
Close_Com(COM3);
```

Remark:

■ ReadEventCounter

Description:

This function is used to obtain the value of event counter of I-7000 series modules. This function only supports I-7011, I-7012, I-7014 and I-7016 modules.

Syntax:

```
[ C ]  
WORD ReadEventCounter(WORD wBuf[], float fBuf[],char szSend[],  
                      char szReceive[])
```

Parameter:

wBuf: WORD Input/Output argument table

wBuf[0]: [Input] COM port number, from 1 to 255

wBuf[1]: [Input] Module address, form 0x00 to 0xFF

wBuf[2]: [Input] Module ID, 0x7011/12/14/16

wBuf[3]: [Input] 0= Checksum disable; 1= Checksum enable

wBuf[4]: [Input] Timeout setting , normal=100 msecond

wBuf[5]: Not used

wBuf[6]: [Input] 0 → no save to szSend &szReceive
 1 → Save to szSend &szReceive

wBuf[7]: [Output] The value of event counter

fBuf: Not used.

szSend: [Input] Command string to be sent to I-7000 series modules.

szReceive: [Output] Result string receiving from I-7000 series modules.

Return Value:

0 is for Success
Not 0 is for Failure

Example:

```
char szSend[80];  
char szReceive[80];  
float fBuf[12];  
WORD Counter;  
WORD wBuf[12];  
WORD m_port=3;  
WORD m_address=1;  
WORD m_timeout=100;  
WORD m_checksum=0;
```



```
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);  
wBuf[0] = m_port;  
wBuf[1] = m_address;  
wBuf[2] = 0x7012;  
wBuf[3] = m_checksum;  
wBuf[4] = m_timeout;  
wBuf[6] = 0;  
ReadEventCounter (wBuf, fBuf, szSend, szReceive);  
Counter=wBuf[7];  
Close_Com(COM3);
```

Remark:

■ ClearEventCounter

Description:

This function is used to clear the value of event counter of I-7000 series modules. This function only supports I-7011, I-7012, I-7014 and I-7016 modules.

Syntax:

```
[ C ]  
WORD ClearEventCounter(WORD wBuf[], float fBuf[], char szSend[],  
                        char szReceive[])
```

Parameter:

wBuf: WORD Input/Output argument talbe
wBuf[0]: [Input] COM port number, from 1 to 255
wBuf[1]: [Input] Module address, form 0x00 to 0xFF
wBuf[2]: [Input] Module ID, 0x7011/12/14/16
wBuf[3]: [Input] 0= Checksum disable; 1= Checksum enable
wBuf[4]: [Input] Timeout setting , normal=100 msecond
wBuf[5]: Not used
wBuf[6]: [Input] 0 → no save to szSend &szReceive
 1 → Save to szSend &szReceive
fBuf: Not used.
szSend: [Input] Command string to be sent to I-7000 series modules.
szReceive: [Output] Result string receiving from I-7000 series modules .

Return Value:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 6.7: "Error Code Definitions" for details of other returned values.

Example:

```
char szSend[80];  
char szReceive[80];  
float fBuf[12];  
WORD wBuf[12];  
WORD m_port=3;  
WORD m_address=1;  
WORD m_timeout=100;  
WORD m_checksum=0;  
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);
```

```
wBuf[0] = m_port;  
wBuf[1] = m_address;  
wBuf[2] = 0x7012;  
wBuf[3] = m_checksum;  
wBuf[4] = m_timeout;  
wBuf[6] = 0;  
ClearEventCounter (wBuf, fBuf, szSend, szReceive);  
Close_Com(COM3);
```

Remark:

6.5 Analog Input Functions

6.5.1 I-7000 series modules

■ AnalogIn

Description:

This function is used to obtain input value from I-7000 series modules.

Syntax:

```
[ C ]  
WORD AnalogIn (WORD wBuf[], float fBuf[],char szSend[],char szReceive[])
```

Parameter:

wBuf: WORD Input/Output argument talbe
wBuf[0]: [Input] COM port number, from 1 to 255
wBuf[1]: [Input] Module address, form 0x00 to 0xFF
wBuf[2]: [Input] Module ID, 0x7005/11/12/13/14/15/16/17/18/19/33
wBuf[3]: [Input] 0= Checksum disable; 1= Checksum enable
wBuf[4]: [Input] Timeout setting , normal=100 msecond
wBuf[5]: [Input] Channel number for multi-channel
wBuf[6]: [Input] 0 → no save to szSend & szReceive
 1 → Save to szSend & szReceive
fBuf: Float Input/Ouput argument table.
fBuf[0]: [Output] Analog input value return
szSend: [Input] Command string to be sent to I-7000 series modules.
szReceive: [Output] Result string receiving from I-7000 series modules.

Note: “wBuf[6]” is the debug setting. If this parameter is set as “1”, user can get whole command string and result string from szSend[] and szReceive[] respectively.

Return Value:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 6.7: “Error Code Definitions” for details of other returned values.

Example:

```
float AI;  
float fBuf[12];  
char szSend[80];
```

```
char szReceive[80];
WORD wBuf[12];
WORD m_port=3;
WORD m_address=1;
WORD m_timeout=100;
WORD m_checksum=0;
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);
wBuf[0] = m_port;
wBuf[1] = m_address;
wBuf[2] = 0x7016;
wBuf[3] = m_checksum;
wBuf[4] = m_timeout;
wBuf[5] = 0;
wBuf[6] = 1;
AnalogIn (wBuf, fBuf, szSend, szReceive); // szSend="#02" , szReceive=">+001.9"
AI = fBuf[0]; // AI = 1.9
Close_Com(COM3);
```

Remark:

■ AnalogInHex

Description:

This function is used to obtain the analog input value in “Hexadecimal” form I-7000 series modules.

Syntax:

```
[ C ]  
WORD AnalogInHex (WORD wBuf[], float fBuf[],char szSend[],char szReceive[])
```

Parameter:

- wBuf: WORD Input/Output argument talbe
 - wBuf[0]: [Input] COM port number, from 1 to 255
 - wBuf[1]: [Input] Module address, form 0x00 to 0xFF
 - wBuf[2]: [Input] Module ID, 0x7005/11/12/13/14/15/16/17/18/19/33
 - wBuf[3]: [Input] 0= Checksum disable; 1= Checksum enable
 - wBuf[4]: [Input] Timeout setting , normal=100 msecond
 - wBuf[5]: [Input] Channel number for multi-channel
 - wBuf[6]: [Input] 0 → no save to szSend & szReceive
1 → Save to szSend & szReceive
 - wBuf[7]: [Ouput] The analog input value in “Hexadecimal “ format
 - fBuf: Not used.
 - szSend: [Input] Command string to be sent to I-7000 series modules.
 - szReceive: [Output] Result string receiving from I-7000 series modules.
- Note**: Users have to use DCON utility to set up the analog input configuration of the module in hex format.

Return Value:

- 0: The function was successfully processed.
 - Other: The processing failed.
- Refer to Chapter 6.7: “Error Code Definitions” for details of other returned values.

Example:

```
float A;  
float fBuf[12];  
char szSend[80];  
char szReceive[80];  
WORD wBuf[12];  
WORD m_port=3;  
WORD m_address=1;
```

```
WORD m_timeout=100;
WORD m_checksum=0;
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);
wBuf[0] = m_port;
wBuf[1] = m_address;
wBuf[2] = 0x7012;
wBuf[3] = m_checksum;
wBuf[4] = m_timeout;
wBuf[5] = 0;
wBuf[6] = 1;
AnalogInHex (wBuf, fBuf, szSend, szReceive);
AI = wBuf[7];           // Hex format
Close_Com(COM3);
```

Remark:

■ AnalogInFsr

Description:

This function is used to obtain the analog input value in “FSR” format from I-7000 series modules. The “FSR” means “**Percent**” format.

Syntax:

```
[ C ]  
WORD AnalogInFsr (WORD wBuf[], float fBuf[],char szSend[],char szReceive[])
```

Parameter:

- wBuf: WORD Input/Output argument talbe
- wBuf[0]: [Input] COM port number, from 1 to 255
- wBuf[1]: [Input] Module address, form 0x00 to 0xFF
- wBuf[2]: [Input] Module ID, 0x7005/11/12/13/14/15/16/17/18/19/33
- wBuf[3]: [Input] 0= Checksum disable; 1= Checksum enable
- wBuf[4]: [Input] Timeout setting , normal=100 msecond
- wBuf[5]: [Input] Channel number for multi-channel
- wBuf[6]: [Input] 0 → no save to szSend & szReceive
1 → Save to szSend &szReceive
- fBuf: Float Input/Output argument table.
- fBuf[0]: [Output] Analog input value return
- szSend: [Input] Command string to be sent to I-7000 series modules.
- szReceive: [Output] Result string receiving from I-7000 series modules.

Note: Users have to use DCON utility to set up the analog input configuration of the module in hex format.

Return Value:

- 0: The function was successfully processed.
 - Other: The processing failed.
- Refer to Chapter 6.7: “Error Code Definitions” for details of other returned values.

Example:

```
float A;  
float fBuf[12];  
char szSend[80];  
char szReceive[80];  
WORD wBuf[12];  
WORD m_port=3;  
WORD m_address=1;
```



```
WORD m_timeout=100;
WORD m_checksum=0;
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);
wBuf[0] = m_port;
wBuf[1] = m_address;
wBuf[2] = 0x7012;
wBuf[3] = m_checksum;
wBuf[4] = m_timeout;
wBuf[5] = 0;
wBuf[6] = 1;
AnalogInFsr (wBuf, fBuf, szSend, szReceive);
AI = wBuf[7];
Close_Com(COM3);
```

Remark:

■ AnalogInAll

Description:

This function is used to obtain the analog input value of all channels from I-7000 series modules.

Syntax:

[C]
WORD AnalogInAll (WORD wBuf[], float fBuf[],char szSend[],char szReceive[])

Parameter:

- wBuf: WORD Input/Output argument talbe
- wBuf[0]: [Input] COM port number, from 1 to 255
- wBuf[1]: [Input] Module address, form 0x00 to 0xFF
- wBuf[2]: [Input] Module ID, 0x7005/15/16/17/18/19/33
- wBuf[3]: [Input] 0= Checksum disable; 1= Checksum enable
- wBuf[4]: [Input] Timeout setting , normal=100 msecond
- wBuf[6]: [Input] 0 → no save to szSend & szReceive
1 → Save to szSend & szReceive
- fBuf: Float Input/Output argument table.
- fBuf[0]: [Output] Analog input value return of channel_0
- fBuf[1]: [Output] Analog input value return of channel_1
- fBuf[2]: [Output] Analog input value return of channel_2
- fBuf[3]: [Output] Analog input value return of channel_3
- fBuf[4]: [Output] Analog input value return of channel_4
- fBuf[5]: [Output] Analog input value return of channel_5
- fBuf[6]: [Output] Analog input value return of channel_6
- fBuf[7]: [Output] Analog input value return of channel_7
- szSend: [Input] Command string to be sent to I-7000 series modules.
- szReceive: [Output] Result string receiving from I-7000 series modules.

Note: Users have to use DCON utility to set up the analog input configuration of the module in hex format.

Return Value:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 6.7: "Error Code Definitions" for details of other returned values.

Example:

```
float AI[12];
float fBuf[12];
char szSend[80];
char szReceive[80];
WORD wBuf[12];
WORD m_port=3;
WORD m_address=1;
WORD m_timeout=100;
WORD m_checksum=0;
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);
wBuf[0] = m_port;
wBuf[1] = m_address;
wBuf[2] = 0x7017;
wBuf[3] = m_checksum;
wBuf[4] = m_timeout;
wBuf[6] = 1;
AnalogInAll (wBuf, fBuf, szSend, szReceive);
AI[0] = fBuf[0];
AI[0] = fBuf[1];
AI[0] = fBuf[2];
AI[0] = fBuf[3];
AI[0] = fBuf[4];
AI[0] = fBuf[5];
AI[0] = fBuf[6];
AI[0] = fBuf[7];
Close_Com(COM3);
```

Remark:

■ ThermocoupleOpen_7011

Description:

This function is used to detect the thermocouple state of I-7011 modules for the supporting type “J, K, T, E, R, S, B, N, C” is open or close. If the response value is “0”, thermocouple I-7011 is working in close state. If the response value is “1”, thermocouple I-7011 is working in open state. For more information please refer to user manual.

Syntax:

```
[ C ]  
WORD ThermocoupleOpen_7011(WORD wBuf[], float fBuf[],char szSend[],  
                             char szReceive[])
```

Parameter:

wBuf: WORD Input/Output argument talbe
wBuf[0]: [Input] COM port number, from 1 to 255
wBuf[1]: [Input] Module address, form 0x00 to 0xFF
wBuf[2]: [Input] Module ID, 0x7011
wBuf[3]: [Input] 0= Checksum disable; 1= Checksum enable
wBuf[4]: [Input] Timeout setting , normal=100 msecond
wBuf[5]: [Output] response value 0 → the thermocouple is close
 response value 1 → the thermocouple is open
wBuf[6]: [Input] 0 → no save to szSend & szReceive
 1 → Save to szSend & szReceive
fBuf: Not used.
szSend: [Input] Command string to be sent to I-7000 series modules.
szReceive: [Output] Result string receiving from I-7000 series modules.

Return Value:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 6.7: “Error Code Definitions” for details of other returned values.

Example:

```
WORD state;  
float fBuf[12];  
char szSend[80];  
char szReceive[80];  
WORD wBuf[12];
```

```
WORD m_port=3;
WORD m_address=1;
WORD m_timeout=100;
WORD m_checksum=0;
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);
wBuf[0] = m_port;
wBuf[1] = m_address;
wBuf[2] = 0x7011;
wBuf[3] = m_checksum;
wBuf[4] = m_timeout;
wBuf[5] = 0;
wBuf[6] = 1;
ThermocoupleOpen_7011(wBuf, fBuf, szSend, szReceive);
state = wBuf[5];
Close_Com(COM3);
```

Remark:

■ SetLedDisplay

Description:

This function is used to configure LED display for specified channel of I-7000 analog input serial modules.

Syntax:

[C]
WORD SetLedDisplay (WORD wBuf[], float fBuf[],char szSend[], char szReceive[])

Parameter:

wBuf: WORD Input/Output argument talbe
wBuf[0]: [Input] COM port number, from 1 to 255
wBuf[1]: [Input] Module address, form 0x00 to 0xFF
wBuf[2]: [Input] Module ID, 0x7013/16/33
wBuf[3]: [Input] 0= Checksum disable; 1= Checksum enable
wBuf[4]: [Input] Timeout setting , normal=100 msecond
wBuf[5]: [Input] Set display channel
wBuf[6]: [Input] 0 → no save to szSend & szReceive
 1 → Save to szSend & szReceive
fBuf: Not used.
szSend: [Input] Command string to be sent to I-7000 series modules.
szReceive: [Output] Result string receiving from I-7000 series modules.

Return Value:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 6.7: “Error Code Definitions” for details of other returned values.

Example:

```
float fBuf[12];  
char szSend[80];  
char szReceive[80];  
WORD wBuf[12];  
WORD m_port=3;  
WORD m_address=1;  
WORD m_timeout=100;  
WORD m_checksum=0;  
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);  
wBuf[0] = m_port;
```

