Java API Reference Guide

For Linux-based PAC

By ICP DAS

Version 1.0

Contents

1. Introduction of Linux-based PAC 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 Linux-based PAC of ICP DAS, so that users can compile and run Java programs in it, such as LinCon, LinPAC, PDS-8x2,ViewPAC -23(25)A1…, etc. 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-8KW and I-87KW 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-8KW,I-87KW 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.

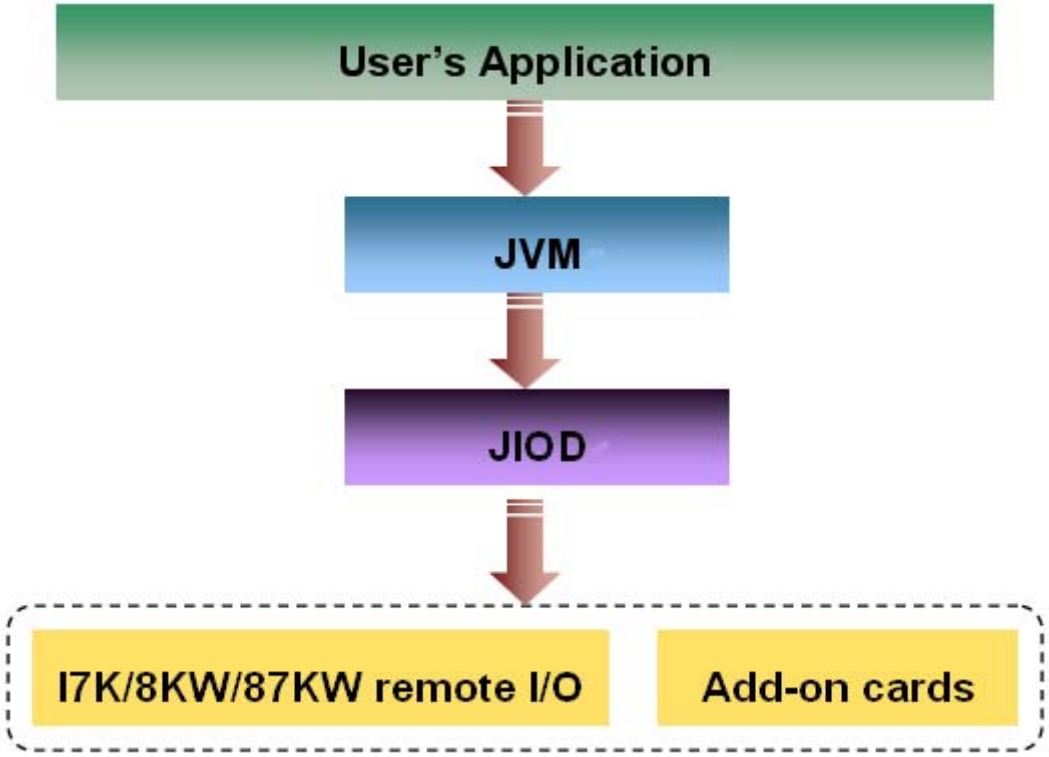

Fig 1-1

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://www.icpdas.com/download/java/index.htm

Using JIOD is very similar to that for C language users because **icpdas.jar** for Java is just like **libi8k.a**(LinPAC library) 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://www.icpdas.com/download/java/index.htm

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-87KW, I-8KW. 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.

Method Summary

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 opened for reading and writing data. Once the application is done with the port, it must call the close method before end program.

Syntax

int open(int slotno)

Description:

This function is used to open and initiate a specifed slot in the Linux-based PAC of ICP DAS. The I-8KW or I-87KW modules 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.

Syntax

void close(int slotno)

Description:

If you have used the function of slot.open() to open the specifed slot in the Linux-based PAC of ICP DAS, you need to use the slot.close() function to close the specifed slot in the Linux-based PAC of ICP DAS. The I-8KW or I-87KW modules 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.

Syntax

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.

Syntax

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.

Example:

```
import    com.icpdas.slot.*;
    public static Slot slot1 = new Slot();
        int data=slot1.inb(1, 0);
        slot_open.setText("data= " + ret);        //Output data= 255
```

Syntax

int setChangeToSlot(int slotno)

Description:

This function is used to dedicate serial control to the specified slots for the control of the I-87KW series. The serial bus in the Linux-based PAC of ICP DAS 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.

