Java API Reference Guide

LinPAC / LinCon

Version 1.1

Contents

1. Introduction of LinPAC/LinCon With Java Language

Java is an independent operating system platform for software development. It consists of a programming language, utility programs and a run time environment (JVM). A Java program can be developed on one computer that must be installed with the JDK and run on any other computer with the correct run time environment. This language is freely available to the public to use on anything from personal Web sites to corporate enterprise systems to NASA spacecraft. Java was written from the ground up to be a clean, safe, secure, and object-orientated programming language. Therefore **JDK for Linux** is also installed in the LinPAC/LinCon, so that users can compile and run Java programs in it. In the meanwhile, we provide the **Java I/O Driver** – **JIOD** to allow users to be able to control I/O modules of ICP DAS with Java language.

1.1 JIOD and JAVA

Java I/O Driver (JIOD) is the Java platform technology of choice for extending and enhancing JVM to make many industry control applications possible. JIOD includes I/O packages for I-7K, I-8K and I-87K remote I/O modules and PCI bus series add-on cards. JIOD provides developers with a simple and easy mechanism for extending the functionality of JVM and accessing ICP DAS products.

The **JIOD** contains three packages namely：**com.icpdas.ixpio, com.icpdas.ixpci** and **com.icpdas.comm** which will support variant ICP DAS's various products. The three packages are all included in **icpdas.jar** and their functionalities are listed below：

Packages Summary	
com.icpdas.comm	For I-7k,I-8K,I-87K series remote I/O modules
com.icpdas.ixpci	For PCI series add-on cards
com.icpdas.ixpio	For PIO series add-on cards

These packages in the JIOD are easy to understand. they provide powerful, easy-to-use packages for developing your data acquisition application. The program can use these packages within applications, applets and servlets with ease. To speed-up your developing process, some demonstration source programs are provided. The relationship between the JIOD and the user's application is depicted in Fig 1-1.

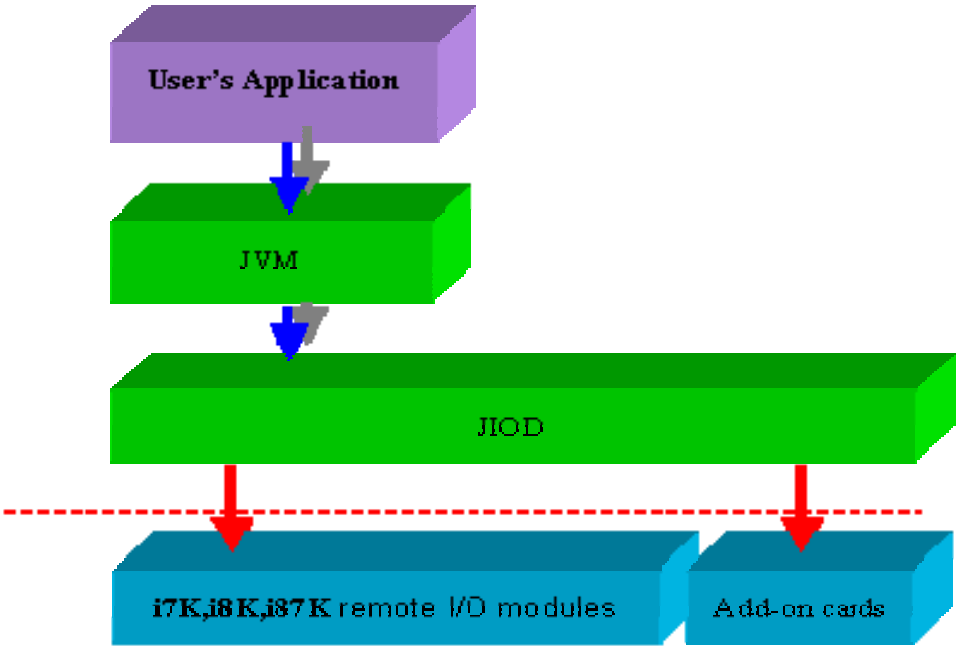

Fig 1-1

1.2 How to Use JIOD

If you want to use Java language to write programs to control ICP DAS products, you must first install the JIOD driver. In order to download the JIOD driver or get more information about it, you are welcome to visit our web site at either - **http://w3.icpdas.com/moki/** or **http://www.icpdas.com/download/index.htm**

Using JIOD is very similar to that for C language users because **icpdas.jar** for Java is just like **libi8k.a** for C. The key points for this are provided below：

1. After you install the JIOD driver, you will see the file - **icpdas.jar**. Then please add the icpdas.jar path to the environment variable - **CLASSPATH**. For more detailed steps relating to this, you can refer to the documents on the web site - **http://w3.icpdas.com/moki/.**

2. Import JIOD into your source program.

1.3 Introduction of com.icpdas.comm Package

Here we just introduce the package - **com.icpdas.comm** of JIOD**.** The com.icpdas.comm package can be used to write platform-independent industry applications. We provides the necessary classes to control ICP DAS remote modules – I-7K, I-87K, I-8K. The command set for these modules will be compatible to ADAM, Nudam, and 6B of Analog Devices in the future.

(ICP DAS remote I/O modules)

1.3.1 Class of com.icpdas.comm

java.lang.Object
|
+--com.icpdas.comm.Comm

The Comm Class includes both the low level serial communication method and the high level remote I/O modules control method. A serial port can be opened for reading and writing data. Once the application is done with the port, it must call the close method before ending the program.

Class Summary	
Comm	Defines methods for communication to serial devices
Slot	Defines methods for communication to parallel devices for I/O of slot.
IoBuf	Remote modules control matrix.

1.3.2 Class Comm

Method Summary	
int	**open**(int portno, int baudrate, int databit, int isparity, int stopbit) Initialize the COM port.
void	**close**(int portno) Free all the resources used by open. This method must be **called before** the program exit.
int	**setSendCmd**(IoBuf ioArg) Create a thread to send a command to module.
int	**getReceiveCmd**(IoBuf ioArg) Create a thread to receive the response-result from module.
int	**getSendReceiveCmd**(IoBuf ioArg) Create a thread to send a command and receive the response-result from module.
int	**getAnalogIn**(IoBuf ioArg) Read the analog input value from module.
int	**getAnalogInAll**(IoBuf ioArg) Read all channels of analog input values from module.
int	**setAnalogOut**(IoBuf ioArg) Send the analog output command to module.
int	**getAnalogOutReadBack**(IoBuf ioArg) Read back the current D/A output value of module.
int	**getDigitalIn**(IoBuf ioArg) Read the digital input value from module.
int	**setDigitalOut**(IoBuf ioArg) To set the digital output value for module.
int	**getDigitalOutReadBack**(IoBuf ioArg) Read back the digital output value of module.
int	**setDigitalBitOut**(IoBuf ioArg) Set the digital value of digital output channel No.
int	**getDigitalInLatch**(IoBuf ioArg) Obtain the latch value of the high or low latch mode of Digital Input module.
int	**setClearDigitalInLatch**(IoBuf ioArg) Clear the latch status of digital input module when latch function has been enabled.

int	**getDigitalInCounter**(IoBuf ioArg) Obtain the counter event value of the channel number of Digital Input module.
int	**setClearDigitalInCounter**(IoBuf ioArg) Clear the counter value of the digital input channel No.
int	**setAlarmMode**(IoBuf ioArg) To set module enter momentary alarm mode or latch alarm mode.
int	**setAlarmConnect**(IoBuf ioArg) Set the link between DO and AI module.
int	**setClearLatchAlarm**(IoBuf ioArg) To clear the latch alarm for module.
int	**setAlarmLimitValue**(IoBuf ioArg) To set a high or low alarm limit value for module.
int	**getAlarmLimitValue**(IoBuf ioArg) To get the high or low alarm limit value for module.
int	**getAlarmStatus**(IoBuf ioArg) Reading the alarm status for a module.
int	**getAlarmMode**(IoBuf ioArg) Reading the alarm mode for a module.
int	**getConfigStatus**(IoBuf ioArg) Obtain the configuration status of the modules.
int	**setStartUpValue**(IoBuf ioArg) Configure the initial analog output of analog output module when its power is on.
int	**getStartUpValue**(IoBuf ioArg) Obtain the initial output setting value of analog output module when the power is on.