```
wBuf[1] = m_address;  
wBuf[2] = 0x7033;  
wBuf[3] = m_checksum;  
wBuf[4] = m_timeout;  
wBuf[5] = 1;           // Set channel 1 display  
wBuf[6] = 1;  
SetLedDisplay (wBuf, fBuf, szSend, szReceive);  
Close_Com(COM3);
```

Remark:

■ GetLedDisplay

Description:

This function is used to get the current setting of the specified channel for LED display channel for specified channel of I-7000 analog input serial modules.

Syntax:

[C]
WORD GetLedDisplay (WORD wBuf[], float fBuf[],char szSend[], char szReceive[])

Parameter:

wBuf: WORD Input/Output argument talbe
wBuf[0]: [Input] COM port number, from 1 to 255
wBuf[1]: [Input] Module address, form 0x00 to 0xFF
wBuf[2]: [Input] Module ID, 0x7013/16/33
wBuf[3]: [Input] 0= Checksum disable; 1= Checksum enable
wBuf[4]: [Input] Timeout setting , normal=100 msecond
wBuf[5]: [Output] Current channel for LED display
 0 = channel_0
 1 = channel_1
wBuf[6]: [Input] 0 → no save to szSend & szReceive
 1 → Save to szSend & szReceive
fBuf: Not used
szSend: [Input] Command string to be sent to I-7000 series modules.
szReceive: [Output] Result string receiving from I-7000 series modules.

Return Value:

0 is for Success
Not 0 is for Failure

Example:

```
WORD led;  
float fBuf[12];  
char szSend[80];  
char szReceive[80];  
WORD wBuf[12];  
WORD m_port=3;  
WORD m_address=1;  
WORD m_timeout=100;  
WORD m_checksum=0;
```



```
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);  
wBuf[0] = m_port;  
wBuf[1] = m_address;  
wBuf[2] = 0x7033;  
wBuf[3] = m_checksum;  
wBuf[4] = m_timeout;  
wBuf[6] = 1;  
GetLedDisplay (wBuf, fBuf, szSend, szReceive);  
Led = wBuf[5];  
Close_Com(COM3);
```

Remark:

6.6 Analog Output Functions

6.6.1 I-7000 series modules

■ AnalogOut

Description:

This function is used to obtain analog value from analog output module of I-7000 series modules.

Syntax:

```
[ C ]  
WORD AnalogOut(WORD wBuf[], float fBuf[],char szSend[], char szReceive[])
```

Parameter:

wBuf: WORD Input/Output argument talbe
wBuf[0]: [Input] COM port number, from 1 to 255
wBuf[1]: [Input] Module address, form 0x00 to 0xFF
wBuf[2]: [Input] Module ID, 0x7016/21/22/24
wBuf[3]: [Input] 0= Checksum disable; 1= Checksum enable
wBuf[4]: [Input] Timeout setting , normal=100 msecond
wBuf[5]: [Input] The analog output channel number
wBuf[6]: [Input] 0 → no save to szSend &szReceive
 1 → Save to szSend &szReceive
fBuf: Float Input/Ouput argument table.
fBuf[0]: [Input] Analog output value
szSend: [Input] Command string to be sent to I-7000 series modules.
szReceive: [Output] Result string receiving from I-7000 series modules.

Return Value:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 6.7: "Error Code Definitions" for details of other returned values.

Example:

```
float fBuf[12];  
char szSend[80];  
char szReceive[80];  
WORD wBuf[12];  
WORD m_port=3;
```

```
WORD m_address=1;
WORD m_timeout=100;
WORD m_checksum=0;
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);
wBuf[0] = m_port;
wBuf[1] = m_address;
wBuf[2] = 0x7016;
wBuf[3] = m_checksum;
wBuf[4] = m_timeout;
// wBuf[5] = 0;           // I-7016 no used
wBuf[6] = 1;
fBuf[0] = 3.5           // Excitation Voltage output +3.5V
AnalogOut (wBuf, fBuf, szSend, szReceive); "
Close_Com(COM3);
```

Remark:

■ AnalogOutReadBack

Description:

This function is used to obtain read back the analog value of analog output modules of I-7000 series modules. There are two types of reading back functions, as described in the following:

1. Last value is read back by \$AA6 command.
2. The analog output of current path is read back by \$AA8 command.

Syntax:

```
[ C ]  
WORD AnalogOutReadBack(WORD wBuf[], float fBuf[],char szSend[],  
                        char szReceive[])
```

Parameter:

- wBuf: WORD Input/Output argument table
- wBuf[0]: [Input] COM port number, from 1 to 255
- wBuf[1]: [Input] Module address, from 0x00 to 0xFF
- wBuf[2]: [Input] Module ID, 0x7016/21/22/24
- wBuf[3]: [Input] 0= Checksum disable; 1= Checksum enable
- wBuf[4]: [Input] Timeout setting , normal=100 msecond
- wBuf[5]: [Input] 0: command \$AA6 read back
1: command \$AA8 read back
- Note** 1) When the module is I-7016: Don't care.
2) When the module is I-7021/22, analog output of current path read back (\$AA8)
3) When the module is I-7024, the updating value in a specific Slew rate (\$AA8)
(For more information, please refer to I-7021/22/24 manual)
- wBuf[6]: [Input] 0 → no save to szSend &szReceive
1 → Save to szSend &szReceive
- wBuf[7]: [Input] The analog output channel No. (0 to 3) of module I-7024
No used for single analog output module
- fBuf: Float Input/Output argument table.
- fBuf[0]: [Output] Analog output read back value
- szSend: [Input] Command string to be sent to I-7000 series modules.
- szReceive: [Output] Result string receiving from I-7000 series modules.

Return Value:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 6.7: “Error Code Definitions” for details of other returned values.

Example:

```
Float Volt;
float fBuf[12];
char szSend[80];
char szReceive[80];
WORD wBuf[12];
WORD m_port=3;
WORD m_address=1;
WORD m_timeout=100;
WORD m_checksum=0;
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);
wBuf[0] = m_port;
wBuf[1] = m_address;
wBuf[2] = 0x7021;
wBuf[3] = m_checksum;
wBuf[4] = m_timeout;
wBuf[5] = 0;                // $AA6 command
wBuf[6] = 1;
wBuf[7] = 1;
AnalogOutReadBack (wBuf, fBuf, szSend, szReceive);
Volt = fBuf[0];            // Receive: “!01+2.57” excitation voltage , Volt=2.57
Close_Com(COM3);
```

Remark:

■ AnalogOutHex

Description:

This function is used to obtain analog value of the analog output modules through Hex format.

Syntax:

[C]

`WORD AnalogOutHex(WORD wBuf[], float fBuf[],char szSend[], char szReceive[])`

Parameter:

wBuf: WORD Input/Output argument talbe

wBuf[0]: [Input] COM port number, from 1 to 255

wBuf[1]: [Input] Module address, form 0x00 to 0xFF

wBuf[2]: [Input] Module ID, 0x7021/21P/22

wBuf[3]: [Input] 0= Checksum disable; 1= Checksum enable

wBuf[4]: [Input] Timeout setting , normal=100 msecond

wBuf[5]: [Input] The analog output channel number
(No used for single analog output module)

wBuf[6]: [Input] 0 → no save to szSend &szReceive
1 → Save to szSend &szReceive

wBuf[7]: [Input] Analog output value in Hexadecimal data format

fBuf: Not used.

szSend: [Input] Command string to be sent to I-7000 series modules.

szReceive: [Output] Result string receiving from I-7000 series modules.

Return Value:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 6.7: "Error Code Definitions" for details of other returned values.

Example:

```
float fBuf[12];
char szSend[80];
char szReceive[80];
WORD wBuf[12];
WORD m_port=3;
WORD m_address=1;
WORD m_timeout=100;
WORD m_checksum=0;
```

```
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);
wBuf[0] = m_port;
wBuf[1] = m_address;
wBuf[2] = 0x7022;
wBuf[3] = m_checksum;
wBuf[4] = m_timeout;
wBuf[5] = 1;           // channel 1
wBuf[6] = 1;
wBuf[7] = 0x250
AnalogOutHex (wBuf, fBuf, szSend, szReceive);
Close_Com(COM3);
```

Remark:

■ AnalogOutFsr

Description:

This function is used to obtain the analog value of analog output modules through % of span data format. This function only can be used after analog output modules is set as “FSR” output mode.

Syntax:

```
[ C ]  
WORD AnalogOutFsr(WORD wBuf[], float fBuf[],char szSend[], char szReceive[])
```

Parameter:

wBuf: WORD Input/Output argument talbe
wBuf[0]: [Input] COM port number, from 1 to 255
wBuf[1]: [Input] Module address, form 0x00 to 0xFF
wBuf[2]: [Input] Module ID, 0x7021/21P/22
wBuf[3]: [Input] 0= Checksum disable; 1= Checksum enable
wBuf[4]: [Input] Timeout setting , normal=100 msecond
wBuf[5]: [Input] The analog output channel number
(No used for single analog output module)
wBuf[6]: [Input] 0 → no save to szSend &szReceive
1 → Save to szSend &szReceive
fBuf: Float Input/Output argument table.
FBuf[0]: [Input] Analog output value in % of Span data format.
szSend: [Input] Command string to be sent to I-7000 series modules.
szReceive: [Output] Result string receiving from I-7000 series modules.

Return Value:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 6.7: “Error Code Definitions” for details of other returned values.

Example:

```
float fBuf[12];  
char szSend[80];  
char szReceive[80];  
WORD wBuf[12];  
WORD m_port=3;  
WORD m_address=1;
```



```
WORD m_timeout=100;
WORD m_checksum=0;
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);
wBuf[0] = m_port;
wBuf[1] = m_address;
wBuf[2] = 0x7022;
wBuf[3] = m_checksum;
wBuf[4] = m_timeout;
wBuf[5] = 1;           // channel 1
wBuf[6] = 1;
fBuf[0] = 50
AnalogOutFsr (wBuf, fBuf, szSend, szReceive);
Close_Com(COM3);
```

Remark:

■ AnalogOutReadBackHex

Description:

This function is used to obtain read back the analog value of analog output modules in Hex format for I-7000 series modules. There are two types of reading back functions, as described in the following:

1. Last value is read back by \$AA6 command.
2. The analog output of current path is read back by \$AA8 command.

Syntax:

```
[ C ]  
WORD AnalogOutReadBackHex(WORD wBuf[], float fBuf[],char szSend[],  
                           char szReceive[])
```

Parameter:

wBuf: WORD Input/Output argument talbe
wBuf[0]: [Input] COM port number, from 1 to 255
wBuf[1]: [Input] Module address, form 0x00 to 0xFF
wBuf[2]: [Input] Module ID, 0x7021/21P/22
wBuf[3]: [Input] 0= Checksum disable; 1= Checksum enable
wBuf[4]: [Input] Timeout setting , normal=100 msecond
wBuf[5]: [Input] 0: command \$AA6 read back
 1: command \$AA8 read back
wBuf[6]: [Input] 0 → no save to szSend &szReceive
 1 → Save to szSend &szReceive
wBuf[7]: [Input] The analog output channel No.
 No used for single analog output module
wBuf[9]: [Output] Analog output value in Hexadecimal data format.
fBuf: Not used.
szSend: [Input] Command string to be sent to I-7000 series modules.
szReceive: [Output] Result string receiving from I-7000 series modules.

Return Value:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 6.7: "Error Code Definitions" for details of other returned values.

Example:

```
WORD Volt;
float fBuf[12];
char szSend[80];
char szReceive[80];
WORD wBuf[12];
WORD m_port=3;
WORD m_address=1;
WORD m_timeout=100;
WORD m_checksum=0;
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);
wBuf[0] = m_port;
wBuf[1] = m_address;
wBuf[2] = 0x7021;
wBuf[3] = m_checksum;
wBuf[4] = m_timeout;
wBuf[5] = 0;                // command $AA6
wBuf[6] = 1;
wBuf[7] = 0;
AnalogOutReadBackHex (wBuf, fBuf, szSend, szReceive);
Volt = wBuf[9];
Close_Com(COM3);
```

Remark:

■ AnalogOutReadBackFsr

Description:

This function is used to obtain read back the analog value of analog output modules through % of span data format for I-7000 series modules. There are two types of reading back functions, as described in the following:

1. Last value is read back by \$AA6 command.
2. The analog output of current path is read back by \$AA8 command.

Syntax:

```
[ C ]  
WORD AnalogOutReadBackFsr(WORD wBuf[], float fBuf[],char szSend[],  
                           char szReceive[])
```

Parameter:

wBuf: WORD Input/Output argument talbe
wBuf[0]: [Input] COM port number, from 1 to 255
wBuf[1]: [Input] Module address, form 0x00 to 0xFF
wBuf[2]: [Input] Module ID, 0x7021/21P/22
wBuf[3]: [Input] 0= Checksum disable; 1= Checksum enable
wBuf[4]: [Input] Timeout setting , normal=100 msecond
wBuf[5]: [Input] 0: command \$AA6 read back
 1: command \$AA8 read back
wBuf[6]: [Input] 0 → no save to szSend &szReceive
 1 → Save to szSend &szReceive
wBuf[7]: [Input] The analog output channel No.
 No used for single analog output module
fBuf: Float input/output argument table.
fBuf[0]: [Output] Analog output value in % Span data format.
szSend: [Input] Command string to be sent to I-7000 series modules.
szReceive: [Output] Result string receiving from I-7000 series modules.

Return Value:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 6.7: “Error Code Definitions” for details of other returned values.

Example:

```
float Volt;
float fBuf[12];
char szSend[80];
char szReceive[80];
WORD wBuf[12];
WORD m_port=3;
WORD m_address=1;
WORD m_timeout=100;
WORD m_checksum=0;
Open_Com(COM3, 9600, Data8Bit, NonParity, OneStopBit);
wBuf[0] = m_port;
wBuf[1] = m_address;
wBuf[2] = 0x7021;
wBuf[3] = m_checksum;
wBuf[4] = m_timeout;
wBuf[5] = 0;                // command $AA6
wBuf[6] = 1;
wBuf[7] = 0;
AnalogOutReadBackFsr (wBuf, fBuf, szSend, szReceive);
Volt = fBuf[0];
Close_Com(COM3);
```

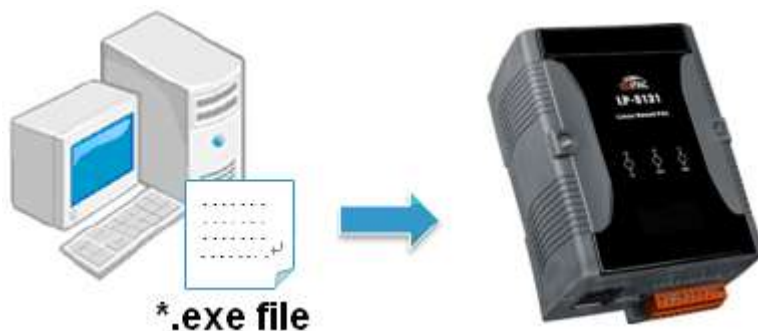
Remark:

6.7 Error Code Explanation

Error Code	Explanation
0	NoError
1	FunctionError
2	PortError
3	BaudrateError
4	DataError
5	StopError
6	ParityError
7	ChecksumError
8	ComPortNotOpen
9	SendThreadCreateError
10	SendCmdError
11	ReadComStatusError
12	StrCheck Error
13	CmdError
14	X
15	TimeOut
16	X
17	ModuleId Error
18	AdChannelError
19	UnderRang
20	ExceedRange
21	InvalidateCounterValue
22	InvalidateCounterValue
23	InvalidateGateMode
24	InvalidateChannelNo
25	ComPortInUse

7. Demo for LP-51xx Modules With C Language

In this section, we will focus on examples for the description and application of the control functions on the I-7000/I-8000/I-87k series modules for use in the LP-51xx. After installing the LP-51xx SDK, the demo programs provided below can be found in the “**c:/cygwin/LinCon8k/examples**” folder.



7.1 DIO Control Demo for I-7k Modules

The **i7kdio.c** demo application illustrates how to control DI/DO function using an I-7050 module (8 DO channels and 7 DI channels) connected to an RS-485 network. The address of the module is 02 and the Baud Rate is 9600 bps.

The result of executing this demo program is that DO channels 0 to 7 on the I-7050 module will be set as the output channels, and DI channel 2 on the I-7050 module will be set as the input channel. The source code for the demo program is as follows:

```
#include<stdio.h>
#include<stdlib.h>
#include "msw.h"

char szSend[80], szReceive[80], ans;
WORD wBuf[12];
float fBuf[12];
/* ----- */
int main()
{
    int wRetVal;
    // Check Open_Com2
    wRetVal = Open_Com(COM2, 9600, Data8Bit, NonParity, OneStopBit);
```

```

if (wRetVal > 0) {
    printf("open port failed!\n");
    return (-1);
}

// ***** 7050 DO && DI Parameter *****
wBuf[0] = 2;           // COM Port
wBuf[1] = 0x02;       // Address
wBuf[2] = 0x7050;     // ID
wBuf[3] = 0;          // CheckSum disable
wBuf[4] = 100;        // TimeOut , 100 msecond
wBuf[5] = 0x0f;       // 8 DO Channels On
wBuf[6] = 0;          // string debug
// 7050 DO Output
wRetVal = DigitalOut(wBuf, fBuf, szSend, szReceive);
if (wRetVal)
    printf("DigitalOut_7050 Error !, Error Code=%d\n", wRetVal);

printf("The DO of 7050 : %u \n", wBuf[5]);

// 7050 DI Input
DigitalIn(wBuf, fBuf, szSend, szReceive);
printf("The DI of 7050 : %u \n", wBuf[5]);

Close_Com(COM2);
return 0;
}

```

Follow the procedure below to achieve the desired results:

Step 1: Write i7kdio.c

Copy the above source code above to a blank text file and save it using the name – i7kdio.c or open the file from the C:\cygwin\LinCon8k\examples\i7k folder.

Step 2: Compile i7kdio.c to an executable file – i7kdio.exe

Two methods can be used to compile the program, each of which is introduced here:

Method One – Using a Batch File (lcc.bat)

Open the LP-51xx Build Environment by clicking the Start > Programs > ICPDAS > LP-51xx SDK > LP-51xx Build Environment to open **LP-51xx SDK** window, and change the path to C:\cygwin\LinCon8k\examples\i7k. To compile the i7kdio.c file to an executable file, type **lcc i7kdio**. (refer to Fig. 7-1)


```

c:\ LinPAC-51XX Build Environment
C:\Documents and Settings\Edward\Desktop>CMD.EXE /k c:\cygwin\lincon8k\setenv.bat
----- LinPAC-51XX SDK Environment Configure -----
Target      :ICPDAS          (Arm based)
Work Directory :C:\Cygwin\LinCon8k
C:\cygwin\LinCon8k>cd examples/i7k
C:\cygwin\LinCon8k\examples\i7k>gcc i7kdio
Compile ok!
C:\cygwin\LinCon8k\examples\i7k>dir/w
Volume in drive C has no label.
Volume Serial Number is 6CF3-2221
Directory of C:\cygwin\LinCon8k\examples\i7k
[.]          [..]          i7kaio.c      i7kaio.exe   i7kdio.c     i7kdio.exe
              4 File(s)          549,049 bytes
              2 Dir(s)  13,700,902,912 bytes free
C:\cygwin\LinCon8k\examples\i7k>

```

Fig. 7-1

Method Two – Using Compile Instructions

When using this method, type `cd C:\cygwin\LinCon8k\examples\i7k` command prompt to change the path. To compile `i7kdio.c` to an executable file, type `arm-linux-gcc -I../include -lm -o i7kdio.exe i7kdio.c ../lib/libi8k.a` (refer to Fig. 7-2).

```

c:\ LinPAC-51XX Build Environment
C:\Documents and Settings\Edward\Desktop>CMD.EXE /k c:\cygwin\lincon8k\setenv.bat
----- LinPAC-51XX SDK Environment Configure -----
Target      :ICPDAS          (Arm based)
Work Directory :C:\Cygwin\LinCon8k
C:\cygwin\LinCon8k>cd examples/i7k
C:\cygwin\LinCon8k\examples\i7k>arm-linux-gcc -I../include -lm -o i7kdio.exe
i7kdio.c ../lib/libi8k.a
C:\cygwin\LinCon8k\examples\i7k>dir/w
Volume in drive C has no label.
Volume Serial Number is 6CF3-2221
Directory of C:\cygwin\LinCon8k\examples\i7k
[.]          [..]          i7kaio.c      i7kaio.exe   i7kdio.c     i7kdio.exe
              4 File(s)          549,049 bytes
              2 Dir(s)  13,700,501,504 bytes free
C:\cygwin\LinCon8k\examples\i7k>

```

Fig. 7-2

Step 3: Transfer i7kdio.exe to the LP-51xx

Two methods can be used to transfer the executable file to the LP-51xx, each of which is introduced here.

Method One – Using an FTP application

(1) Open a FTP application and create a new FTP connection. Enter the login details for the LP-51xx, including the Host name (or IP address), Username and Password. The default value for both the **User_Name** and the **Password** is “**root**”. Click the “**Connect**” button to connect to the FTP server on the LP-51xx. Refer to Fig.7-3 below for more details.

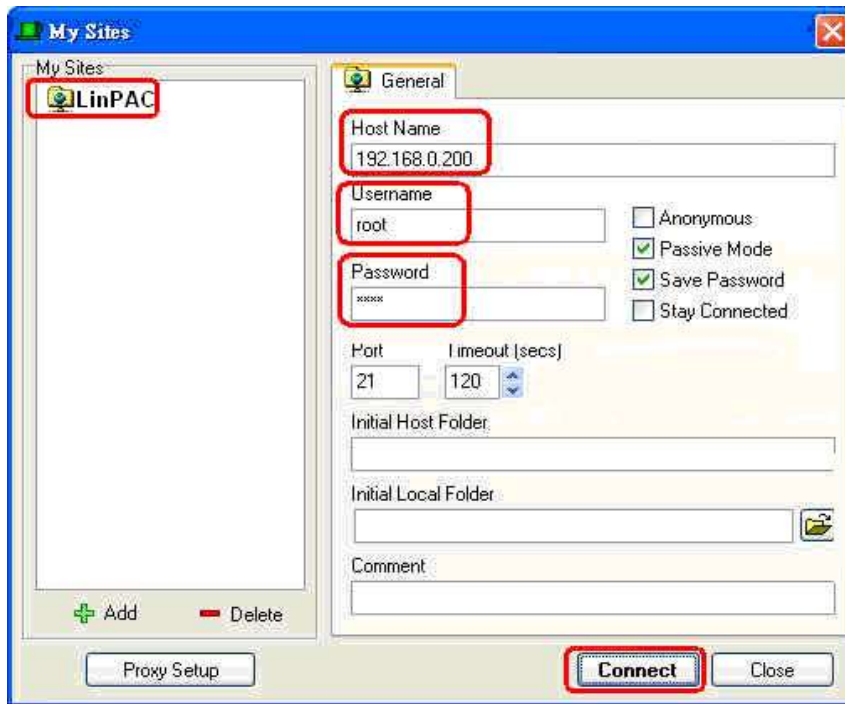


Fig.7-3

(2) Upload the file **i7kdio.exe** file to the LP-51xx. (refer to Fig.7-4).

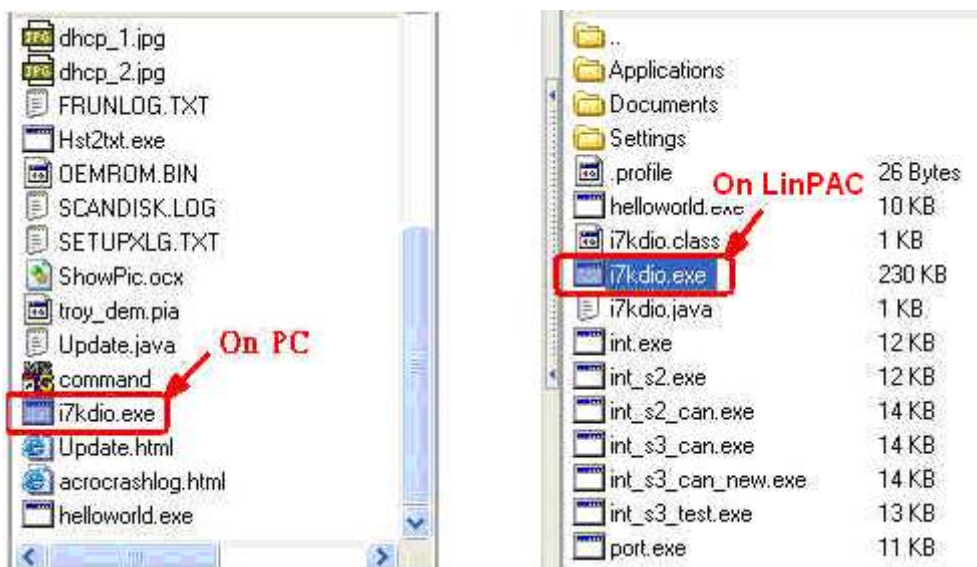


Fig.7-4

(3) Choose `i7kdio.exe` in the LP-51xx and Click the right mouse button to select the “Permissions” option for the menu. Enter “777” in the Numeric textbox to set the file permissions to readable, writeable, and executable. Refer to Fig.7-5 and 7-6 below.

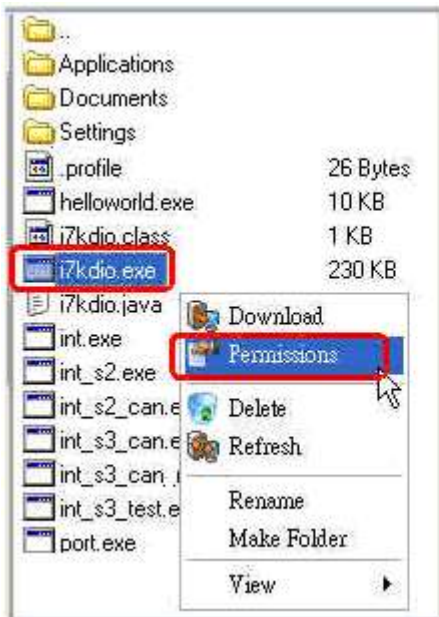


Fig.7-5



Fig.7-6

Method Two – Using a DOS Command Prompt

Open DOS Command Prompt and enter the IP Address of the server on the LP-51xx in order to connect to the FTP server of the LP-51xx. Enter the **User Name** and **Password** (the default value is root) to login to the LP-51xx FTP server.

Files must be transferred in binary mode, so type “bin” to set the mode.

At Command Prompt, type put c:/cygwin/lincon8k/examples/i7k/i7kdio.exe i7kdio.exe to transfer the `i7kdio.exe` file to the LP-51xx. Once the file has been transferred, the “Transfer complete” message will be displayed. Refer to Fig. 7-7 below for more details.

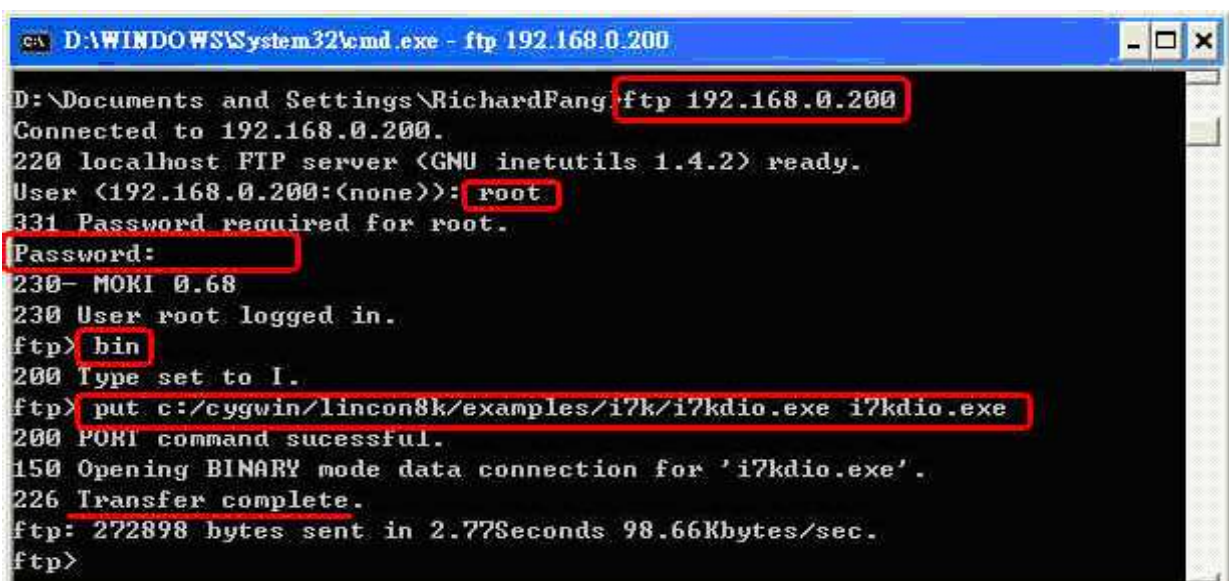


Fig. 7-7

Step 4: Use Telnet to the LP-51xx to execute i7kdio.exe

At the Command Prompt, type telnet IP Address of the LP-51xx to establish a connection to the LP-51xx. Enter **User Name** and **Password** (the default value is **root**) to login to the LP-51xx.

At Command Prompt, type chmod 777 i7kdio.exe to set the i7kdio.exe file to executable, and then type i7kdio.exe to execute the i7kdio.exe file. Refer to Figs. 7-8 and 7-9 below for more details.

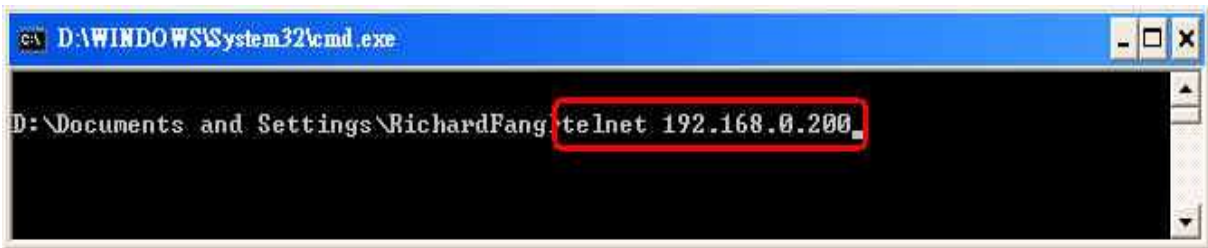


Fig. 7-8

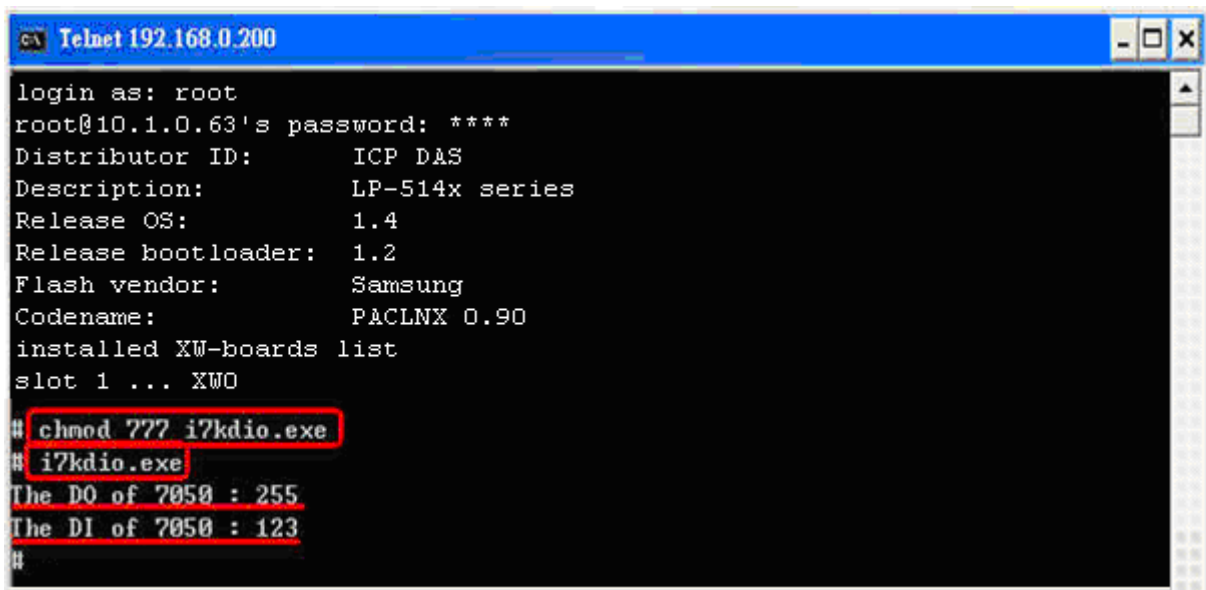


Fig. 7-9

The message "**The DO of I-7050: 255** ($=2^8 - 1$)" indicates that DO channels 0 to 7 will be used to output data, and the message "**The DI of I-7050: 123** ($=127 - 2^2$)" indicates that DI channel 2 will be used as the input channel.

7.2 AIO Control Demo for I-7k Modules

The **i7kaio.c** demo application illustrates how to control the AI/AO functions using an I-7017 module (8 AI channels) and an I-7021 module (1 AO channel) connected to an RS-485 network. The addresses for the I-7021 and I-7017 modules are 05 and 03, respectively, and the baud rate for both modules is 9600 bps.

The result of executing this demo program is that the AO channel on the I-7021 module will be set to output a voltage of 3.5V, and AI channel 2 on the I-7017 module will be set as the input channel. The source code for this demo program is as follows:

```
#include<stdio.h>
#include<stdlib.h>
#include "msw.h"

char szSend[80], szReceive[80];
WORD wBuf[12];
float fBuf[12];
/* ----- */

int main()
{
    int i,j, wRetVal;
    DWORD temp;

    wRetVal = Open_Com(COM2, 9600, Data8Bit, NonParity, OneStopBit);
    if (wRetVal > 0) {
        printf("open port failed!\n");
        return (-1);
    }
    //--- Analog output ----   ****   7021 -- AO   ****
    i = 0;
    wBuf[0] = 2;                // COM Port
    wBuf[1] = 0x05;             // Address
    wBuf[2] = 0x7021;           // ID
    wBuf[3] = 0;                // CheckSum disable
    wBuf[4] = 100;              // TimeOut , 100 msecond
    //wBuf[5] = i;              // Not used if module ID is 7016/7021
                                // Channel No.(0 to 1) if module ID is 7022
                                // Channel No.(0 to 3) if module ID is 7024
}
```

```

wBuf[6] = 0;           // string debug
fBuf[0] = 3.5;        // Analog Value
wRetVal = AnalogOut(wBuf, fBuf, szSend, szReceive);
if (wRetVal)         // There was an error with the Analog Output on the I-7021
    printf("AO of 7021 Error !, Error Code=%d\n", wRetVal);
else
    printf("AO of 7021 channel %d = %f \n",i,fBuf[0]);

//--- Analog Input ----  ****   7017 -- AI   ****
j = 1;
wBuf[0] = 2;         // COM Port
wBuf[1] = 0x03;     // Address
wBuf[2] = 0x7017;   // ID
wBuf[3] = 0;        // CheckSum disable
wBuf[4] = 100;     // TimeOut , 100 msecond
wBuf[5] = j;       // Channel of AI
wBuf[6] = 0;       // string debug
wRetVal = AnalogIn(wBuf, fBuf, szSend, szReceive);

if (wRetVal)         // There was an error with the Analog Input on the I-7017
    printf("AI of 7017 Error !, Error Code=%d\n", wRetVal);
else
    printf("AI of 7017 channel %d = %f \n",j,fBuf[0]);

Close_Com(COM2);
return 0;
}

```

For this example, the programming and execution procedures are the same as those described in section 7.1. Fig. 7-10 below illustrates the result of the execution.

```

Telnet 192.168.0.200
login as: root
root@10.1.0.63's password: ****
Distributor ID:      ICP DAS
Description:         LP-514x series
Release OS:          1.4
Release bootloader:  1.2
Flash vendor:        Samsung
Codename:            PACLNX 0.90
installed XW-boards list
slot 1 ... XWO
# i7kaio.exe
AO of 7021 channel 0 = 3.500000
AI of 7012 channel 1 = 3.500000

```

Fig. 7-10

7.3 Overview of the Module Control Demo Program

Fig. 7-11 provides a summary of the various communication functions that can be used depending on the for the different locations of the I-7000/I-8000/I-87000 modules when using the ICP DAS modules in conjunction with the LP-51xx, which can be helpful in understanding which communication functions should be used.

Module Location Communication Functions	I-87k in Expansion Unit	I-8k or I-87k in I-8000 Controller	I-7k
Open_Com()	✓	✓	✓
Close_Com()	✓	✓	✓

Fig. 7-11

Fig. 7-12 provides an overview of the source files from the libi8k.a library that can be used depending on the different locations of I-7000/I-8000/I-87000 modules when using ICP DAS modules in conjunction with the LP-51xx, which can be helpful in understanding which source files should be called.

Module Location Source File	I7000.c	I8000.c	I87000.c
I-7K	✓		
I-8K or I-87K in I-8000 Controller		✓	
I-87K in Expansion Unit			✓

Fig. 7-12

7.4 Timer Function Demo

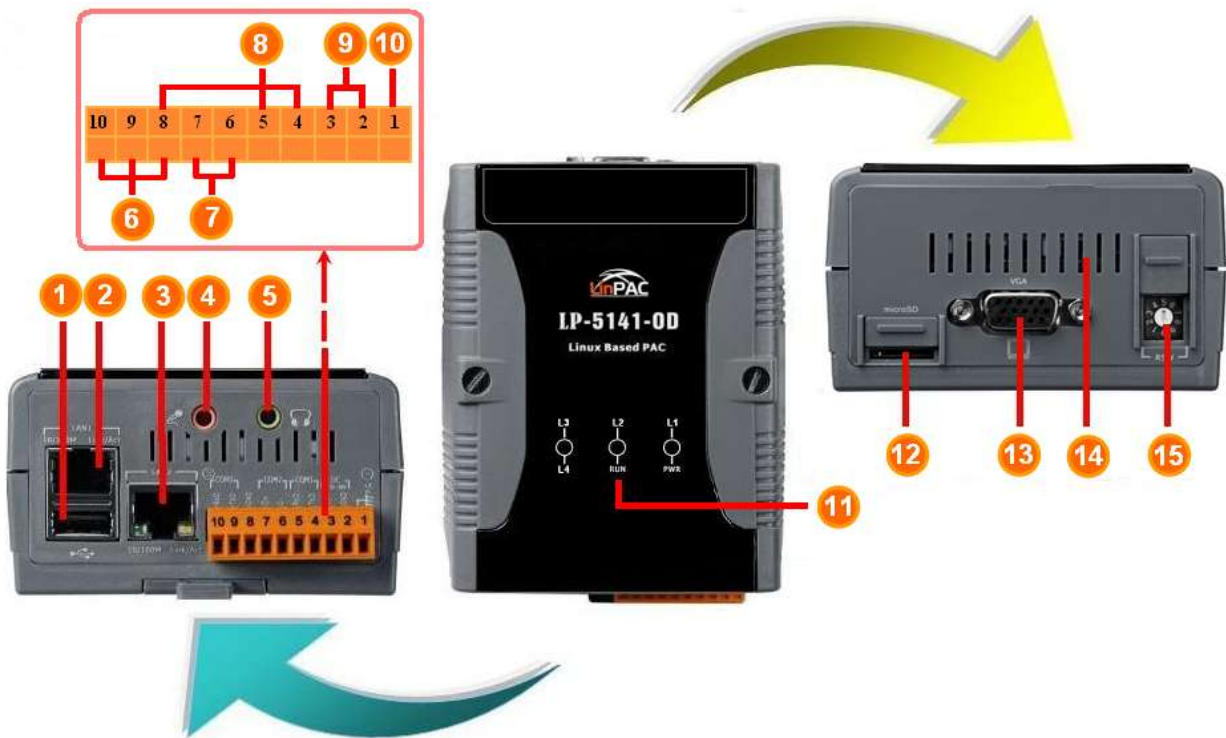
For an example of how to use the “**Timer**” function in conjunction with the LP-51xx, refer to the – [timer.c](#) and [time2.c](#) demo programs provided in the LinPAC SDK – (C:\cygwin\LinCon8k\examples\common) folder. The [timer.c](#) program can be used when the execution period is between 0.5 to 10 ms (Real-Time) and the [timer2.c](#) program can be used when the execution is period greater than 10 ms (General).

8. Overview of the Serial Ports on the LP-51xx

The following is a description of the three serial ports contained in the LP-51xx embedded controller, and are based on the RS-232 or RS-485 interfaces. Fig 8-1 illustrates the ports contained on the LP-5141. The information in this section is organized as follows:

- **COM1 Port** – Internal communication with the XW-board modules
- **COM2 Port** – RS-485 (D2+,D2-) ; 2500V_{DC} isolation
- **COM3 Port** – RS-232 (RXD, TXD, and GND); Non-isolation
- **Console Port** – RS-232 (RXD, TXD, and GND); Non-isolation (For console)

COM port	Definitions in LP-51xx SDK	Device name	Default baudrate
None	COM1	None	115200
1 (RS-232/console)	None	ttySA0	115200
2 (RS-485)	COM2	ttyS0	9600
3 (RS-232)	COM3	ttyS1	115200



1	USB Port	6	COM 1 (RS-232)	11	LED Indicator
2	Ethernet Port 1	7	COM 2 (RS-485)	12	microSD socket
3	USB Port	8	COM 3 (RS-232)	13	VGA Port
4	Microphone-In	9	Power	14	XWboard (optional)
5	Earphone-Out	10	Frame Ground	15	Operating Modes Selector

Fig. 8-1

Use the **stty** command to query or configure the COM port. For example, to modify the baud rate 9600 to 115200 bps via COM2 port:

```
# stty -F /dev/ttyS0 ispeed 115200 ospeed 115200
```


8.1 COM1 Port

COM1 is the internal I/O expansion port on the LP-51xx and is used to connect to an I-87KW series module inserted into the LP-51xx embedded controller. A serial command must be used to control the I-87KW series module.

To control the I-87KW module, the Com port parameters and call the **Open_Com()** function to open the **COM1** port based on the appropriate settings.

Finally, call the **ChangeToSlot(slot)** function to specify which slot will be controlled. This is like the serial address, meaning that control commands can be sent to an I/O module that is inserted in the specified slot. Consequently, the serial address for the slot that contains the module is 0. A detailed example is provided below:

For Example:

```
int slot=1;
unsigned char port=1;           // for all modules in COM1 port of LP-51xx
DWORD baudrate=115200;
char data=8, parity=0, stopbit=1 ;
Open_Slot(slot);
Open_Com(port, baudrate, data, parity, stopbit);
// send command...
Close_Com(port);
Close_Slot(slot);
```

8.2 COM2 Port

This COM2 port provides **RS-485** serial communication functionality (DATA+ and DATA-) and is located on the bottom-right corner on the LP-51xx. This port allows a connection to be made to modules that contain an RS-485 interface such as the I-7000 serial modules (DCON Module), meaning that ICP DAS I-7000 series modules can be directly controlled via this port with any converter. ICP DAS provides a very easy to use the library of functions (libi8k.a) that can use to easily communicate with I-7000 series modules. Below is an application example of the program code demo.

- ❑ Test using C language:

```
unsigned char port=2; data=8, parity=0, stopbit=1;
DWORD baudrate=9600;
Open_Com(port, baudrate, data, char parity, stopbit);
// send command...
Close_Com(port);
```

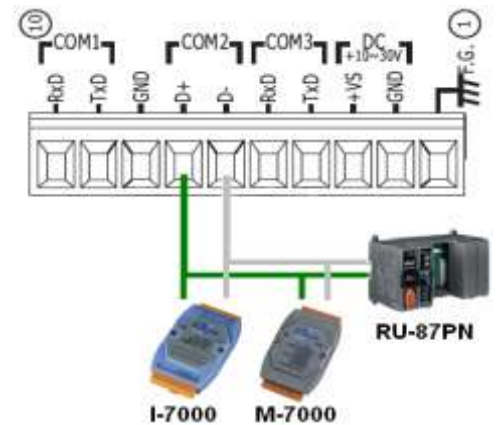


Fig. 8-2

- ❑ Test using command line: (PC ↔ i-7520 ↔ COM2 of LP-51xx)

A) Open “**Hyper Terminal**” of the Host PC to monitor the test process. The default settings for the COM2 port are 9600, 8, N, 1

B) Send data via COM2 port:

On the LP-51xx:

Type command: **echo send-485>/dev/ttyS0**

Check that the word “send-485” is displayed on the “Hyper Terminal” screen on the PC.

C) Receive data via COM2 port:

On the LP-51xx:

Type the command: **cat /dev/ttyS0**

On the PC:

Enter some words in the “Hyper Terminal” screen on the PC. Check that the same text displayed on the LP-51xx.

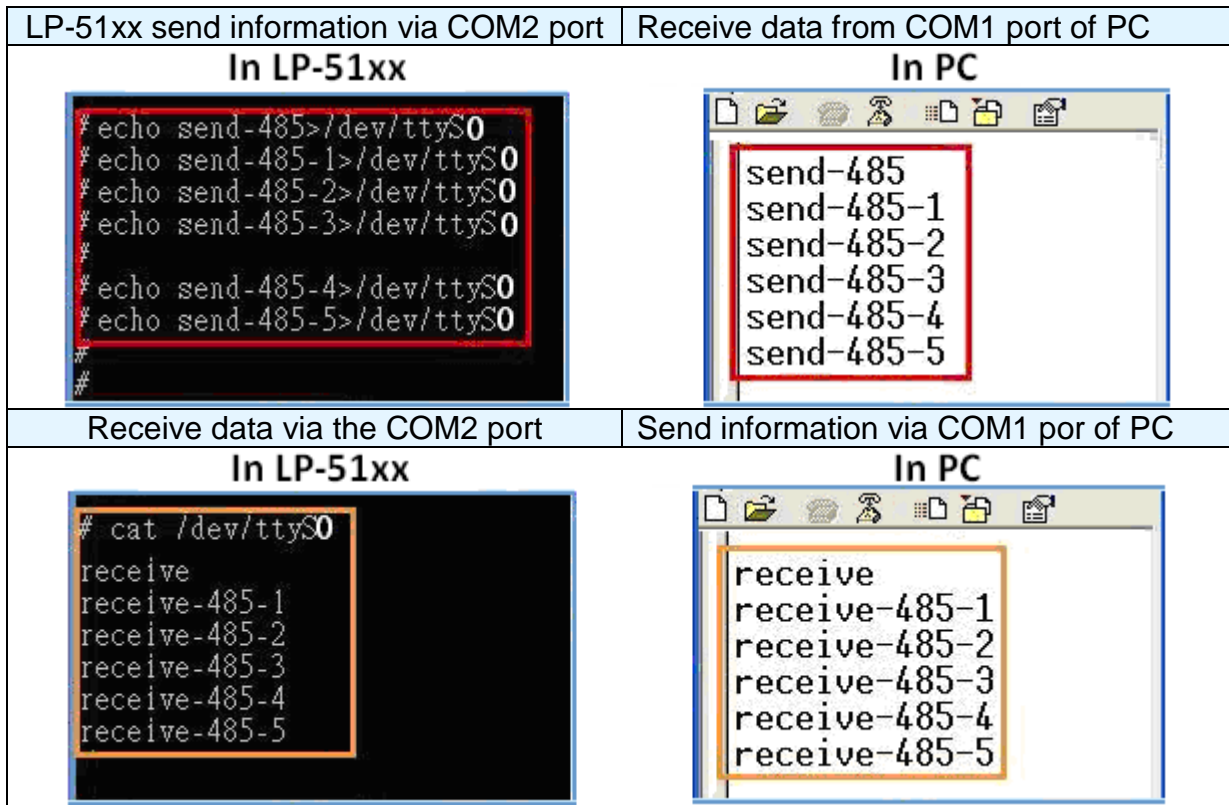


Fig. 8-3

8.3 COM3 Port

This COM3 port is located on the right-upper corner on the LP-51xx and is a standard **RS-232** serial port that provides TxD, RxD, GND, non-isolated.

This port can also be used to connect to an I-7520 module in order to provide general RS-485 communication functionality. The COM3 port can also be used to connect to a wireless modem so that the module can be controlled from a remote device. The application example and the code are demonstrated below:

- Test using C language:

```
unsigned char port=3; data=8, parity=0, stopbit=1;
DWORD baudrate=9600;
Open_Com(port, baudrate, data, parity, stopbit);
// send command...
Close_Com(port);
```

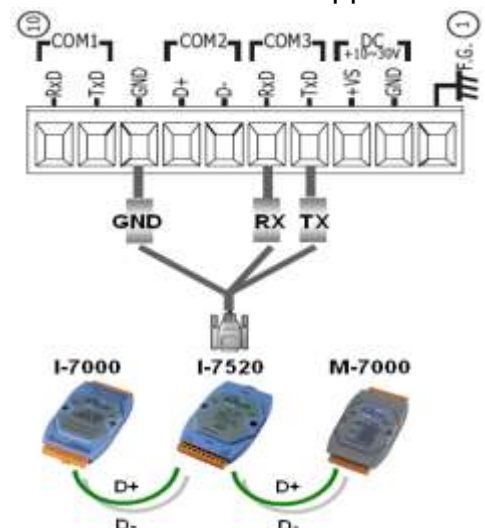


Fig. 8-4

- ❑ Test using the command line interface:
 - (PC connected to COM3 on the LP-51xx)
 - A) Open “**Hyper Terminal**” on the Host PC to monitor the test process. The default settings for COM3 port are 9600, 8, N, 1
 - B) Send data via COM3 port:
 - On the LP-51xx:
 - Type the command: **echo send-232>/dev/ttyS1**
 - Check that the word “send-232” is displayed on the “Hyper Terminal” screen on the PC.
 - C) Receive data via the COM3 port:
 - On the LP-51xx:
 - Type the command: **cat /dev/ttyS1**
 - On the PC:
 - Enter some text in the “Hyper Terminal” screen on the PC.
 - Check that some words on the LP-51xx.

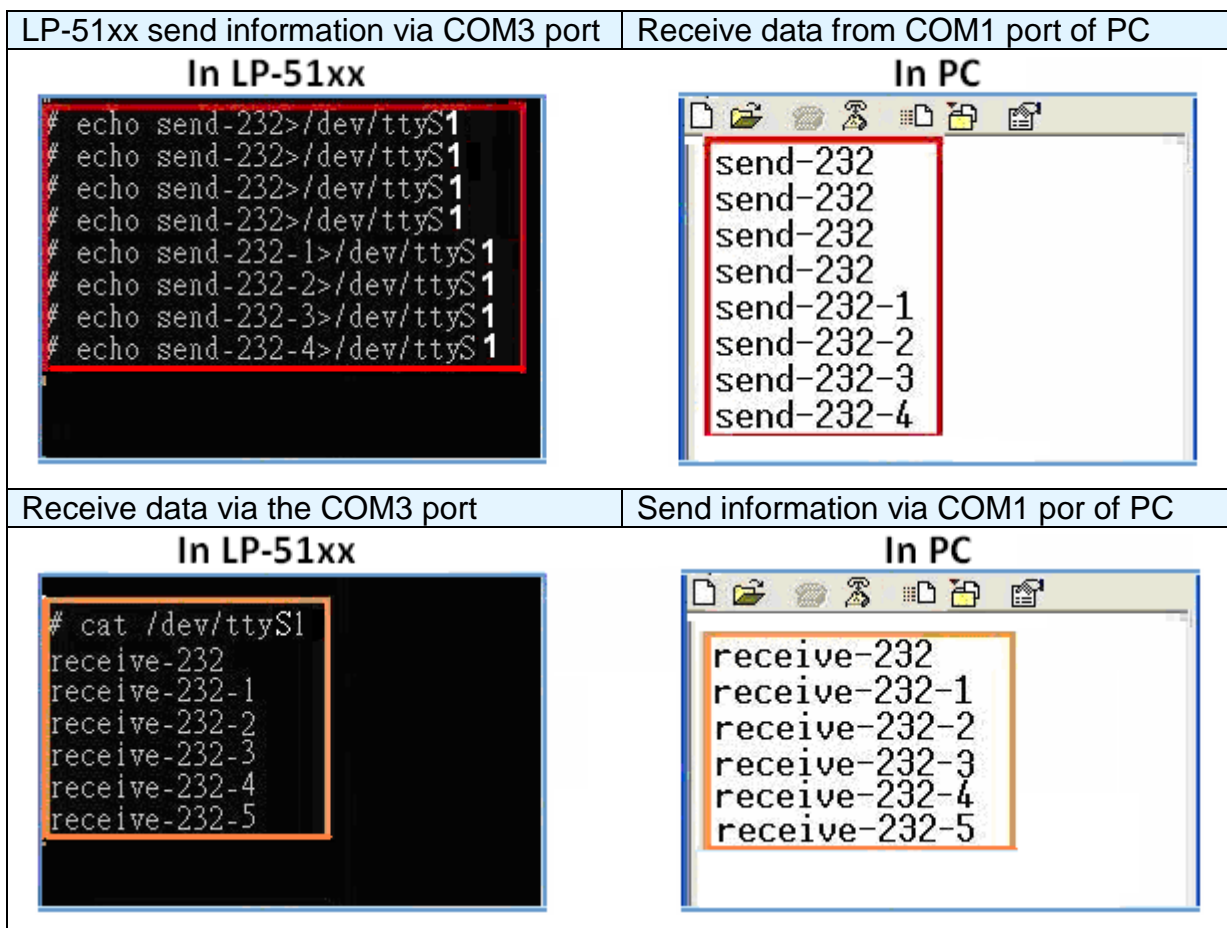


Fig. 8-5

9. LP-51xx Library Reference in C Language

In this chapter, all the functions of **libi8k.a** will be listed to allow users to be able to look them up quickly.

9.1 List Of System Information Functions

```
int Open_Slot(int slot)
void Close_Slot(int slot)
int Open_Slot(void)
void Close_SlotAll(void)
void ChangeToSlot(char slot)
WORD Open_Com(char port, DWORD baudrate, char cData, char cParity, char cStop)
BOOL Close_Com(char port)
WORD Send_Receive_Cmd (char port, char szCmd[ ], char szResult[ ], WORD wTimeOut,
                       WORD wChksum, WORD *wT)
WORD Send_Cmd (char port, char szCmd[ ], WORD wTimeOut, WORD wChksum)
WORD Receive_Cmd (char port, char szResult[ ], WORD wTimeOut, WORD wChksum)
WORD Send_Binary(char port, char szCmd[ ], int iLen)
WORD Receive_Binary(char cPort, char szResult[], WORD wTimeOut, WORD wLen,
                   WORD *wT)

int sio_open(int slot)
int sio_close(int slot)
int sio_set_noncan(int port)
int GetModuleType(char slot)
void Read_SN(unsigned char serial_num[] )
int GetNameOfModule(char slot)
void setLED(unsigned int addr, unsigned int value)
int GetBackPlaneID()
int GetRotaryID()
float GetSDKversion(void)
```

9.2 List Of Digital Input/Output Functions

■ For I-7000 modules via serial port

`WORD` DigitalOut(`WORD` wBuf[], `float` fBuf[], `char` szSend[], `char` szReceive[])
`WORD` DigitalBitOut(`WORD` wBuf[], `float` fBuf[], `char` szSend[], `char` szReceive[])
`WORD` DigitalOutReadBack(`WORD` wBuf[], `float` fBuf[],`char` szSend[], `char` szReceive[])
`WORD` DigitalOut_7016(`WORD` wBuf[], `float` fBuf[], `char` szSend[],`char` szReceive[])
`WORD` DigitalIn(`WORD` wBuf[], `float` fBuf[], `char` szSend[], `char` szReceive[])
`WORD` DigitalInLatch(`WORD` wBuf[], `float` fBuf[], `char` szSend[], `char` szReceive[])
`WORD` ClearDigitalInLatch(`WORD` wBuf[], `float` fBuf[],`char` szSend[],`char` szReceive[])
`WORD` DigitalInCounterRead(`WORD` wBuf[], `float` fBuf[], `char` szSend[],`char` szReceive[])
`WORD` ClearDigitalInCounter(`WORD` wBuf[], `float` fBuf[],`char` szSend[],`char` szReceive[])
`WORD` ReadEventCounter(`WORD` wBuf[], `float` fBuf[],`char` szSend[],`char` szReceive[])
`WORD` ClearEventCounter(`WORD` wBuf[], `float` fBuf[], `char` szSend[],`char` szReceive[])

9.3 List Of Watch Dog Timer Functions

`void` EnableWDT(`unsigned int` msecond)
`void` DisableWDT(`void`)
`unsigned int` WatchDogSWEven(`void`)
`void` ClearWDTSWEven(`unsigned int` rcsr)

9.4 List Of EEPROM Read/Write Functions

`void` Enable_EEP(`void`)
`void` Disable_EEP(`void`)
`unsigned char` Read_EEP(`int` block, `int` offset)
`void` Write_EEP(`int` block, `int` offset, `unsigned char` data)

9.5 List Of Analog Input Functions

■ For I-7000 modules via serial port

WORD AnalogIn(wBuf, fBuf, szSend, szReceive)
WORD AnalogInHex(wBuf, fBuf, szSend, szReceive)
WORD AnalogInFsr (wBuf, fBuf, szSend, szReceive)
WORD AnalogInAll (wBuf, fBuf, szSend, szReceive)
WORD ThermocoupleOpen_7011(wBuf, fBuf, szSend, szReceive)
WORD SetLedDisplay (wBuf, fBuf, szSend, szReceive)
WORD GetLedDisplay (wBuf, fBuf, szSend, szReceive)

9.6 List Of Analog Output Functions

■ For I-7000 modules via serial port

WORD AnalogOut(wBuf, fBuf, szSend, szReceive);
WORD AnalogOutReadBack(wBuf, fBuf, szSend, szReceive)
WORD AnalogOutHex(wBuf, fBuf, szSend, szReceive)
WORD AnalogOutFsr(wBuf, fBuf, szSend, szReceive)
WORD AnalogOutReadBackHex(wBuf, fBuf, szSend, szReceive)
WORD AnalogOutReadBackFsr(wBuf, fBuf, szSend, szReceive)

10. Additional Support

This chapter provides additional information related to the modules supported, together with instructions that can be used to enhance the functionality and efficiency of the LP-51xx module.

10.1 Support for GUI Functionality

Now “**X-window**“ is supported the **VGA** solution. After the LP-51xx boots, a GUI environment similar to a standard ‘**Windows screen**’ desktop will be displayed, as illustrated in Fig.10-1. This means that custom GUI applications can be created and then executed on the LP-51xx. The GUI Library in the LP-51xx is provided by the **GTK+ Library (v1.2 and v2.0)**, which is a multi-platform toolkit for creating the graphical user interface, allowing custom **SCADA** screens to be designed using the GTK+ Library on the LP-51xx.

ICP DAS provides a number of demo programs that illustrate how to use the GUI to control I/O modules and assist in quickly developing custom GUI programs.

Once the LP-51xx SDK has been installed, these demo programs can be found in the **C:\cygwin\LinCon8k\examples\gui** folder.

In addition to the GTK+ GUI functionality, the “**Java GUI**” is also supported on the LP-51xx. This means that developers familiar with the Java environment can also develop custom GUI applications. However, only the Awe and Swing v1.1 elements are supported on the LP-51xx.

To execute the Stylepad.jar Java GUI program on the LP-51xx, open a Xterm window by clicking the ‘Start’ button and choose ‘Xterm’. At the Xterm window, enter “**java -jar Stylepad.jar -cp .: Stylepad.jar**”. Note that it may take some time to execute the Java GUI program.

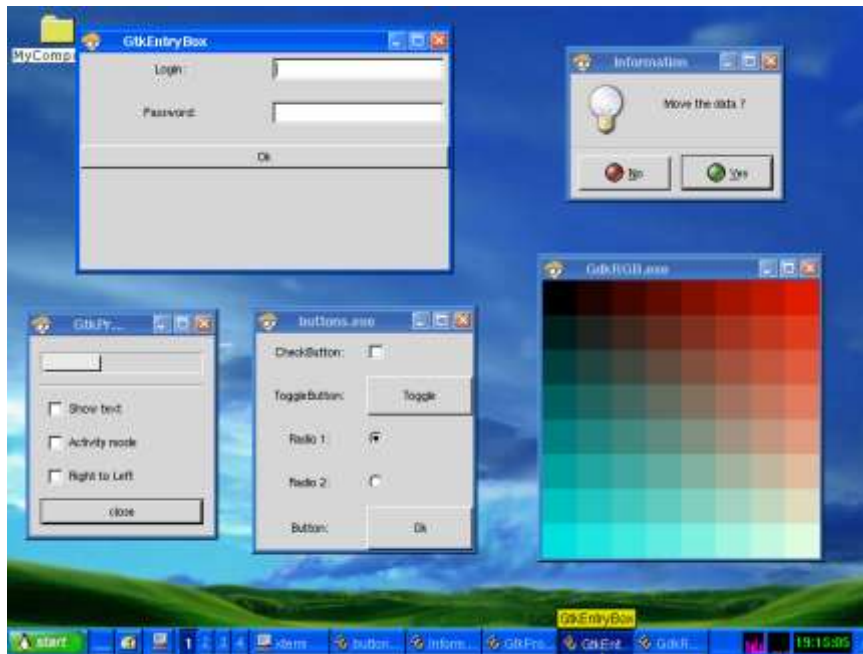


Fig. 10-1

[10.1.1 Booting the LP-51xx without loading the X-window environment](#)

The LP-51xx can be configured to boot without loading the X-window environment by following the procedure described below:

- (1) Enter the command "**cd /etc/rc2.d**" to switch to the default run level.
- (2) Enter the command "**ls -al**" to list the S98Xserver link information into ../init.d/startx.
- (3) To disable X-windows environment, enter the command "**mv S98Xserver Xs98Xserver**" to rename the S98Xserver then exit the Xterm window and reboot the LP-51xx to apply the new configuration.

[10.1.2 Enabling the X-window environment to load at boot time](#)

To enable the X-windows environment, enter the command "**ls -al /etc/rc2.d**" at the Command Prompt to view the S98Xserver link information ../init.d/startx, and then enter the command "**mv Xs98Xserver S98Xserver**" to rename and enable the Xs98Xserver. If the S98Xserver link is not listed, follow the procedure described below:

- (1) Enter the command "**cd /etc/rc2.d**" to switch to the default run level.
- (2) Enter the command "**ln -s ../init.d/startx /etc/rc2.d/S98Xserver**" to create a symbolic link to the X-window script file, which will enable the X-window environment and then exit the Command Prompt and reboot the LP-51xx to apply the new configuration.

10.2 Support for ScreenShot functionality

The “**fbshot**” screenshot application is an embedded program that enables an image of the screen for the LP-51xx to be conveniently captured, as illustrated in Fig. 10-2 below.

Open a Xterm window by clicking the Start button and then clicking Xterm. At the Command Prompt, enter the command “**fbshot -d /dev/fb0 /mnt/hda/catch1.png**” and an image of the current screen will be captured and saved to a file — catch1.png, and will be stored in the /mnt/hda/.directory, as illustrated in Fig. 10-2 below.

To view the image, enter the command “**vi /mnt/hda/catch1.png**”. Note that vi application is included in the path: /mnt/hda/opt/bin directory, so a microSD card must be inserted into the LinPAC before attempting to save the file. To view details of the parameters that can be used in conjunction with the **fbshot** application, enter the command “**fbshot -help**” at the Command Prompt.



Fig. 10-2

10.3 Support for WebCAM functionality

The LP-51xx embedded Controller provides support for WebCAM functionality. Logitech brand cameras have been tested and have been found to work successfully. If a different brand of camera is to use, testing should be performed first to ensure compatibility.

Follow the procedure described below to ensure that the Webcam is configured correctly:

- (1) Connect the webcam to the LP-51xx using the “**USB Interface**”.
- (2) Reboot the LP-51xx.
- (3) Open a “**Command Prompt**” by clicking the Start button and then clicking Xterm. At the Command Prompt, enter the command “**insmod pwc.ko**” to load the gqcam program

decompressor, as illustrated in Fig. 10-3 and then enter the command “**gqcam**” to view the webcam screen. To view details of the parameters that can be used in conjunction with the gqcam application, enter the command “**gqcam -help**” at the Command Prompt.