Syntax

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.

Ret:

Input data

Example:

```
int slot=1;
unsigned int data;
data=DI_16(slot);
// The I-8053W card is plugged in slot 1 of ViewPAC and has inputs in channel 0 and 1.
// Returned value: data=0xfffC
```

There are two kind of Input type:

Input Type	On State	Off State	Modules
1	LED On, Readback as **1**	LED Off, Readback as **0**	I-8046W, I-8058W
2	LED On, Readback as **0**	LED Off, Readback as **1**	I-8040W, I-8040PW, I-8048W, I-8051W, I-8052W, I-8053W, I-8053PW

Syntax

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-8017HW)

Syntax

int init8017 (int slotno)

Description:

This function is used to initialize the I-8017HW modules (Analog input module) into the specified slot. Users must execute this function before trying to use other functions within the I-8017HW modules.

Parameter:

slotno : [Input] specified slot of the Linux-based PAC of ICP DAS system

Syntax

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-8017HW modules in the specified slot before using the ADC (analog to digital converter).

Parameter:

slotno : [Input] Specify the slot in the Linux-based PAC of ICP DAS system

channel : [Input] Specify the I-8017HW channel (Range: 0 to 7)

Specify the I-8017HW 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)

Syntax

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 I-8017HW input modules.

Parameter:

slotno : [Input] specified slot of the Linux-based PAC of ICP DAS system

channel : [Input] Specify the I-8017HW channel (Range: 0 to 7)

Specify the I-8017HW 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.

Syntax

float getAnalogInAll(int slotno)

Description:

Obtains the calibrated analog input value of all channels in the Float format directly from the analog I-8017HW input modules.

Parameter:

slotno : [Input] specified slot of the Linux-based PAC of ICP DAS system

Return Value:

The analog input value in Calibrated Float format.

Analog Output(I-8024W)

Syntax

int init8024 (int slotno)

Description:

This function is used to initialize the I-8024W 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 Linux-based PAC of ICP DAS system slot

Syntax

int setCurrentOut(int slotno, int channel, float value)

Description:

This function is used to initialize the I-8024W module in the specified slot for current output. Users must call this function before trying to use the other I-8024W functions for current output.

Parameter:

slotno : [Input] Specify the VP-23(25)A1 system slot

channel : [Input] Output channel (Range: 0 to 3)

value : [Input] Output data with engineering unit (Current Output: 0~20 mA)

Syntax

int setVoltageOut(int slotno, int channel, float value)

Description:

This function is used to send the voltage float value to the I-8024W module with the specified channel and slot in the Linux-based PAC of ICP DAS system.

Parameter:

slotno : [Input] Specified the Linux-based PAC of ICP DAS 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.

1.3.5 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

2. I-7K Modules DIO Control Demo

When using the I-87KW modules for I/O control in the VP-23(25)A1, the program will be a little different depending on the location of I-87KW modules. There are three conditions for the location of I-87KW modules：

(1) When I-87KW DIO modules are **in the VP-23(25)A1 slots**, please refer to the demo in the section 2.1.

(2) When I-87KW DIO modules are **in the I-87KW I/O expansion unit slots**, please refer to the demo in the section 2.2.

2.1 I-7K DI Modules in COM port of VP-23(25)A1

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：

Find the IP address of ViewPAC-25A1, please press Menu → Dev Tools → Terminal Emulator and enter the command “getprop” to get IP address. (refer to Fig.2-1)