Field Summary

DATABITS_5, DATABITS_6, DATABITS_7, DATABITS_8, PARITY_EVEN, PARITY_NONE, PARITY_ODD, STOPBITS_1, STOPBITS_1_5, STOPBITS_2

1.3.3 Class Slot

java.lang.Object
|
+--com.icpdas.slot.Slot

The Slot Class included high level local I/O modules control method. AI/O slot can be opend for reading and writing data. Once the application is done with the port, it must call the close method before end program.

Method Summary

Common	
int	**open**(int slotno) Description: This function is used to open and initiate a specifed slot in the LinPAC/LinCon. The I-8K or I-87K modules in the LinPAC/LinCon will use this function. For example, if you want to send or receive data from a specified slot, this function must be called first. Then the other functions can be used later. Parameter: slotno : [Input] Specify the slot number in which the I/O module is plugged into.
void	**close**(int slotno) Description: If you have used the function of slot.open() to open the specifed slot in the LinPAC/LinCon, you need to use the slot.close() function to close the specifed slot in the LinPAC/LinCon. The I-8k or I-87k modules in the LinPAC/LinCon will use this function. For example, once you have finished sending or receiving data from a specified slot, this function would then need to be called. Parameter: slotno : [Input] Specify the slot number in which the I/O module is plugged into.
void	**outb**(int slotno, int offset,int value) Description: Output the 8-bit data to a module. Parameter: slotno : [Input] Specify the slot number in which the I/O module is plugged into. offset : [Input] Specify the I/O address. value : [Input] output data.

int	**inb**(int slotno, int offset) Description: This function is used to obtain 8-bit raw data from a module. Parameter: slotno : [Input] Specify the slot number in which the I/O module is plugged into. offset : [Input] Specify the I/O address. Return Value: The raw data in I/O address.
int	**setChangeToSlot**(int slotno) Description: This function is used to dedicate serial control to the specified slots for the control of the I-87k series. The serial bus in the LinPAC/LinCon backplane is for mapping through to COM1. For example, if you want to send or receive data from a specified slot, you need to call this function first. Then you can use the other series functions. Parameter: slotno : [Input] Specify the slot number in which the I/O module is plugged into.

Digital Input/Output	
long	**getDigitalIn**(int slotno,int type) Description: This function is used to obtain 32-bit input data from a digital input module. The 0~31 bits of input data correspond to the 0~31 channels of digital input module respectively. Parameter: slotno : [Input] the slot number where the I/O module is plugged into. type : [Input] output data type, 8, 16 or 32 bit.
int	**setDigitalOut**(int slotno, int type, int value) Description: Output the 32-bit data to a digital output module. The 0~31 bits of output data are mapped into the 0~31 channels of digital output modules respectively. Parameter: slotno : [Input] the slot number where the I/O module is plugged into. type : [Input] output data type, 8, 16 or 32 bit. value : [Input] output data.

Analog Input(i-8017H)	
int	**init8017**(int slotno) Description: This function is used to initialize the I-8017H modules (Analog input module) into the specified slot. Users must execute this function before trying to use other functions within the I-8017H modules. Parameter: slotno : [Input] specified slot of the LinPAC/LinCon system
int	**setChannelGainMode**(int slotno,int channel,int gain,int mode) Description: This function is used to configure the range and mode of the analog input channel for the I-8017H modules in the specified slot before using the ADC (analog to digital converter). Parameter: slotno : [Input] Specify the slot in the LinPAC/LinCon system channel : [Input] Specify the I-8017H channel (Range: 0 to 7) Specify the I-8017HS channel (Range: 0 to 15) gain : [Input] input range: 0: +/- 10.0V, 1: +/- 5.0V, 2: +/- 2.5V, 3: +/- 1.25V, 4: +/- 20mA. mode : [Input] 0: normal mode (polling)
float	**getAnalogIn**(int slotno,int channel,int gain,int mode) Description: Obtains the calibrated analog input value in the Float format directly from the analog i8017H input modules. Parameter: slotno : [Input] specified slot of the LinPAC/LinCon system channel : [Input] Specify the I-8017H channel (Range: 0 to 7) Specify the I-8017HS channel (Range: 0 to 15) gain : [Input] input range: 0: +/- 10.0V, 1: +/- 5.0V, 2: +/- 2.5V, 3: +/- 1.25V, 4: +/- 20mA. mode : [Input] 0: normal mode (polling) Return Value: The analog input value in Calibrated Float format.
float	**getAnalogIn**(int slotno) Description: Obtains the calibrated analog input value in the Float format directly from the analog I-8017H input modules. Parameter: slotno : [Input] specified slot of the LinPAC/LinCon system Return Value: The analog input value in Calibrated Float format.

Analog Output(i-8024H)	
int	**init8024**(int slotno) Description: This function is used to initialize the I-8024 module in the specified slot. You must implement this function before you try to use the other I-8024 functions. Parameter: slotno : [Input] Specify the LinPAC/LinCon system slot
int	**setCurrentOut**(int slotno, int channel, float value) Description: This function is used to initialize the I-8024module in the specified slot for current output. Users must call this function before trying to use the other I-8024 functions for current output. Parameter: slotno : [Input] Specify the LinPAC-8000 system slot channel : [Input] Output channel (Range: 0 to 3) value : [Input] Output data with engineering unit (Current Output: 0~20 mA)
int	**setVoltageOut**(int slotno, int channel, float value) Description: This function is used to send the voltage float value to the I-8024 module with the specified channel and slot in the LinPAC/LinCon system. Parameter: slotno : [Input] Specified the LinPAC/LinCon system slot channel : [Input] Output channel (Range: 0 to 3) value : [Input] Output data with engineering unit (Voltage Output: -10~ +10)

1.3.4 Class IoBuf

java.lang.Object
|
+--com.icpdas.comm.IoBuf

The IoBuf class provides a variable that the high level remote I/O modules control method use.

Field Summary

dwBuf, **fBuf**, **szSend**, **szReceive**

Field Detail

dwBuf

public int dwBuf[]

Double word length matrix for module control.

fBuf

public float fBuf[]

Floating matrix for module control.

szSend

public java.lang.String szSend

Send a Command string to the module.

szReceive

public java.lang.String szSend

Receives a string from the module.

2. I-7k Modules DIO Control Demo

This demo – **i7kdio.java** will illustrate how to control the DI/DO with the I-7065 module (5 DO channels and 4 DI channels). The address and baudrate of I-7065 module in the RS-485 network are 01 and 9600 separately.

The result of this demo lets DO channel 0 ~ 5 output and DI channel 2 input. The source code of this demo program is as follows：

```
import java.io.*;                       //For System.in.read()
import com.icpdas.comm.*;               //ICP DAS communication packages
public class i7kdio
{
    public static void main(String[] args) throws java.io.IOException
    {
        int rev;
        int fd;
        byte a[] = new byte[100];
        Comm comm1 = new Comm();        //ICP DAS communication object
        IoBuf i7kBuf = new IoBuf();     //control matrix

        rev = comm1.open(3,9600,comm1.DATABITS_8,comm1.PARITY_NONE,
                         comm1.STOPBITS_1);          //open serial port
        if (rev!=0) System.out.println("Open port error code : "+rev);
        else{
            i7kBuf.dwBuf[0] = 3;                    //Serial Port no
            i7kBuf.dwBuf[1] = 1;                    //Address
            i7kBuf.dwBuf[2] = 0x7065 ;              //0x7060; //module name
            i7kBuf.dwBuf[3] = 0;                    //check sum disable
            i7kBuf.dwBuf[4] = 100;                  //Timeout 100ms
            i7kBuf.dwBuf[5] = 31;                   //DO Value
            i7kBuf.dwBuf[6] = 1;                    //Enable String Debug

            rev = comm1.setDigitalOut(i7kBuf);      //DO 7065 module
            System.out.println("DO Send = "+ i7kBuf.szSend);

            if (rev!=0) System.out.println("Digital Out Error Code : "+ rev+"\n");

            rev = comm1.getDigitalIn(i7kBuf);       //Digital Input to i7065 module
            System.out.println("DI Send = "+ i7kBuf.szSend+"\n");

            if (rev!=0) System.out.println("Digital In Error Code : "+ rev);
            else System.out.println("DI ACK : "+ i7kBuf.dwBuf[5]);
        }
        comm1.close(3);
    }
}
```

Follow the below steps to achieve the desired result：

STEP 1：(Write i7kdio.java)

Copy the above source code and save it with the name - **i7kdio.java**. Remember! the file name – i7kdio.java must be the same as the class name. Or get the file from C:\cygwin\LinCon8k\examples\Java\i7k.