```
# lsmod
Module                Size      Used by    Tainted: P
pwc                   84996    0
8250                  29140    0
8250_5270             1920     0 [permanent]
slot                 35052    0
pxamci                8352     0
dm9000x              276180   0
#
```

Fig. 10-3

The gqcam program can also be used to capture an image via a webcam. To capture an image, follow the procedure described below:

- (1) Click “**File/ Save Image...**”.
- (2) On the “**Gqcam: Save Image**” screen, enter the path to the folder where the image is to be stored in the “**File Field**”, together with the file name, and then click the “**OK**” button.

10.4 Support for Touch Screen Devices

The LP-51xx embedded Controller provides support for both USB and Serial Touch Screen devices, each of which is discussed in more detail below:

10.4.1 USB Touch Screen interface

Before a USB touch screen can be used, it must first be calibrated. There are seven steps involved in adjusting the calibration for a touch screen connected to an LP-51xx via the USB interface, as follows:

Step 1: To execute the script at startup and shutdown.

- ❑ By default, scripts of USB touch screen are disabled at startup, if the else user can use ‘**mv**’ command to rename files in **/etc/rc2.d**. After reboot, it will be executed automatically at boot time (Refer to the Fig. 10-4).

```

# cd /etc/rc2.d
# ls
S04sd                S70slot                S99rmnologin
S11lifupdown         S71Serial              old
S20ssh               S72Ramdriver           xS81gqcam
S40inetd             S80hwclock             xS88penmount_serial.sh
S50apache            S97fbman               xS89tsdev_serial
S60snmp              S98Xserver             xS90tsdev_usb
# mv xS90tsdev_usb   S90tsdev_usb
#
# ls
S04sd                S70slot                S98Xserver
S11lifupdown         S71Serial              S99rmnologin
S20ssh               S72Ramdriver           old
S40inetd             S80hwclock             xS81gqcam
S50apache            S90tsdev_usb           xS88penmount_serial.sh
S60snmp              S97fbman               xS89tsdev_serial
#

```

Fig. 10-4

Step 2: Open a “**Xterm windows**” by clicking the Start button and then clicking Xterm. At the Command Prompt, ensure that the `usbtouchscreen.ko` and `tsdev.ko` files have been mounted, enter the command ‘`lsmod`’ as illustrated in Fig. 10-5.

```

# lsmod
Module                Size  Used by  Tainted: P
tsdev                  10024  0
usbtouchscreen         9284  0
8250                   29204  0
8250_linpac            2656  0 [permanent]
slot                   35788  0
pxamci                 8352  0
dm9000x                276180  0
#

```

Fig. 10-5

Step 3: At the Command Prompt, ensure that a microSD card has been mounted, enter the command ‘`mount`’ as illustrated in Fig. 10-6.

```

# mount
rootfs on / type rootfs (rw)
/dev/root on / type jffs2 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
tmpfs on /var type tmpfs (rw)
shmfs on /dev/shm type tmpfs (rw)
usbfs on /proc/bus/usb type usbfs (rw)
/dev/mmcbk0p1 on /mnt/hda type vfat (rw, fmask=0022, dmask=0022, codepage=cp437,
iocharset=iso8859-1)
/dev/ram0 on /mnt/ramfs type minix (rw)
#

```

Fig. 10-6

Step 4: At the Command Prompt, edit the `/etc/init.d/fbman` file by modifying the settings so that they are the same as below:

- ❑ After opening the file: `/etc/init.d/fbman`, users can see the following lines :

`/usr/sbin/fbset -n 640x480-60`

`#!/usr/sbin/fbset -n 800x600-70`

These lines indicate that the resolution is currently set to 640*480. The # character indicates that a setting is not currently being used.

- ❑ To change the resolution settings to 800*600, remove the “#” character in line 2 and add the “#” character in line 1 as indicated below:

`#!/usr/sbin/fbset -n 640x480-60`

`/usr/sbin/fbset -n 800x600-70`

Step 5: At the Command Prompt, enter the command ‘`cat /proc/bus/input/devices`’ to view a list of devices that are currently connected and the associated device can be obtained as illustrated in Fig. 10-7.

```
# cat /proc/bus/input/devices
I: Bus=0003 Vendor=04d9 Product=1702 Version=0101
N: Name=" USB Keyboard"
P: Phys=usb-pxa27x-1.1/input0
S: Sysfs=/class/input/input4
H: Handlers=kbd event0
B: EV=120003
B: KEY=10000 7 ff800000 7ff febeffdf f3cfffff ffffffff fffffffe
B: LED=7

I: Bus=0003 Vendor=04d9 Product=1702 Version=0101
N: Name=" USB Keyboard"
P: Phys=usb-pxa27x-1.1/input1
S: Sysfs=/class/input/input5
H: Handlers=kbd event1
B: EV=3
B: KEY=39fa d801d101 1e0000 0 0 0

I: Bus=0003 Vendor=14e1 Product=6000 Version=a4b4
N: Name="DIALOGUE INC PenMount USB"
P: Phys=usb-pxa27x-1.2/input0
S: Sysfs=/class/input/input6
H: Handlers=event2
B: EV=b
B: KEY=70000 0 0 0 0 0 0 0 0
B: ABS=3

I: Bus=0003 Vendor=15d9 Product=0a33 Version=0100
N: Name="USB Mouse"
P: Phys=usb-pxa27x-1.3/input0
S: Sysfs=/class/input/input7
H: Handlers=mouse0 event3 ts0
B: EV=7
B: KEY=70000 0 0 0 0 0 0 0 0
B: REL=103
#
```

Fig. 10-7

Step 6: We are providing the calibration program to test and get the calibration data. For example, open a 'Xterm windows' and execute the command '**calibrator /dev/input/event2**', and then the calibration windows displayed as illustrated in Fig. 10-8)

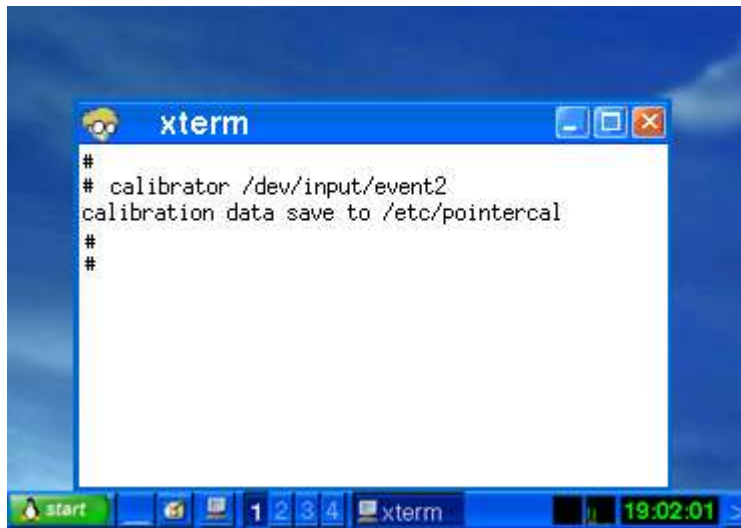


Fig. 10-8

Step 7: Rebooting the LP-51xx to apply the new configuration.

10.4.2 Serial Touch Screen interface

There are three kinds of Touch Screen LCD monitor, so the relevant driver needs to be installed before it can be used. An overview of the respective device drivers and the installation location is provided below:

Module name	Install loadable kernel module
ADP-1080T	/lib/modules/2.6.19/ pm9000.ko
TPM-4100 / TP-4100	/lib/modules/2.6.19/ pm6000.ko

Before a Serial touch screen device and be used, it must first be calibrated. There are nine steps involved in adjusting the calibration for a touch screen calibrated to an LP-51xx via the serial interface, as follows:

Step 1: Open a "**Xterm windows**" by clicking the Start button and then clicking Xterm. At the Command Prompt, enter the command '**cat /etc/init.d/penmount_serial**' to check that the penmount serial driver has been mounted from **/etc/init.d/penmount_serial**, as illustrated in Fig. 10-9.

<pre>usage() { echo "Usage: \$0 {start stop restart}" } EXITCODE=1 for x in "1" ; do if [\$# -lt 1] ; then usage ; break ; fi action=\$1 case "\$action" in start) echo "Starting Penmount /sbin/insmod pm9000.ko # /sbin/insmod pm6000.ko EXITCODE=0 ;; esac</pre>	OR	<pre>usage() { echo "Usage: \$0 {start stop restart}" } EXITCODE=1 for x in "1" ; do if [\$# -lt 1] ; then usage ; break ; fi action=\$1 case "\$action" in start) echo "Starting Penmount # /sbin/insmod pm9000.ko /sbin/insmod pm6000.ko EXITCODE=0 ;; esac</pre>
---	----	---

Fig. 10-9

Step 2: Edit the `/etc/init.d/tsdev_serial` script to modify the device mode. By default, the serial interface is COM3 port, and device mode is `ttyS1`, as illustrated in Fig. 10-10.

```
# /etc/init.d/tsdev_serial 0.1 2012/09/10 ( moki matsushima )
usage()
{
    echo "Usage: $0 {start|stop|restart}"
}
EXITCODE=1
for x in "1" ; do
    if [ $# -lt 1 ] ; then usage ; break ; fi
    action=$1
    case "$action" in
        start)
            echo "Starting Touch Device services: "
            /opt/bin/inputattach --penmount /dev/ttyS1 --daemon
            EXITCODE=0
            ;;
        stop)
            echo -n "Shutting down Touch Device services: "
            /usr/bin/killall inputattach
            echo "done."
            EXITCODE=0
            ;;
        restart)
            $0 stop
            $0 start
            EXITCODE=$?
            ;;
    *)
        usage
        ;;
    esac
done
```

Fig. 10-10

Step 3: Configure the script to be executed at startup and shutdown.

- ❑ By default, the scripts for serial touch screens are disabled at startup. The `'mv'` command can be used to rename the files in `/etc/rc2.d`, which is the file containing instructions used to start processes. The file will be automatically executed when LP-51xx is rebooted, as illustrated in Fig. 10-11.

```

# cd /etc/rc2.d
# ls
S04sd                S70slot                S99rmnologin
S11lifupdown         S71Serial              old
S20ssh               S72Ramdriver           xS81gqcam
S40inetd             S80hwclock             xS88penmount_serial.sh
S50apache            S97fbman               xS89tsdev_serial
S60snmp              S98Xserver             xS90tsdev_usb
# mv xS88penmount_serial.sh S88penmount_serial.sh
# mv xS89tsdev_serial S89tsdev_serial
#
# ls
S04sd                S70slot                S97fbman
S11lifupdown         S71Serial              S98Xserver
S20ssh               S72Ramdriver           S99rmnologin
S40inetd             S80hwclock             old
S50apache            S88penmount_serial.sh xS81gqcam
S60snmp              S89tsdev_serial        xS90tsdev_usb
#

```

Fig. 10-11

Step 4: At the Command Prompt, enter the command '**lsmod**' to check that the **pm9000.ko** or **pm6000.ko** have been mounted, as illustrated in Fig. 10-12.

<pre> # lsmod Module Size Used by Tainted: PF pm9000 2912 0 8250 29204 2 8250_linpac 2656 0 [permanent] slot 35788 0 pxamci 8352 0 dm9000x 276180 0 # </pre>	<pre> # lsmod Module Size Used by Tainted: PF pm6000 2912 0 8250 29204 2 8250_linpac 2656 0 [permanent] slot 35788 0 pxamci 8352 0 dm9000x 276180 0 # </pre>
---	---

Fig. 10-12

Step 5: At the Command Prompt, enter the command '**mount | grep mmc**' and '**ls /mnt/had**' to check the microSD card has been mounted, as illustrated in Fig. 10-13 and 10-14.

```

# mount | grep mmc
/dev/mmcblk0p1 on /mnt/hda type vfat (rw,fnmask=0022,dmask=0022,
codepage=cp437,iocharset=iso8859-1)
#

```

Fig. 10-13

```

# ls /mnt/hda
boot  opt
#

```

Fig. 10-14

Step 6: At the Command Prompt, enter the command '**vi /etc/init.d/fbman**' to edit the **/etc/init.d/fbman** file by modifying the setting so that that are the sam as below:

- ❑ After opening the file locate the following lines:

#/usr/sbin/fbset -n 640x480-60

/usr/sbin/fbset -n 800x600-70

These lines indicate that the resolution is currently set to 640*480. The # character indicates that a setting is not currently being used.

- ❑ To change the resolution setting to be **640*480**, remove the “#” character in line 1 and add it to line 2, as indicated below:

/usr/sbin/fbset -n 640x480-60

#!/usr/sbin/fbset -n 800x600-70

Step 7: At the Command Prompt, enter the command ‘**cat /proc/bus/input/devices**’ to view a list of devices that are currently connected and the associated device can be obtained, as illustrated in Fig. 10-15.