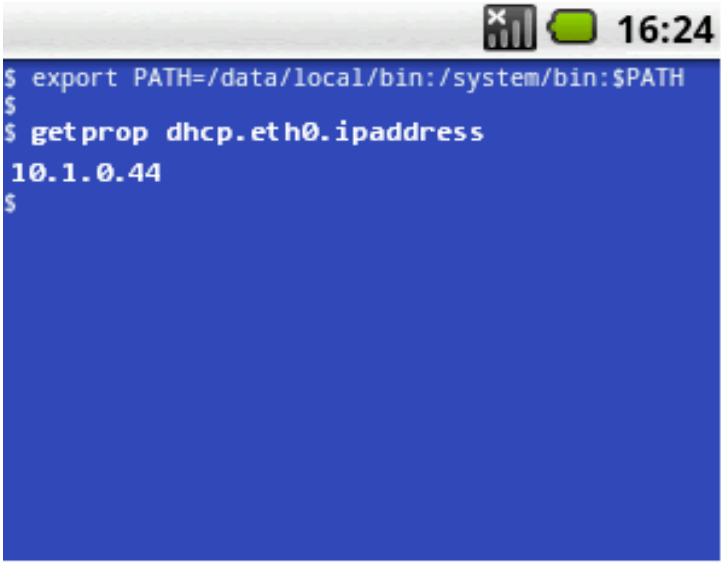

Fig.2-1

Open a “DOS Command Prompt” and into <Android sdk>/platform-tools/ first, and using ADB shell to connect your ViewPAC-25A1 via TCP/IP. Then type **adb push i7kdio.java /data/app** to transfer the i7kdi_demo.apk file to the VP-23(25)A1. (refer to Fig.2-2, 2-3)

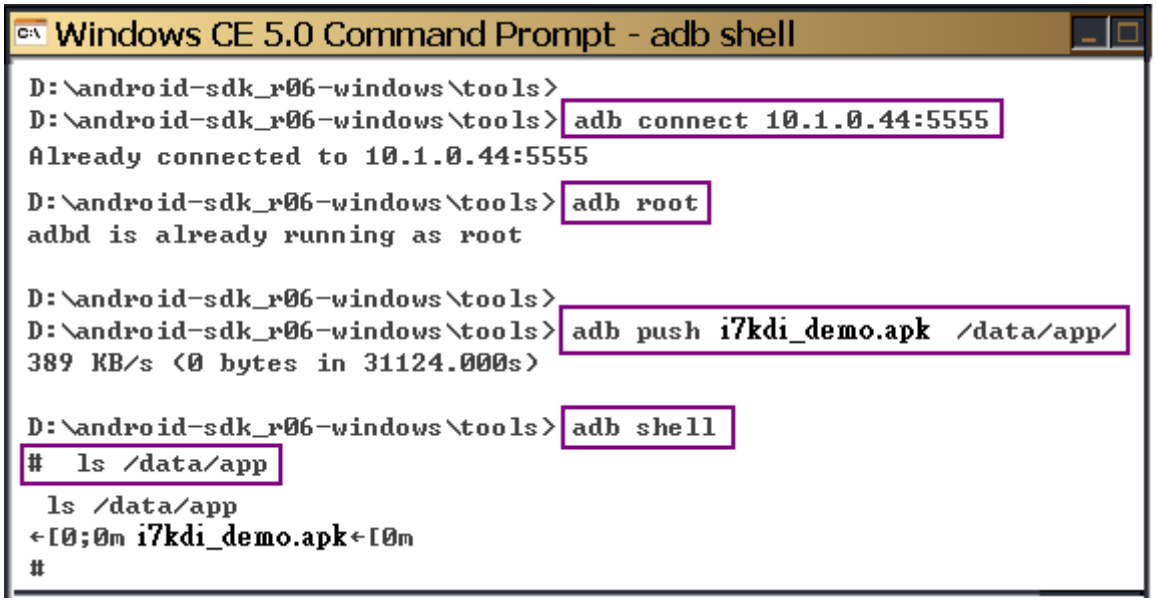
Windows CE 5.0 Command Prompt - adb shell
D:\android-sdk_r06-windows\tools>
D:\android-sdk_r06-windows\tools>adb connect 10.1.0.44:5555
Already connected to 10.1.0.44:5555
D:\android-sdk_r06-windows\tools>adb root
adbd is already running as root
D:\android-sdk_r06-windows\tools>
D:\android-sdk_r06-windows\tools>adb push i7kdi_demo.apk /data/app/
389 KB/s (0 bytes in 31124.000s)
D:\android-sdk_r06-windows\tools>adb shell
ls /data/app
ls /data/app
←[0;0m i7kdi_demo.apk←[0m
#