STEP 2：(Transfer i7kdio.java to LinPAC-8000)

Here we will introduce two methods to accomplish step 2.

< Method One > Using FTP Software

(1) Open a FTP Software and add a ftp site of the LinPAC-8000. The **User_Name** and **Password** default value is **“ root ”**. (refer to Fig.2-1).

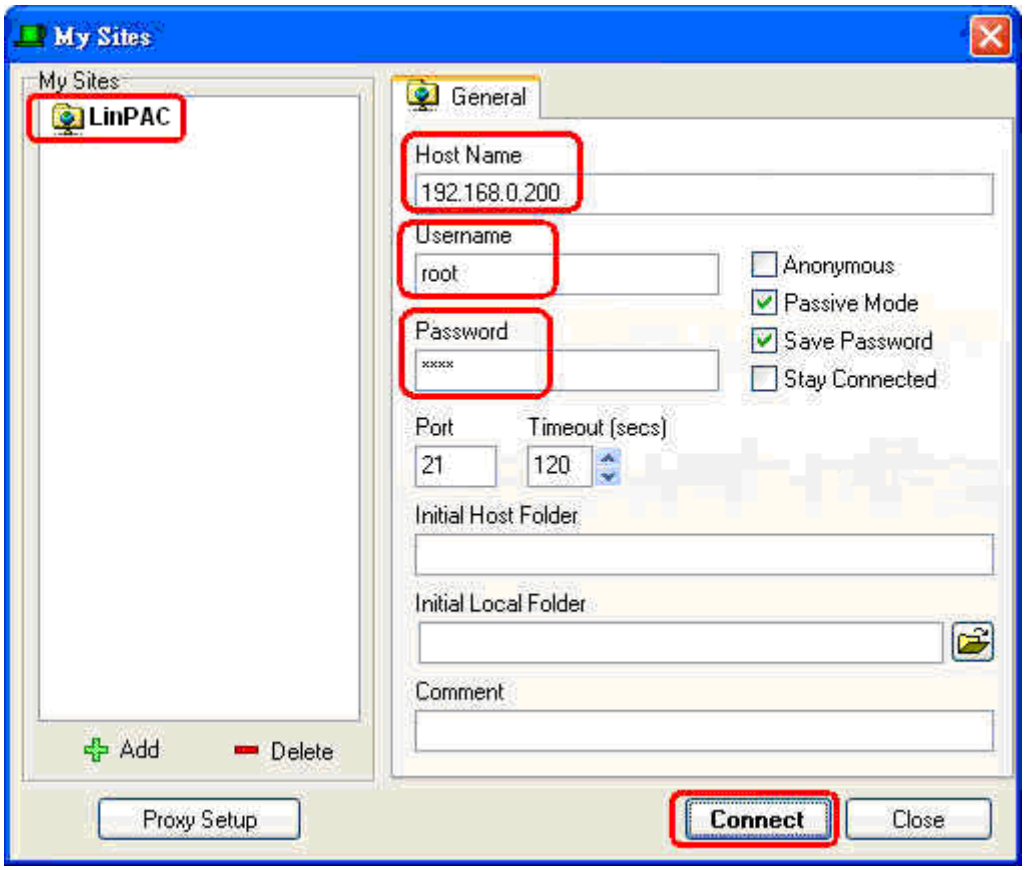

Fig.2-1

(2) Click the **"Connect"** button to connect to the ftp server on the LinPAC-8000. Then upload the file – **i7kdio.java** to the LinPAC-8000. (refer to Fig.2-2).

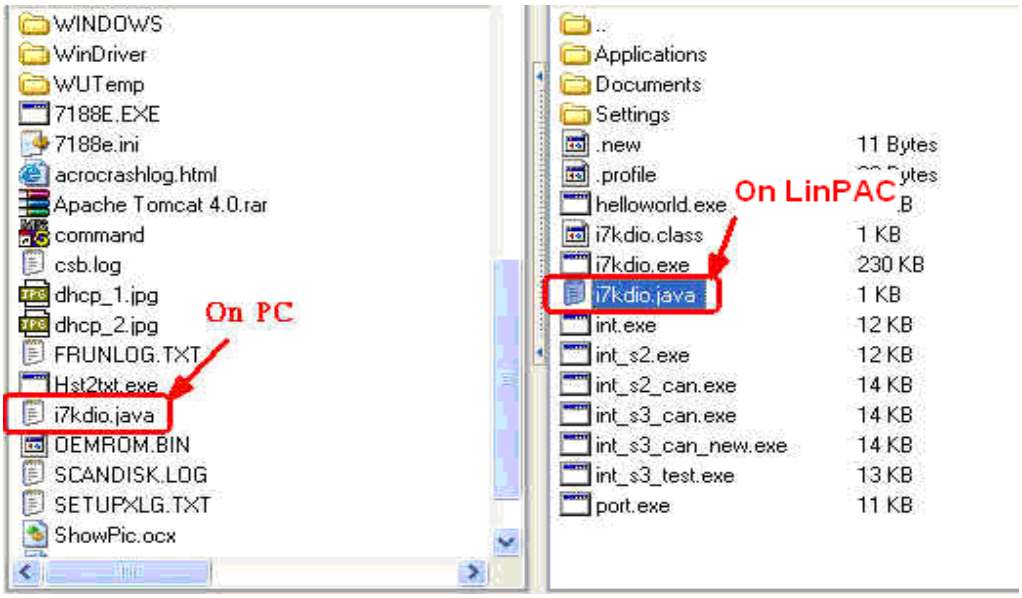

Fig.2-2

< Method Two > Using DOS Command Prompt

Open a DOS Command Prompt and type in the ftp IP Address of LinPAC-8000 to connect to the ftp server of your LinPAC-8000. Then input your **User Name** and **Password** (**root** is the default value) to login to the LinPAC-8000. Type in **bin** to make transfer files by using the binary mode. Then type **put c:/i7kdio.java i7kdio.java** to transfer the i7kdio.exe file to the LinPAC-8000. After the message of Transfer complete has appeared, the transfer will be complete.(refer to Fig. 2-3)

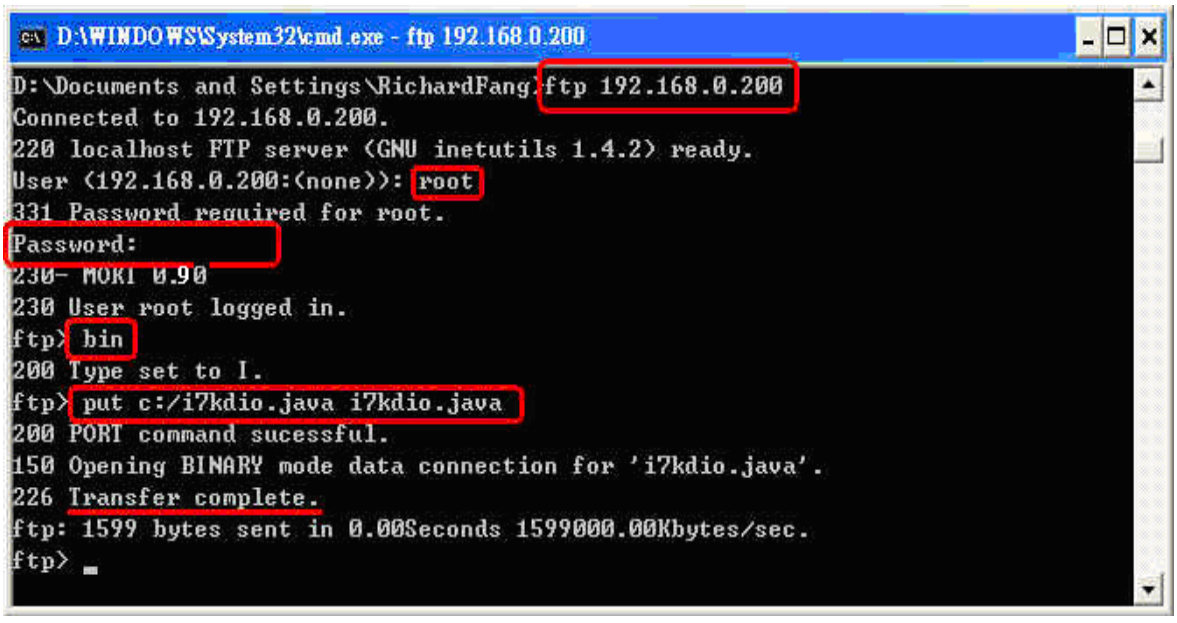

Fig. 2-3

STEP 3：**(Telnet to LinPAC-8000)**

Type the telnet IP Address of LinPAC-8000 to remote control LinPAC-8000 and input your **User Name** and **Password** (**root** is the default value) to login to the LinPAC-8000. (refer to Fig. 2-4 and Fig. 2-5)