```
# cat /proc/bus/input/devices
I: Bus=0003 Vendor=04d9 Product=1702 Version=0101
N: Name=" USB Keyboard"
P: Phys=usb-pxa27x-1.3/input0
S: Sysfs=/class/input/input0
H: Handlers=kbd event0
B: EV=120003
B: KEY=10000 7 ff800000 7ff febeffdf f3cfffff ffffffff fffffffe
B: LED=7

I: Bus=0003 Vendor=04d9 Product=1702 Version=0101
N: Name=" USB Keyboard"
P: Phys=usb-pxa27x-1.3/input1
S: Sysfs=/class/input/input1
H: Handlers=kbd event1
B: EV=3
B: KEY=39fa d801d101 1e0000 0 0 0

I: Bus=0003 Vendor=15ca Product=00c3 Version=0512
N: Name="USB Optical Mouse"
P: Phys=usb-pxa27x-1.4/input0
S: Sysfs=/class/input/input2
H: Handlers=mouse0 event2
B: EV=7
B: KEY=70000 0 0 0 0 0 0 0 0
B: REL=103

I: Bus=0013 Vendor=0031 Product=0000 Version=0100
N: Name="Penmount Serial TouchScreen"
P: Phys=ttySA2/serio0/input0
S: Sysfs=/class/input/input3
H: Handlers=mouse1 event3
B: EV=b
B: KEY=400 0 10000 0 0 0 0 0 0 0
B: ABS=3
#
```

Fig. 10-15

Step 8: We are providing the calibration program to test and get the calibration data, as illustrated in Fig. 10-16. For example, open a ‘**Xterm**’ windows and execute the command ‘**calibrator /dev/input/event3**’, and then the calibration windows will be displayed, correct 4 point locations on screen with the panel, as illustrated in Fig. 10-17

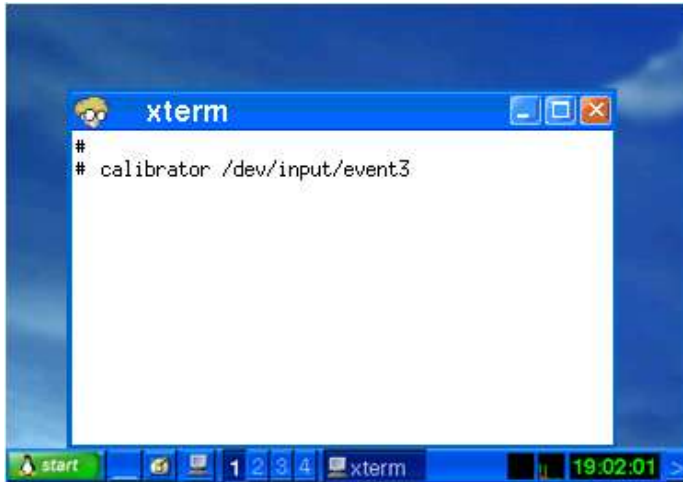


Fig. 10-16

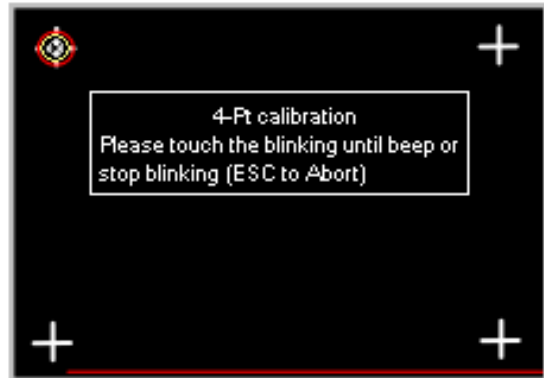


Fig. 10-17

Step 9: Reboot the LP-51xx to apply the new configuration.

10.5 Network Support

The LP-51xx embedded controller already includes a variety of network functions. The following is an overview of the network functions supported in the LP-51xx:

(1) UpnP

UpnP refers to “**Universal Plug and Play**” which is a set of networking protocols that allows other devices to escription y discover and control of services available on a network, without the need for user intervention. Devices that act as servers can advertise their services to clients. Client systems, known as control points, are able to search for devices that provide specific services on the network. When a device that provides the desired service is discovered, the Control Points is able to retrieve a detailed descriptions of the devices and services allowing the server and client devices to interact from that point on.

(2) VPN

VPN refers to “**Virtual Private Network**” and is used to securely extend a private network across a public network, as illustrated in Fig. 10-18. VPN describes a network that includes secure remote access for client devices, enabling data to be securely sent and received across shared or public networks as if the device was directly connected to the private network.

The term “**Virtual**” refers to the fact that the devices on the network do not need to be physically connected.

The term “**Private**” implies that the data is encrypted and can only be viewed by a defined group connected to the VPN, meaning that it is extremely difficult for the unauthorized user to access confidential information. The last word, “**Network**”, means that the users configured for VPN can be connected and share files or information. So it’s extremely difficult for anyone to snoop on confidential information through VPN.

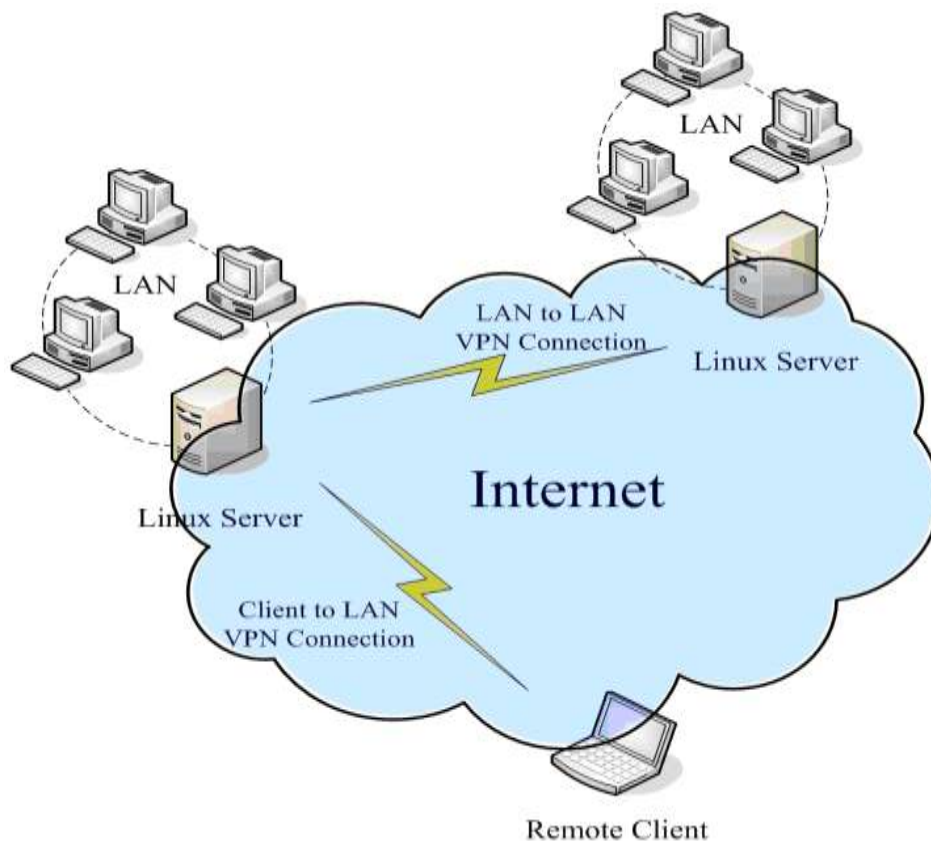


Fig. 10-18

(3) QoS

QoS refers to “**Quality of Service**” and describes the overall performance of a network, particularly that send by users of the network. QoS is based on a set of techniques that are used to manage network resources. For example, if there are a number of packets that need to be sent to a network device, the device has to first determine a transmission priority, i.e., which packets to send first, which ones to delay, and, if necessary, which ones to drop. The transmission is then performed based on that priority, thereby achieving the required QoS by managing the delay, jitter, bandwidth, and packet loss parameters on the network. Using the Linux QoS subsystem, it is possible to create a highly flexible traffic control system that ensures the flow rate to an assigned port can be effectively controlled, improving network quality in the process.

(4) Wireless LAN (WLAN)

A “**Wireless Local Area Network (WLAN)**” is a network technology that allows the connection of two or more devices without requiring the installation of any wires or cables, mostly using **radio** technology, and sometimes **infrared**. The range of the WLAN is targeted on a limited area, generally within an office building, a commercial area, a small campus, or a home, etc. As technology has become more prevalent, Linux has adopted many of the technologies and tools that take advantage of wireless networking.

If a wireless card is inserted into a slot on the LP-51xx, the parameters contained in the `/etc/network/interfaces` file need to be modified.

(5) Dual LAN

The Dual LAN functionality provided on the LP-51xx allows a wireless network and a cable network to be combined through the LP-51xx, meaning that it is possible to establish communication between the cabled LAN and the wireless LAN. This ensures that as long as either of the two LANs can connect to the Internet, then all devices connected to the network will be able to access the Internet as illustrated in Fig. 10-19.

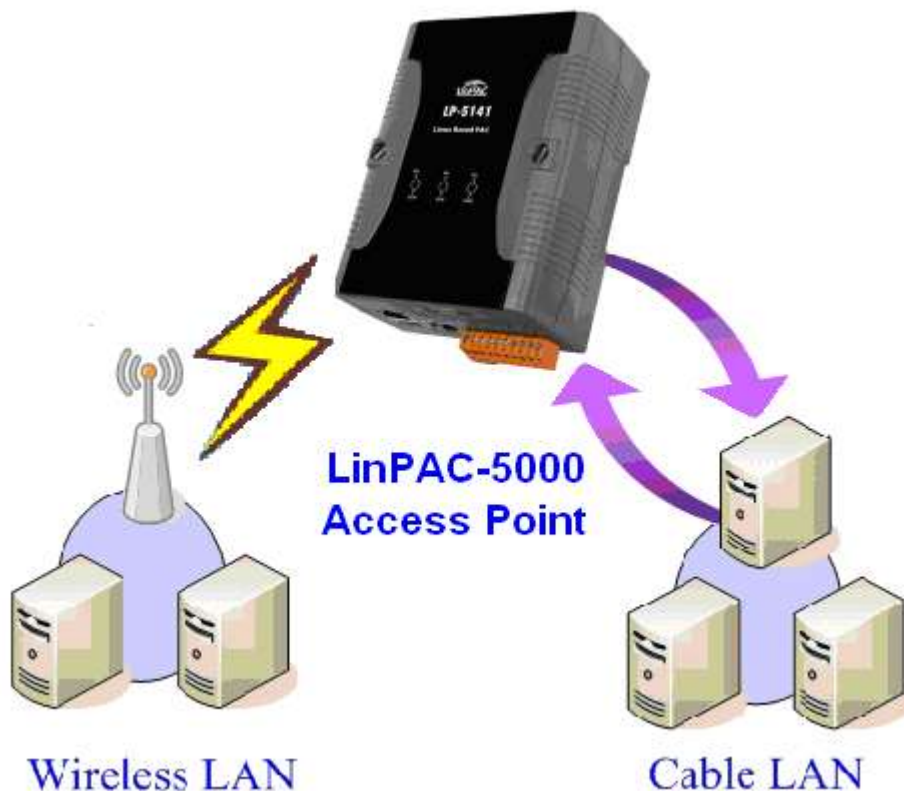


Fig. 10-19

(6) BlueTooth

Bluetooth is a worldwide specification for a small-form factor, the low-cost wireless technology that is used for exchanging data over short distances between both fixed and mobile computers, mobile phones, and other portable handheld devices, as well as providing connectivity to the Internet. “**BlueZ**” is a Bluetooth stack designed for Linux operating system that is now embedded in the LP-51xx, providing support for the core Bluetooth layers and protocols. Bluetooth technology is flexible and efficient and based on a modular implementation.

(7) Modem / GPRS / ADSL Connectivity

GPRS modem selection guide:

Module	Install loadable kernel module	Execute command
GTM-201-USB	/lib/modules/2.6.19/ usbserial.ko /lib/modules/2.6.19/ sim5218.ko	pppd call 3g &
GTM-201-RS232	/lib/modules/2.6.19/ ftdi_sio.ko	pppd call wavecom &

For more information, please refer to http://m2m.icpdas.com/m2m_layer2_gprs.html

Note: If user want to try GTM-201-USB, please type “insmod [usbserial.ko](#)” first, and “insmod [sim5218.ko](#)” to load the program decompressor.

The following is a description of the procedure for configuring the GPRS modem, a GTM-201-RS232 GPRS Modem for example.

❑ Part 1

In order to connect a GPRS modem to the COM3 port on an LP-51xx, the [/etc/ppp/peers/wavecom](#) file must first be modified to define the COM port. Connect the GTM-201-RS232 (GPRS Modem) using an RS-232 interface by following the instructions below:

- (1) Open a “**Xterm windows**” by clicking the Start button and then clicking Xterm. At the Command Prompt, enter the command “**vi /etc/ppp/peers/wavecom**” to edit the file.
- (2) Locate the statement “Serial device to which the GPRS phone is connected:”, and add the device name for the COM port, as illustrated in Fig. 10-20.
- (3) At the Command Prompt, enter the command “**:wq**” to save the changes and close the script.

Note: In order to provide support for 2G GPRS Modems the — [ftdi_sio.ko](#) loadable kernel module must be installed using the **insmod** command.

```

# Serial device to which the GPRS phone is connected:
# /dev/ttyS0 for serial port (COM1 in Windows),
# /dev/ircomm0 for IrDA,
# /dev/ttyUB0 for Bluetooth (Bluez with rfcomm running) and
# /dev/ttyUSB0 for USB
#/dev/ttyS34 # serial port one
#/dev/ttyS0 # serial port one
/dev/ttyS1 # serial port two → Connect the gprs to the COM3
#/dev/ircomm0 # IrDA serial port one
#/dev/rfcomm0 # Bluetooth serial port one
#/dev/ttyUSB0 # USB serial device, for example Orange SPV

```

Fig. 10-20

□ Part 2

The default baud rate for GPRS chip is “115200” bps. Consequently, both the GPRS module and the device node, such as /dev/ttyS2, should be configured to use the same baudrate. User the ‘stty’ command to set the input and output speed of the device node, as illustrated in Fig. 10-21.

```

# login 1
linpac-8000 login: root
Password:
MOKI 0.90
Jan  3 18:01:25 login[1240]: root login on 'console'
-sh: can't access tty; job control turned off
installed modules list
slot 1 ... 8112
# insmod /lib/modules/2.6.19/ftdi_sio.ko 2
drivers/usb/serial/usb-serial.c: USB Serial support registered for FTDI USB
Serial Device
usbcore: registered new interface driver ftdi_sio
drivers/usb/serial/ftdi_sio.c: v1.4.3:USB FTDI Serial Converters Driver
# stty -F /dev/ttyS2 ispeed 115200 ospeed 115200 3
# stty -F /dev/ttyS2
speed 115200 baud;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W;
lnext = ^V; flush = ^O; min = 1; time = 0;
-brkint -imaxbel

```

Fig. 10-21

Before starting the GPRS modem, the network interfaces for both eth0 and eth1 must first be deactivated. Remove the Ethernet cable, and then enter the command “ifdown eth0” and “ifdown eth1” at the Command Prompt to deactivate the interfaces.

At the Command Prompt, enter the command “pppd call wavecom &”. The LP-51xx will then be automatically connected to the internet. It should be remembered that the network interface for the LinPAC device must be deactivated first. Enter the command “ifconfig” at the Command Prompt to display the “ppp0” section, as illustrated in Fig. 10-22.