Fig.2-2

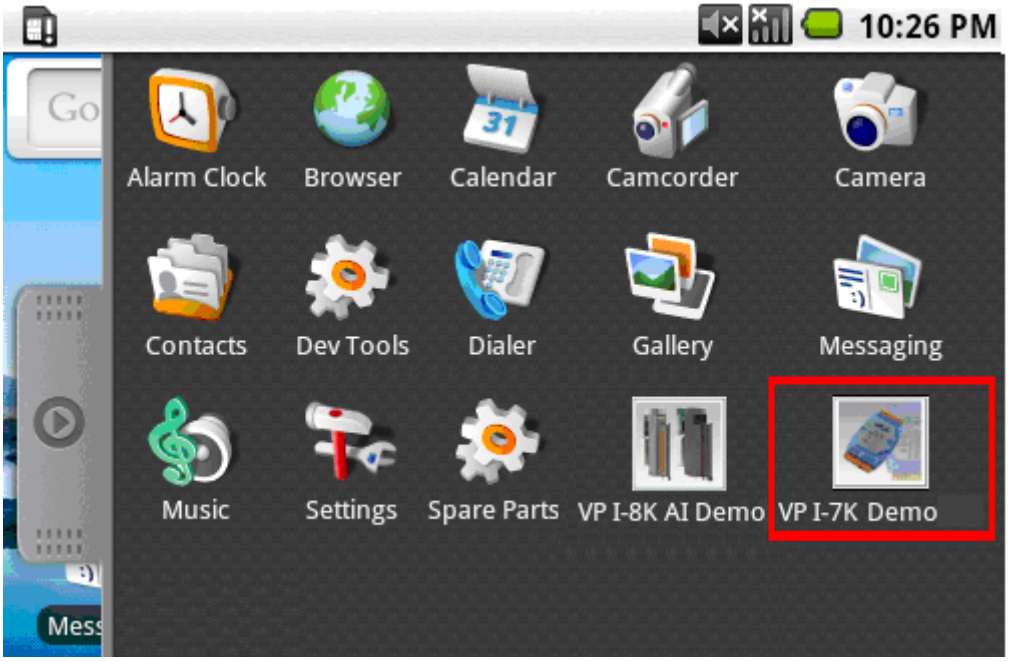
10:26 PM
Alarm Clock
Browser
Calendar
Camcorder
Camera
Contacts
Dev Tools
Dialer
Gallery
Messaging
Music
Settings
Spare Parts
VP I-8K AI Demo
VP I-7K Demo