Fig. 2-4

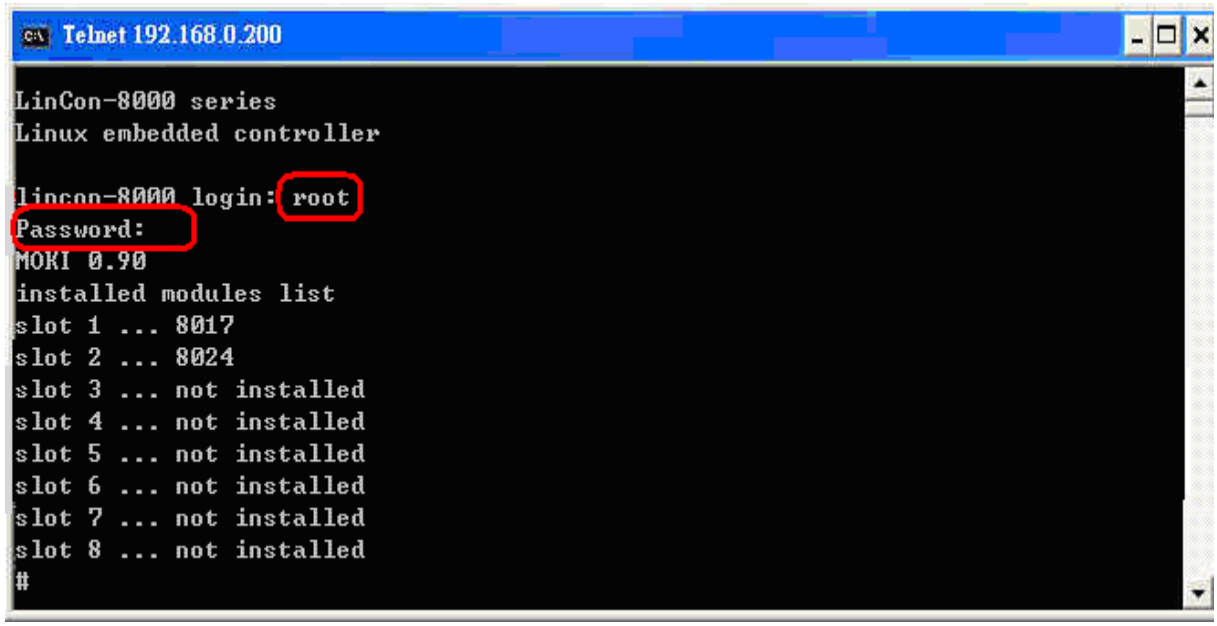

Fig. 2-5

STEP 4：**(Compile i7kdio.java to i7kdio.class and execute i7kdio.class)**

Type in **javac i7kdio.java** to compile i7kdio.java to i7kdio.class. The process of compiling in the LinPAC-8000 is slower than that in the PC. Then type **java i7kdio** to execute the i7kdio.class. (refer to Fig. 2-6)

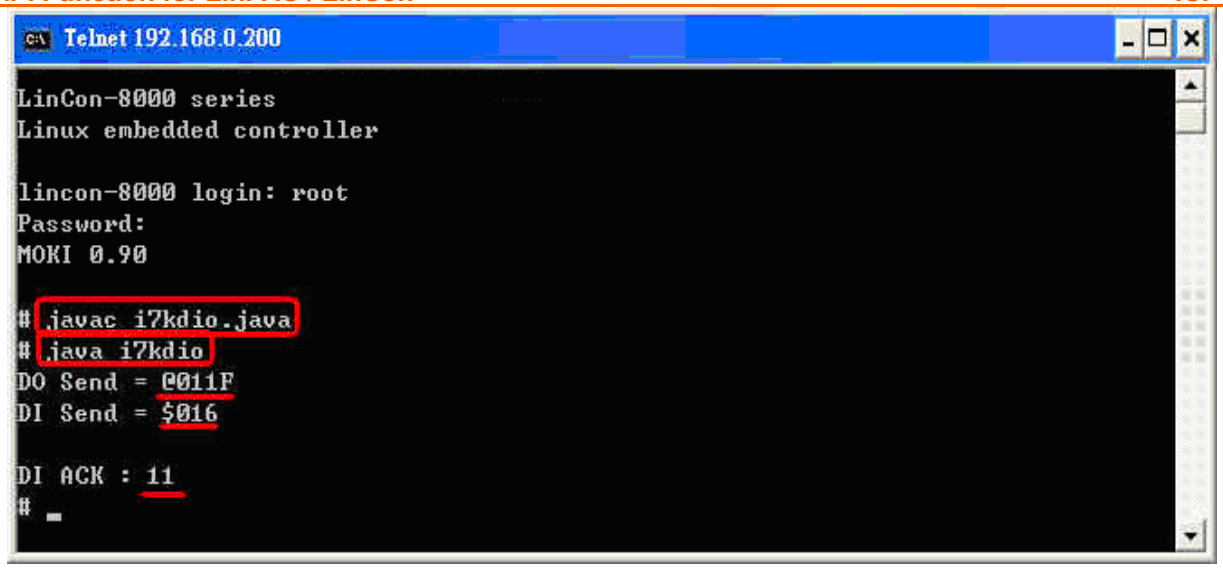

Fig. 2-6

" @**011F** " is an ICP DAS command type where **01** means address and **1F** means the DO channel 0 ~ 5 will output. " **$016** " is used to read back all DI Channels status. " **DI ACK : 11** (15-2^**2**)" means there is an input in DI channel 2.

3. I-7K Modules AIO Control Demo

This demo – **i7kaio.java** will illustrate how to control the AI/AO with the I-7012 (1 AI channels) and I-7022 module (2 AO channel). The address of I-7012 and I-7022 module in the RS-485 network are 11 and 10 separately and the baudrate is 9600.

The result of this demo is to allow the AO channel 0 of I-7022 to output at 3.5V and AI channel of I-7012 input from the AO channel 0 of I-7022. The source code of this demo program is provided below：

```
import java.io.*;
import com.icpdas.comm.*;

public class i7kaio
{
    public static void main(String[] args) throws java.io.IOException
    {
        int rev;
        int fd;
        float ao=3.5f;
        byte a[] = new byte[100];
        Comm comm1 = new Comm();
        IoBuf i7kBuf = new IoBuf();

        rev = comm1.open(3,9600,comm1.DATABITS_8,comm1.PARITY_NONE,
                         comm1.STOPBITS_1);

        if (rev!=0) System.out.println("Open port error code : "+rev);
        else{
            i7kBuf.dwBuf[0] = 3;                    //Serial Port
            i7kBuf.dwBuf[1] = 10;                   //Address
            i7kBuf.dwBuf[2] = 0x7022;               //0x7022; //module name
            i7kBuf.dwBuf[3] = 0;                    //check sum disable
            i7kBuf.dwBuf[4] = 100;                  //Timeout 100ms
            i7kBuf.dwBuf[6] = 1;                    //Enable String Debug

            //I-7022 AO
            i7kBuf.fBuf[0] = ao;                    //Output AO Value
            rev = comm1.setAnalogOut(i7kBuf);       //AO Output
            if (rev!=0) System.out.println("Analog Out Error Code : "+ rev);
            System.out.println("AO Send = "+ i7kBuf.szSend);
            System.out.println("Press any key !!");
            System.in.read(a);

            //I-7012 AI
            i7kBuf.dwBuf[1] = 11;                   //Address
            i7kBuf.dwBuf[2] = 0x7012;               //0x7012; //module name
```

```
            rev = comm1.getAnalogIn(i7kBuf);
            if (rev!=0) System.out.println("AI Error Code : "+ rev);

            //System.out.println("szSend = "+ i7kBuf.szSend);
            System.out.println("AI Receive : "+i7kBuf.fBuf[0]);
        }
        comm1.close(3);
    }
}
```

All the steps from programming to execution are the same as those in the section 2. The result of execution refers to Fig. 3-1.