```

# ifconfig
eth0  Link encap:Ethernet HWaddr 00:0D:E0:AB:CD:
UP BROADCAST RUNNING MULTICAST MTU:
RX packets:0 errors:0 dropped:0 overruns:0
TX packets:3 errors:0 dropped:0 overruns:0
collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B) TX bytes:1770 (1.7 KiB)
Interrupt:41 Base address:0x8000

eth1  Link encap:Ethernet HWaddr 00:0D:
UP BROADCAST RUNNING MULTICAST
RX packets:0 errors:0 dropped:0 overruns:0
TX packets:3 errors:0 dropped:0 overruns:0
collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B) TX bytes:1770 (1.7 KiB)
Interrupt:114 Base address:0xc000

lo    Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
UP LOOPBACK RUNNING MTU:16384 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 ca
collisions:0 txqueuelen:0
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

ppp0  Link encap:Point-to-Point Protocol
inet addr:111.81.57.21 P-t-P:10.64.64.64 Mask:255.255.255.255
UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1500 Metric:1
RX packets:8 errors:0 dropped:0 overruns:0 frame:0
TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:3
RX bytes:186 (186.0 B) TX bytes:129 (129.0 B)

```

ppp0 Link encap:Point-to-Point Protocol
inet addr:111.81.57.21 P-t-P:10.64.64.64 Mask:255.255.255.255
UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1500 Metric:1
RX packets:8 errors:0 dropped:0 overruns:0 frame:0
TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:3
RX bytes:186 (186.0 B) TX bytes:129 (129.0 B)

Fig. 10-22

Fig. 10-23 below provides an example of a routing table.

```

# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
192.200.1.21 * 255.255.255.255 UH 0 0 0 ppp0
default 192.200.1.21 0.0.0.0 UG 0 0 0 ppp0
#
# ftp ftp.speed.hinet.net
Connected to ftp.speed.hinet.net.
220- Welcome to HiNet SpeedTest FTP site.
220- (ftp.speed.hinet.net)
220
Name (ftp.speed.hinet.net:root): ftp
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> by
221 Goodbye.
#

```

Fig. 10-23

[ADSL]

To use an ADSL modem, the ADSL option must be configured first. To do this, enter the command “**adsl-setup**” at the Command Prompt, then enter the command “**adsl-connect**” to establish an Internet connection on the LP-51xx. To disable the ADSL connection, enter the command “**adsl-stop**” at the Command Prompt.

(8) Firewall (ip tables function)

A firewall is used to control unauthorized access to a local network, even against an intentional attack from an external network, locking out intruders and ensuring that both the system and the data is safe.

(9) Web Browser

The LP-51xx contains an embedded Web Browser that can be used to access the Internet. Open a “Dillo windows” by click the “Start” > ”Programs” button and then clicking ‘Dillo’. At the Command Prompt to open the web browser and then enter the address of a web site, as illustrated in Fig. 10-24.

Note that the dillo command is located in the path: `/mnt/hda/opt/bin`, so a microSD card must first be inserted into the SD socket on the LinPAC.



Fig. 10-24

(10) Apache Server

The LP-51xx contains an embedded “**Apache**” **Web Server**, which will be automatically loaded when the module is booted up. The files for the server can be found in the **/opt/apache2** directory. To connect to the web server embedded on the LP-51xx, enter the URL “<http://192.168.0.200>”. If a web page is successfully displayed, it means that the web page on the LP-51xx has been started. The index page for the Apache Server can be found in the “**/opt/apache2/htdocs/**” directory.

The files that provide the full functions of the Apache Server are located on the microSD card. This means if other Apache Server functions need to be used that are not supported on the LP-51xx, these files can be copied from the microSD card to the **/opt/apache2** directory from the microSD card. The new or additional functions will then be available once the LP-51xx is rebooted.

10.6 Audio Function

LP-5131-OD and LP-5141-OD support audio function— MAD(MP3 Audio Recorder, MAD), the MP3 Audio Recorder is a powerful sound recording and playing program. With the user can record sound from microphone and play sound from the speaker. Recorded sound can be saved in Wav-file, MP3, WMA format, etc. There are three major types of audio functions:

□ Volume adjustment

The **smixer** is a command-line and scriptable program to control and display the mixer volume levels on a sound card in LP-51xx. If users want to adjust the MIC/Speaker volume, please follow the steps:

- (1) Type “**vi /etc/smixer.conf**” to adjust the volume of Mic, Igain, Spkr, Rec, etc.
- (2) Type “**smixer -a /etc/smixer.conf**” to set settings from the file.

If users want to know the detailed parameters of madplay, just type in “**smixer help**” or refer to <http://centerclick.org/programs/smixer/man.html>.

□ Sound player

In LP-51xx, the **madplay** is a command-line MPEG audio decoder and player. After users download music files into LP-51xx, please refer to the following ways for play:

Parameter / Function		Description	Example
Normal		Normal volume	madplay test.mp3
-a	amplify signal by DECIBELS (+)	increase 10 decibels	madplay test.mp3 -a +10
	attenuate signal by DECIBELS (-)	decrease 10 decibels	madplay test.mp3 -a -10
Note: Apply to OS version 1.9 and earlier then version only.			

Parameter / Function		Description	Example
Normal		Normal volume	madplay test.mp3 -a -50
-a	amplify signal by DECIBELS	increase 10 decibels	madplay test.mp3 -a -40
	attenuate signal by DECIBELS	decrease 10 decibels	madplay test.mp3 -a -60
Note: Apply to OS version 2.0 and later then version only.			

❑ Sound recorder

Please follow the steps to make the sound recoder function work smoothly:

- (1) Type “**cat /dev/dsp > /dev/dsp**” to listen to the speaker from microphone.
- (2) Type “**cat /dev/dsp > /var/test.wav**” to save file from microphone recorder.
- (3) Type “**cat /var/test.wav > /dev/dsp**” to listen to the test.wav from speaker.

10.7 Support for USB to RS-232 Conversion

The LP-51xx provides support for USB to RS-232 converter modules, such as the I-7560 for example. The I-7560 module contains a Windows serial COM port that is implemented via its USB connection and is compatible with both new and legacy RS-232 devices. USB Plug-and-Play allows easy serial port expansion and requires no IRQ, DMA, or I/O port resources. For more details related to the I-7560 module, refer to http://www.icpdas.com/products/Remote_IO/i-7000/i-7560.htm. Follow the procedure described below to ensure the successful operation of the USB to RS-232 converter:

- (1) Connect the I-7560 to the LP-51xx using a “**USB Interface**”, as illustrated in Fig. 10-25.



Fig. 10-25

- (2) Power on the LP-51xx.
- (3) Open a “**Xterm windows**” by clicking the Start button and then clicking Xterm. At the Command Prompt, enter the command Open a “**Command Prompt**” by clicking “**insmod usbserial.ko**”, and then enter the command “**insmod pl2303.ko**” to load the program decompressor.
- (4) After successfully executing the **insmod** command, a new **/dev/ttyUSB0** serial device will be created. Use the “**echo**” and “**cat**” commands to send and receive messages, as illustrated in Fig. 10-26 and 10-27 below.



```
# echo 7560_com3>/dev/ttyUSB0
# echo 7560_com3>/dev/ttyUSB0
# echo 7560_com3>/dev/ttyUSB0
# echo 7560_com3>/dev/ttyUSB0
#
#
#
```

Fig. 10-26



```
# cat /dev/ttyUSB0
7560_com3
7560_com3
7560_com3
7560_com3
```

Fig. 10-27

10.8 Additional Optional Functions

The LP-51xx provides support for a number of additional functions, a description of which is listed below. To activate any of these functions so that they can be used in combination with the LP-51xx, copy the relevant directory to the “**opt**” directory on the microSD card. After rebooting LP-51xx, the new function will be loaded automatically.

(1) MySQL

MySQL is a small open source “Relational DataBase Management System” RDBMS that provides support for a wide range of platforms, including UNIX, Linux or Windows, allowing data to be easily added or deleted.

- Start the MySQL service

To install MySQL for use in combination with the LP-51xx, check the “**mysql**” directory in the /opt directory of microSD card, and then choose one of the following installation methods:

a) Manual	b) Auto
<pre># mysql_install_db # mysqld_safe --user=root & # mysql</pre>	<pre># cd /etc/rc2.d # ln -s ../init.d/mysql.server S88mysql # cd /etc/rc0.d # ln -s ../init.d/mysql.server K15mysql # cd /etc/rc6.d # ln -s ../init.d/mysql.server K15mysql # shutdown -r now : # mysql</pre>

Fig. 10-28

```
# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 4.1.10

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
mysql>
```

Fig. 10-29

- Compile a mysql demo program

Follow the procedure described below to compile a MySQL demo program using the LinPAC SDK:

- (1) Copy the mysql directory from the **/opt** directory on the microSD card to **C:\cygwin\opt**, as illustrated in Fig. 10-30.
- (2) Coding a demo program in the **C:\cygwin\LinCon8k\examples** directory, as illustrated in Fig. 10-31.

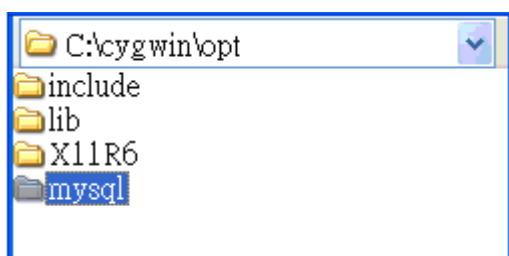


Fig. 10-30

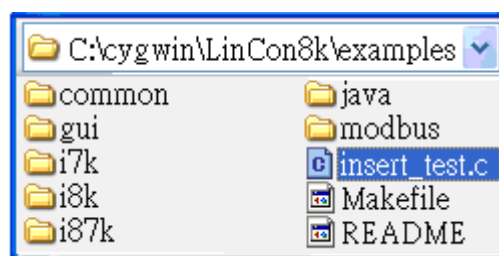


Fig. 10-31

- (3) Double click the **“LP-51xx Build Environment”** to compile the applications.
- (4) Compile the demo program, as illustrated in Fig. 10-32:

At the Command Prompt, enter the following:

```
C:\cygwin\LinCon8k\examples> arm-linux-gcc -I..\opt\mysql\include\mysql\
-L..\opt\mysql\lib\mysql\ insert_test.c -o insert_test.exe -lmysqlclient
```

```

C:\cygwin\LinCon8k>cd examples
C:\cygwin\LinCon8k\examples>arm-linux-gcc -I..\..\opt\mysql\include\mysql\
-L..\..\opt\mysqlf\lib\mysql\ insert_test.c -o insert_test.exe -lmysqlcli
ent
C:\cygwin\LinCon8k\examples>ls ins*
insert_test.c insert_test.exe
C:\cygwin\LinCon8k\examples>

```

Fig. 10-32

(2) PHP

PHP is a server-side open source scripting language that can be used to design dynamic web pages. When PHP is implemented in combination with MySQL, the resulting applications are cross-platform, which means that applications can be developed on a Windows-based system and served on a Linux platform, as illustrated in Fig. 10-33 below.

PHP functionality has been embedded into the kernel on the LP-51xx, meaning that PHP can be used on the LP-51xx directly after booting the device.

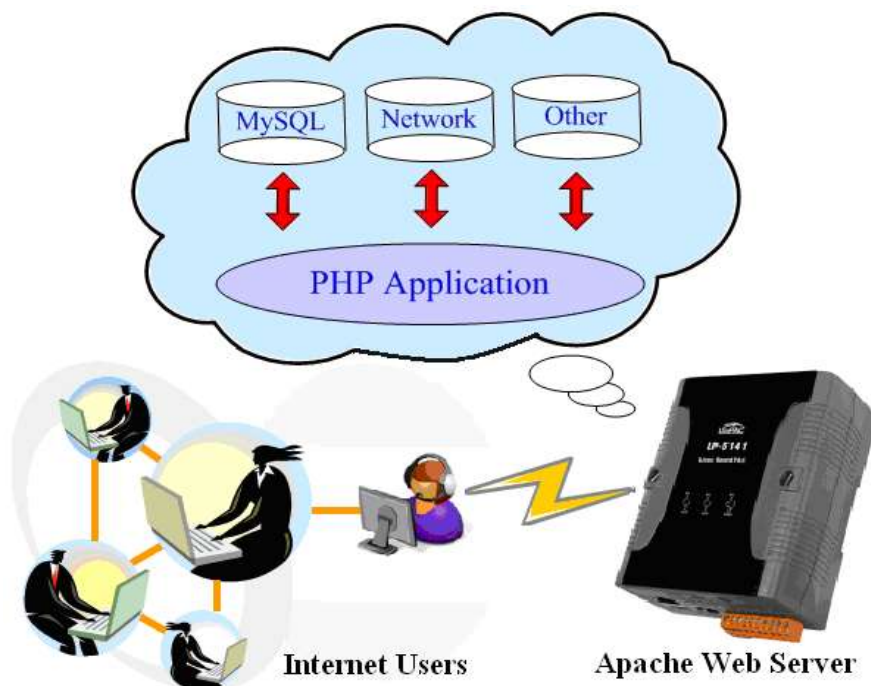


Fig. 10-33

(3) Perl Support

Perl (Practical Extraction and Report Language) is also an “open source scripting language” and has been embedded into the kernel on the LP-51xx, meaning that Perl can be used on the LP-51xx directly after booting the LP-51xx.

Appendix A. Service Information

This appendix will show how to contact ICP DAS when you have problems in the LP-51xx or other products.

Internet Service:

The [internet service](#) provided by ICP DAS will be satisfied and it includes [Technical Support](#), [Driver Update](#), [OS_Image](#), [LP-5000 SDK](#) and [User's Manual Download](#) etc. Users can refer to the following web site to get more information:

1. ICP DAS Web Site: <http://www.icpdas.com>
2. Software Download: <http://www.icpdas.com/download/index.htm>
3. Java Supported Document: <http://www.icpdas.com/download/java/index.htm>
4. E-mail for Technical Support: service@icpdas.com or service.icpdas@gmail.com

Manual Revision:

Manual Edition	Revision Date	Revision Details
v1.0	2010. 11	1. Modify the LP-5000 SDK installation path 2. Add demo description in chapter 7
V1.1	2010. 12	1. Add mysql description 2. Add microSD card instruction 3. Add 4.2.3 scan and repair microSD card 4. Add e-mail account (gmail) 5. Add quick installation guide for Linux 6. Add USB to serial support
V1.2	2011. 12	1. Add SDK guide in Linux PC 2. Add detailed description for GPRS usage
V1.3	2012. 02	1. Update 8250_linpac.ko and modify the COM port definition in chapter 8
V1.4	2012. 06	1. Rename the product name and change the FTP download site. 2. Add USB Touch screen support
V1.5	2012. 09	1. Add Touch screen support for USB and serial interface
V1.6	2017. 06	1. Add notes for flash and microSD disk. 2. Add Code::block IDE application 3. Add description for new Auido version: AIC33x usange
V2.0	2018. 03	1. Upgrade I/O Library and LinPAC PXA270 SDK