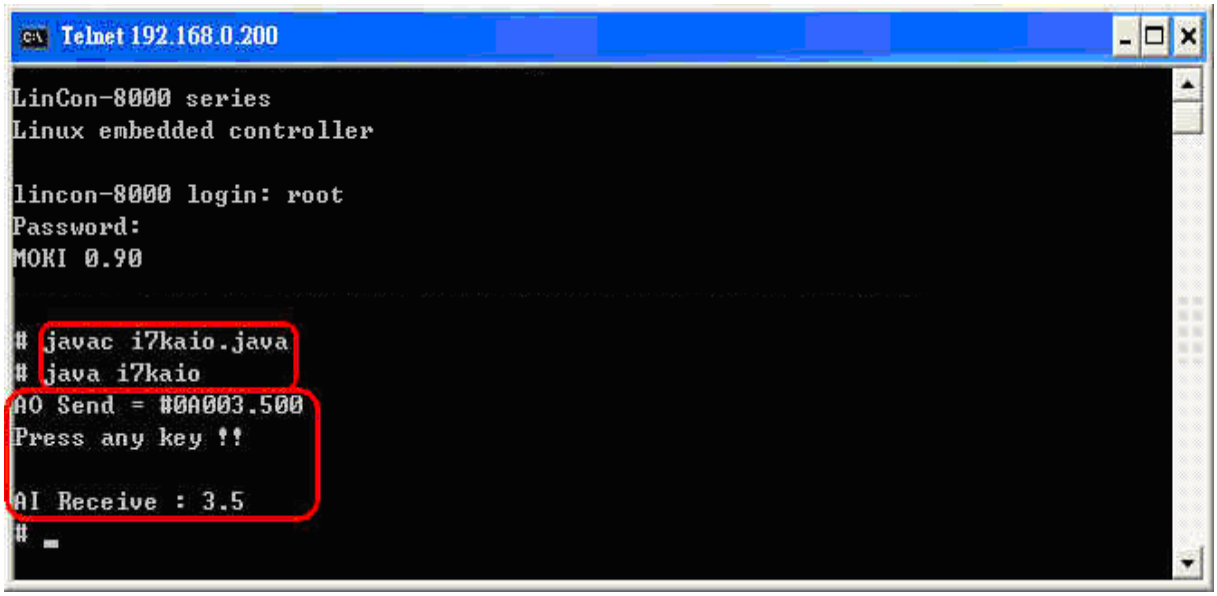

Fig. 3-1

4. I-87K Modules DIO Control Demo

When using the I-87K modules for I/O control in the LinPAC-8000, the program will be a little different depending on the location of I-87k modules. There are three conditions for the location of I-87K modules：

(1) When I-87K DIO modules are **in the LinPAC-8000 slots**, please refer to the demo in the section 4.1.

(2) When I-87K DIO modules are **in the I-87K I/O expansion unit slots**, please refer to the demo in the section 4.2.

(3) When I-87K DIO modules **in the I-8000 controller slots**, I-87K modules will be regarded as I-8K modules so please refer to the I/O control of I-8K modules in the section 4.3.

4.1 I-87K Modules in slots of LinPAC-8000

I-87057 and I-87051 are plugged in the slot 01 and 02 in the LinPAC-8000 separately. The source code of this demo program –**i87kdio.java** is as follows：

```
import java.io.*;                          //For System.in.read()
import com.icpdas.comm.*;                  //ICP DAS communication packages
import com.icpdas.slot.*;                  //Slot I/O packages

public class i87kdio
{
    public static void main(String[] args) throws java.io.IOException
    {
        int rev;
        int fd;
        int i=1;
        byte a[] = new byte[100];
        Comm comm1 = new Comm();           //ICP DAS communication object
        Slot slot1 = new Slot();
        IoBuf i87kBuf = new IoBuf();       //control matrix
        slot1.open(0);                     //open slot 0 fot i87k slot select
        //open serial port
        rev = comm1.open(0,115200,comm1.DATABITS_8,comm1.PARITY_NONE,
                         comm1.STOPBITS_1);

        if (rev!=0) System.out.println("Open port error code : "+rev);
        else{
            i87kBuf.dwBuf[0] = 1;                  //Serial Port
            i87kBuf.dwBuf[1] = 0;                  //Module Address
```

```
                i87kBuf.dwBuf[3] = 0;                    //check sum disable
                i87kBuf.dwBuf[4] = 100;                  //Timeout 100ms
                i87kBuf.dwBuf[6] = 1;                    //Enable String Debug

                while(a[0]!=113) {
                    i87kBuf.dwBuf[2] = 0x87057 ;         //module name
                    i87kBuf.dwBuf[5] =i;                 //Digital Output Value
                    slot1.setChangeToSlot(1);            //change to slot 1
                    rev = comm1.setDigitalOut(i87kBuf);
                    System.out.println("szSend = "+ i87kBuf.szSend +
                                       " szReceive = "+i87kBuf.szReceive);
                    if (rev!=0) System.out.println("Digital Out Error Code : "+ rev);
                    i87kBuf.dwBuf[2] = 0x87051 ;         //module name
                    slot1.setChangeToSlot(2);            //change to slot 2
                    rev = comm1.getDigitalIn(i87kBuf);
                    System.out.println("szSend = "+ i87kBuf.szSend +
                                       " szReceive = "+i87kBuf.szReceive);
                    if (rev!=0) System.out.println("Digital In Error Code : "+ rev);
                    else System.out.println("Digital In : "+ i87kBuf.dwBuf[5]);
                    System.in.read(a);
                    i=(i>255)?1:(i<<1);
                }
            }
            slot1.close(0);
            comm1.close(0);
            System.out.println("End of program");
        }
}
```

4.2 I-87K Modules in slots of I-87K I/O expansion unit

The address of I-87057 and I-87051 in the RS-485 network are set to be 01 and 02 separately and the baudrate is 9600 via the DCON Utility. The source code of this demo program – **i87kdio_87k.java** is as follows：

```
import java.io.*;                          //For System.in.read()
import com.icpdas.comm.*;                  //ICP DAS communication packages

public class i87kdio_87k
{
    public static void main(String[] args) throws java.io.IOException
    {
        int rev;
        int fd;
        int i=65535;                       //(int)(Math.pow(2,16)-1)
        byte a[] = new byte[100];
        Comm comm1 = new Comm();           //ICP DAS communication object
```

```
        IoBuf i87kBuf = new IoBuf();          //control matrix

        rev = comm1.open(3,9600,comm1.DATABITS_8,comm1.PARITY_NONE,
                         comm1.STOPBITS_1);//open serial port

        if (rev!=0) System.out.println("Open port error code : "+rev);
        else{
            i87kBuf.dwBuf[0] = 3;                  //Serial Port
    i87kBuf.dwBuf[3] = 0;                  //check sum disable
            i87kBuf.dwBuf[4] = 100;                //Timeout 100ms
            i87kBuf.dwBuf[6] = 1;                  //Enable String Debug

            //I-87057 DO
            i87kBuf.dwBuf[1] = 1;              //Module Address
            i87kBuf.dwBuf[2] = 0x87057 ;       //module name
            i87kBuf.dwBuf[5] =i;               //Digital Output Value

            rev = comm1.setDigitalOut(i87kBuf);

            System.out.println("DO Send = "+ i87kBuf.szSend);

            if (rev!=0) System.out.println("Digital Out Error Code : "+ rev);

            //I-87051 DI
            i87kBuf.dwBuf[1] = 2;              //Module Address
            i87kBuf.dwBuf[2] = 0x87051 ;       //0x87051 //module name

            rev = comm1.getDigitalIn(i87kBuf);
            System.out.println("szSend = "+ i87kBuf.szSend);
            if (rev!=0) System.out.println("Digital In Error Code : "+ rev);
            else System.out.println("DI Receive: "+ i87kBuf.dwBuf[5]);
        }
        comm1.close(3);
    }
}
```

All the steps from writing to execution are the same as those in the section 2. The result of execution refers to Fig. 4-1.

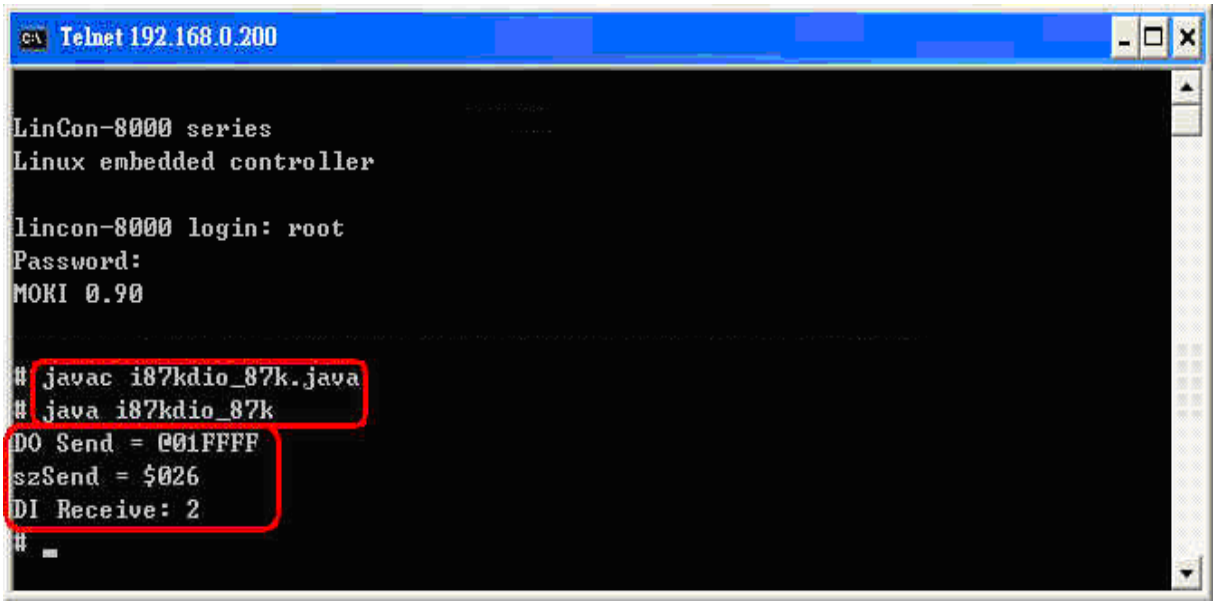

Fig. 4-1

4.3 I-87K Modules in slots of I-8000 Controller

If the I-87K DIO modules are in the I-8000 controller slots, the I-87K modules will be regarded as I-8K modules. In this case, users can refer to DI/DO control demo of I-8K modules in the section 4.2.

5. I-87K Modules AIO Control Demo

When using I-87K modules for I/O control of the LinPAC-8000, the program will be a little different depending on the location of I-87K modules. There are three conditions for the location of I-87k modules：

(1) When I-87K AIO modules are **in the LinPAC-8000 slots**, please refer to the demo in the section 5.1.

(2) When I-87K AIO modules are **in the I-87k I/O expansion unit slots**, please refer to the demo in the section 5.2.

(3) When I-87K AIO modules **in the I-8000 controller slots**, I-87K modules will be regarded as I-8K modules. In this case users can refer to I/O control demo of I-8k modules in the section 5.2

5.1 I-87K Modules in slots of LinPAC-8000

I-87024 and I-87017 are plugged in the slot 01 and 02 in the LinPAC-8000 separately. The source code of this demo program –**i87kaio.java** is as follows：

```
import java.io.*;                          //For System.in.read()
import com.icpdas.comm.*;                  //ICP DAS communication packages
import com.icpdas.slot.*;                  //Slot I/O packages

public class i87kaio
{
    public static void main(String[] args) throws java.io.IOException
    {
        int rev;
        int fd;
        float aoValue = 0.5F;
        byte a[] = new byte[100];
        Comm comm1 = new Comm();           //ICP DAS communication object
        Slot slot1 = new Slot();
        IoBuf i87kBuf = new IoBuf();       //control matrix
        slot1.open(0);                     //open slot 0 fot i87k slot select

        rev = comm1.open(1,115200,comm1.DATABITS_8,comm1.PARITY_NONE,
                         comm1.STOPBITS_1);//open serial port

        if (rev!=0) System.out.println("Open port error code : "+rev);
        else{
```

```
                i87kBuf.dwBuf[0] = 1;                   //Serial Port
                i87kBuf.dwBuf[1] = 0;                   //Address
                i87kBuf.dwBuf[3] = 0;                   //check sum disable
                i87kBuf.dwBuf[4] = 100;                 //Timeout 100ms
                i87kBuf.dwBuf[5] = 0;                   //channel 0
                i87kBuf.dwBuf[6] = 1;                   //Enable String Debug

                while(a[0]!=113) {
                    i87kBuf.dwBuf[2] = 0x87024 ;            //module name
                    i87kBuf.fBuf[0] = aoValue;              //AO output value
                    slot1.setChangeToSlot(1);               //change to slot 1

                    rev = comm1.setAnalogOut(i87kBuf);  //Set Analog Output Value
                    if (rev!=0) System.out.println("Analog Out Error Code : "+ rev);

                    System.out.println("szSend = "+ i87kBuf.szSend +
                                       " szReceive = "+i87kBuf.szReceive);
                    System.out.println("Press 'Enter' to Read AI");
                    System.in.read(a);

                    i87kBuf.dwBuf[2] = 0x87017 ;            //module name
                    slot1.setChangeToSlot(2);               //change to slot 2

                    rev = comm1.getAnalogIn(i87kBuf);    //Get Analog Output Value
                    if (rev!=0) System.out.println("Analog In Error Code : "+ rev);
                    System.out.println("szSend = "+ i87kBuf.szSend +
                                       " szReceive = "+i87kBuf.szReceive);
                    System.out.println("Analog In Value : "+i87kBuf.fBuf[0]);
                    System.in.read(a);
                    aoValue=(aoValue>8.0F)?0.5F:(aoValue + 1.0F);
                }
            }
            slot1.close(0);
            comm1.close(1);
            System.out.println("End of program");
        }
}
```

5.2 I-87K Modules in slots of I-87k I/O expansion unit

The address of I-87024 and I-87017 in the RS-485 network are set to be 06 and 07 separately and the baudrate is 9600 via the DCON Utility. The source code of this demo program – **i87kaio_87.java** is as follows：

```
import java.io.*;                       //For System.in.read()
import com.icpdas.comm.*;               //ICP DAS communication packages

public class i87kaio_87k
```

```
{
    public static void main(String[] args) throws java.io.IOException
    {
        int rev;
        int fd;
        float ao=2.5f;
        byte a[] = new byte[100];
        Comm comm1 = new Comm();        //ICP DAS communication object
        IoBuf i87kBuf = new IoBuf();    //control matrix

        rev = comm1.open(3,9600,comm1.DATABITS_8,comm1.PARITY_NONE,
                         comm1.STOPBITS_1);//open serial port

        if (rev!=0) System.out.println("Open port error code : "+rev);
        else{
            i87kBuf.dwBuf[0] = 3;                   //Serial Port
            i87kBuf.dwBuf[3] = 0;                   //check sum disable
            i87kBuf.dwBuf[4] = 100;                 //Timeout 100ms
            i87kBuf.dwBuf[5] = 2;                   //channel 2
            i87kBuf.dwBuf[6] = 1;                   //Enable String Debug

            //I-87024 AO
            i87kBuf.dwBuf[1] = 6;                   //Address
            i87kBuf.dwBuf[2] = 0x87024 ;            //0x87024; //module name
            i87kBuf.fBuf[0] = ao;
            rev = comm1.setAnalogOut(i87kBuf);  //Set Analog Output Value
            if (rev!=0) System.out.println("Analog Out Error Code : "+ rev);

            System.out.println("AO Send = "+ i87kBuf.szSend);
            System.out.println("Press any key to read AI Data !!");
            System.in.read(a);

            //I-87017 AI
            i87kBuf.dwBuf[1] = 7;                   //Address
            i87kBuf.dwBuf[2] = 0x87017 ;            //0x87017; //module name
            rev = comm1.getAnalogIn(i87kBuf);   //Get Analog Output Value
            if (rev!=0) System.out.println("Analog In Error Code : "+ rev);
            System.out.println("AI Receive : "+i87kBuf.fBuf[0]);
        }
        comm1.close(3);
    }
}
```

All the steps from writing to execution are the same as those in the section 2. The result of execution refers to Fig.5-1

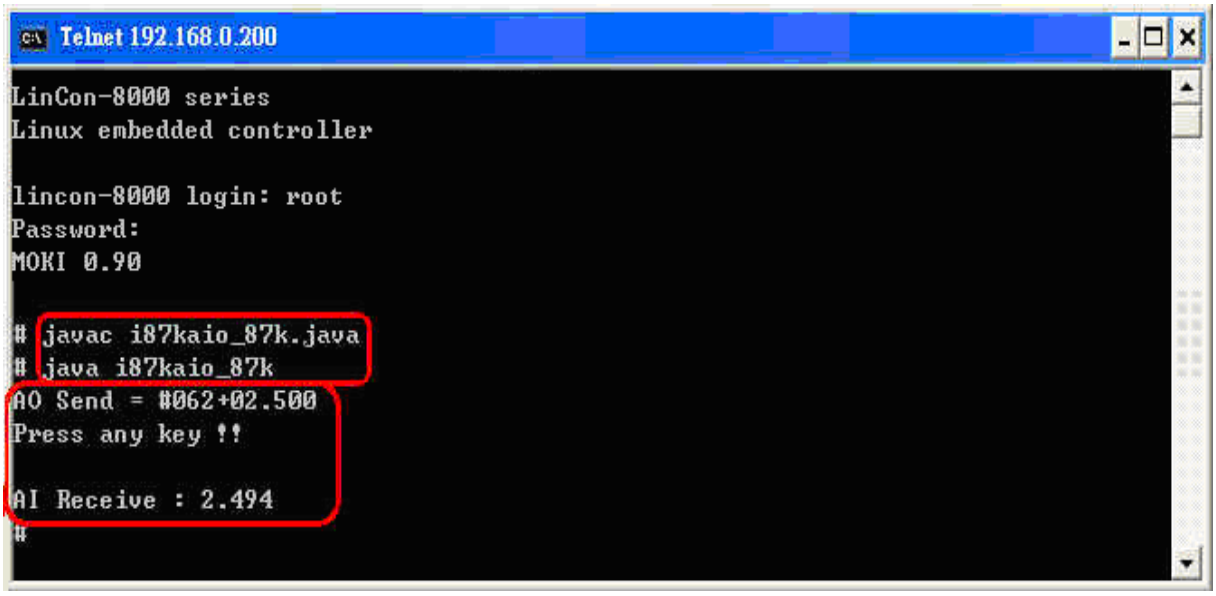

Fig.5-2

5.3 I-87k Modules in slots of I-8000 Controller

If the I-87K AI/AO modules are in the I-8000 controller slots, I-87K modules will be regarded as I-8K modules and please refer to AI/AO control demo of I-8k modules in the section 5.2.

6. I-8K Modules DIO Control Demo

When using I-8K modules for I/O control of the LinPAC-8000, the program will be a little different depending on the location of I-8k modules. There are two conditions for the location of I-8k modules：

(1) When I-8K DIO modules are **in the LinPAC-8000 slots.** Please refer to I/O control demo of I-8k DIO modules in the section 6.1

(2) When I-8K DIO modules **in I-8000 controller slots.** Please refer to I/O control demo of I-8K DIO modules in the section 6.2

6.1 I-8K Modules in slots of LinPAC-8000

I-8055 is plugged in the slot 3 of LinPAC-8000. The source code of this demo program – **i8kdio.java** is as follows：

```
import java.io.*;                          //For System.in.read()
import com.icpdas.slot.*;                  //Slot I/O packages

public class i8kdio
{
    public static void main(String[] args) throws java.io.IOException
    {
        int rev;
        long diValue;
        int i=1;
        byte a[] = new byte[100];

        Slot slot1 = new Slot();

        rev = slot1.open(3);                          //open slot 1 for digital in/out
        if (rev!=0) System.out.println("Open port error code : "+rev);
        else{
            while(a[0]!=113) {
                rev = slot1.setDigitalOut(3,8,i);
                if (rev!=0) System.out.println("Digital Out Error Code : "+ rev);
                diValue = slot1.getDigitalIn(3,8);
                System.out.println("Digital Out :" + i + "    Digital In : "+ diValue);
                System.in.read(a);
                i=(i>255)?1:(i<<1);
```

```
            }
        }
        slot1.close(1);
        System.out.println("End of program");
    }
}
```

6.2 I-8K Modules in slots of I-8000 Controller

In this section, the demo program - **i8kdio_8k.java** will introduce how to control the DI/DO with the I-8055 (8 DO channels and 8 DI channels) module. Please follow the below steps to configure the hardware correctly：

(1) Put the I-8055 module into slot 0 on the I-8000 controller.

(2) Install the **R232_300.exe** to flash memory of I-8000 controller as firmware.

(3) Connect **com3** on the LinPAC-8000 to the com1 on the I-8000 controller with the RS-232 cable.

The address of I-8000 controller is 01 and the baudrate is 115200 and they can be modified via the DCON Utility. The result of this demo allows DO channels 0 ~7 of I-8055 output and DI channel 0 of I-8055 input. The source code of this demo program is as follows：

```
import java.io.*;                       //For System.in.read()
import com.icpdas.comm.*;               //ICP DAS communication packages

public class i8kdio_8k
{
    public static void main(String[] args) throws java.io.IOException
    {
        int rev;
        int fd;
        int i=255;
        byte a[] = new byte[100];
        Comm comm1 = new Comm();        //ICP DAS communication object
        IoBuf i87kBuf = new IoBuf();    //control matrix

        rev = comm1.open(3,115200,comm1.DATABITS_8,comm1.PARITY_NONE,
                         comm1.STOPBITS_1);//open serial port
        if (rev!=0) System.out.println("Open port error code : "+rev);
        else {
            i87kBuf.dwBuf[0] = 3;           //Serial Port
            i87kBuf.dwBuf[1] = 1;           //Address
            i87kBuf.dwBuf[3] = 0;           //check sum disable
            i87kBuf.dwBuf[4] = 100;         //Timeout 100ms
            i87kBuf.dwBuf[6] = 1;           //Enable String Debug
```

```
            //I-8055 DO
            i87kBuf.dwBuf[2] = 0x8055 ;     //0x8055; //module name
            i87kBuf.dwBuf[5] =i;            //Digital out
            i87kBuf.dwBuf[7] = 0;           //Slot number
            rev = comm1.setDigitalOut(i87kBuf);//Digital Output Value
            System.out.println("DO Send = "+ i87kBuf.szSend);
            if (rev!=0) System.out.println("Digital Out Error Code : "+ rev);

            //I-8055 DI
            rev = comm1.getDigitalIn(i87kBuf);// Digital Input Value
            System.out.println("DI Send = "+ i87kBuf.szSend);
            if (rev!=0) System.out.println("Digital In Error Code : "+ rev);
            else System.out.println("DI Receive : "+ i87kBuf.dwBuf[5]);                }
        comm1.close(3);
    }
}
```

All the steps from writing to execution are the same as those in the section 2. The result of execution refers to Fig.6-1

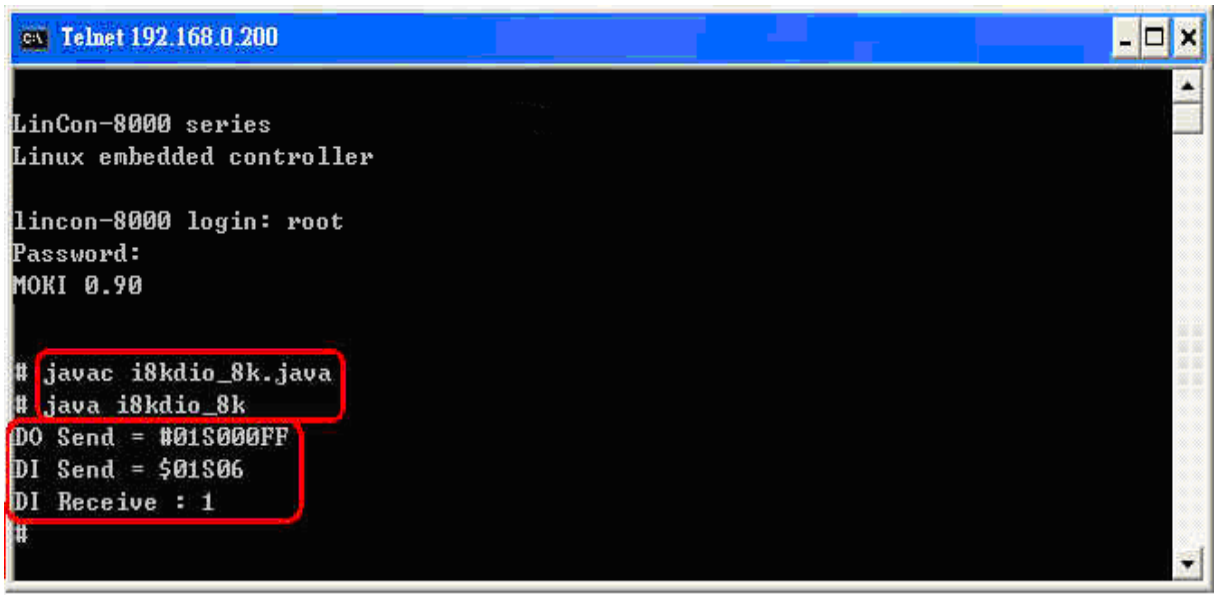

Fig.6-1

7. I-8K Modules AIO Control Demo

When using I-8K modules for I/O control of the LinPAC-8000, the program will be a little different depending on the location of I-8k modules. There are two conditions for the location of I-8k modules：

(1) When I-8K AIO modules are **in the LinPAC-8000 slots.** Please refer to I/O control of I-8k AIO modules in the section 7-1

(2) When I-8K AIO modules **in the I-8000 controller slots.** Please refer to I/O control of I-8k AIO modules in the section 7-2

7.1 I-8K Modules in slots of LinPAC-8000

I-8024 (4 AO channels) and I-8017 (8 AI channels) is plugged in the slot 1 and slot 2 of LinPAC-8000. The source code of this demo program – **i8kaio.java** is as follows：

```
import java.io.*;                          //For System.in.read()
import com.icpdas.slot.*;                  //Slot I/O packages

public class i8kaio
{
    public static void main(String[] args) throws java.io.IOException
    {
        int rev;
        float aiValue;
        float i=1;
        byte a[] = new byte[100];
        Slot slot1 = new Slot();

        rev = slot1.open(1);               //open slot 1 for analog output
        if (rev!=0) System.out.println("Open slot error code : "+rev);
        else{
            rev = slot1.init8024(1);
            rev = slot1.open(2);           //open slot 2 for analog input
            if (rev!=0) System.out.println("Open slot error code : "+rev);
            else
            {
                rev = slot1.init8017(2);
                rev = slot1.setChannelGainMode(2,0,0,0);

                while(a[0]!=113)
                {
                    rev = slot1.setVoltageOut(1,0,i);
```

```
                        if (rev!=0) System.out.println("Analog Out Error Code : "+ rev);
                        aiValue = slot1.getAnalogIn(2);
                        System.out.println("Analog Out :" + i + " Analog In : "+ aiValue);
                        System.in.read(a);
                        i=(i>8)?1:(i+1.5F);
                    }
                }
            }
            slot1.close(1);
            slot1.close(2);
            System.out.println("End of program");
        }
}
```

7.2 I-8K Modules in slots of I-8000 Controller

In this section, the demo program - **i8kaio_8k.java** will illustrate how to control the AI/AO with the I-8024 (4 AO channels) module and I-8017 (8 AI channels) module when they are plugged into the slot 0 and slot 1 on the I-8000 controller separately. Please follow the below steps to configure the hardware correctly：

(1) Put the I-8024 and I-8017 modules into slot 0 and slot 1 of the I-8000 controller.

(2) Install the R232_300.exe to flash memory of the I-8000 controller as firmware.

(3) Connect **com3** on the LinPAC-8000 to com1 on the I-8000 controller with RS-232 cable.

The address and baudrate of I-8000 controller are 01 and 115200 and they can be modified via the DCON Utility. The result of this demo allows the channel 0 of I-8024 output at 3.5V and the AI channel 0 of I-8017H input from AO channel 0 of I-8024. The source code of this demo program is as follows：

```
import java.io.*;                    //For System.in.read()
import com.icpdas.comm.*;            //ICP DAS communication packages

public class i8kaio_8k
{
    public static void main(String[] args) throws java.io.IOException
    {
        int rev;
        int fd;
        float ao=5.5f;
        byte a[] = new byte[100];
        Comm comm1 = new Comm();          //ICP DAS communication object
        IoBuf i8kBuf = new IoBuf();       //control matrix
```

```
        rev = comm1.open(3,115200,comm1.DATABITS_8,comm1.PARITY_NONE,
                         comm1.STOPBITS_1); //open serial port

        if (rev!=0) System.out.println("Open port error code : "+rev);
        else {
            i8kBuf.dwBuf[0] = 3;            //Serial Port
            i8kBuf.dwBuf[1] = 1;            //Address
            i8kBuf.dwBuf[3] = 0;            //check sum disable
            i8kBuf.dwBuf[4] = 100;              //Timeout 100ms
            i8kBuf.dwBuf[6] = 1;            //Enable String Debug

            //I-8024 AO
            i8kBuf.dwBuf[2] = 0x8024 ;      //0x8024; //module name
            i8kBuf.dwBuf[7] = 0;            //Slot number
            i8kBuf.fBuf[0] = ao;

            rev = comm1.setAnalogOut(i8kBuf);       //Set AO Value
            if (rev!=0) System.out.println("Analog Out Error Code : "+ rev);
            System.out.println("AO Send = "+ i8kBuf.szSend);
            System.out.println("Press any key !!");
            System.in.read(a);


            //I-8017 AI
            i8kBuf.dwBuf[2] = 0x8017 ;
            i8kBuf.dwBuf[7] = 1;
            rev = comm1.getAnalogIn(i8kBuf);        //Get AI Value
            if (rev!=0) System.out.println("Analog In Error Code : "+ rev);
            System.out.println("AI Receive : "+i8kBuf.fBuf[0]);
        }
        comm1.close(3);
    }
}
```

All the steps from writing to execution are the same as those in the section 2. The result of execution refers to Fig.7-1

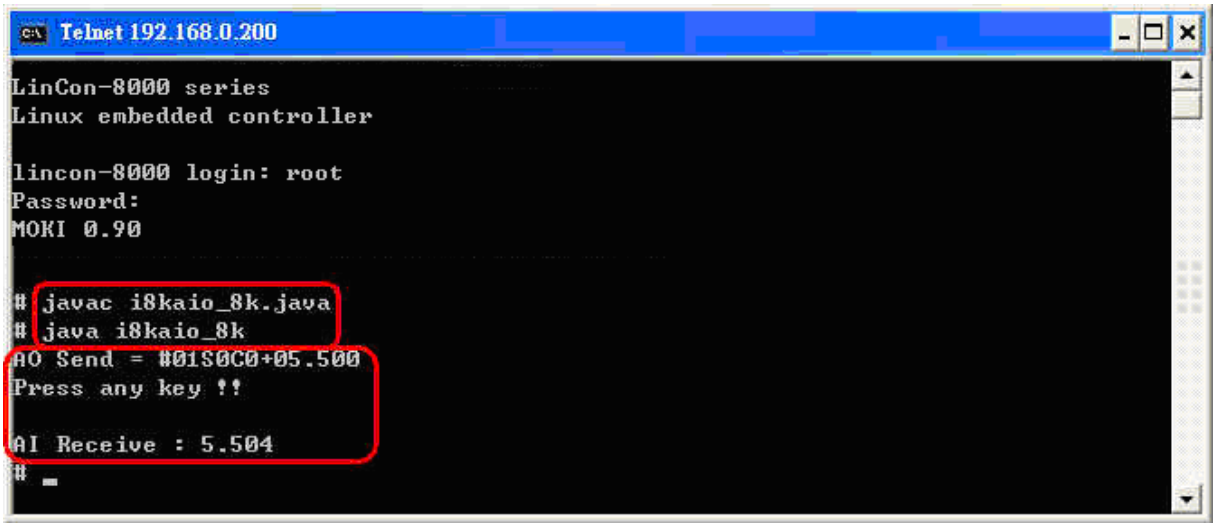

Fig.7-1