Fig.2-3

2.2 I-7K DO Modules in COM port of VP-23(25)A1

Fig. 2-4

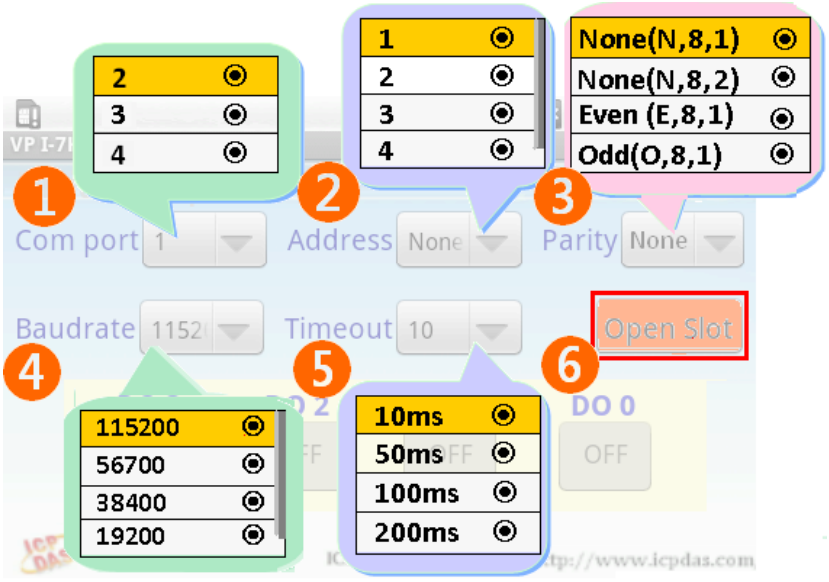

Fig. 2-5

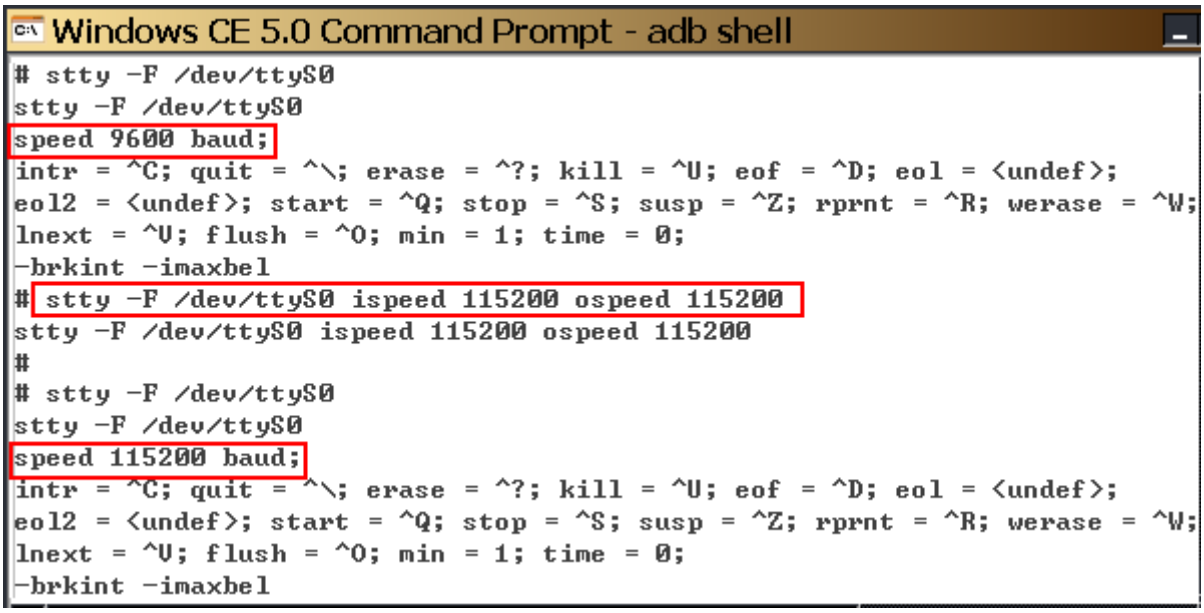
Windows CE 5.0 Command Prompt - adb shell
stty -F /dev/ttyS0
stty -F /dev/ttyS0
speed 9600 baud;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W;
lnext = ^V; flush = ^O; min = 1; time = 0;
-brkint -imaxbel
stty -F /dev/ttyS0 ispeed 115200 ospeed 115200
stty -F /dev/ttyS0 ispeed 115200 ospeed 115200
#
stty -F /dev/ttyS0
stty -F /dev/ttyS0
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. 2-6

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);
        }
}
```

4. I-87KW Modules DIO Control Demo

When using the I-87KW modules for I/O control in the VP-23(25)A1, the program will be a little different depending on the location of I-87KW modules. There are three conditions for the location of I-87KW modules：

(1) When I-87KW DI modules are in the VP-23(25)A1 slots, please refer to the demo in the section 4.1.

(2) When I-87KW DO modules are in the VP-23(25)A1 slots, please refer to the demo in the section 4.2.

(3) When I-87KW DIO modules are in the VP-23(25)A1 slots, please refer to the I/O control of in the section 4.3.

4.1 I-87KW DI Modules in slots of VP-23(25)A1

Fig. 4-1


1
2
3
11:39 PM
Slot No. 1
Channels 4
Open Slot
DI 7 DI 6 DI 5 DI 4 DI 3 DI 2 DI 1 DI 0
DI 15 DI 14 DI 13 DI 8
DI 23 DI 22 DI 21 DI 16
DI 31 DI 30 DI 29 DI 24
OFF
4
8
16
32

Fig. 4-2


11:39 PM
ViewPAC I-87K DI Demo
Digital Input
Slot 1
Channels 4
Open Slot
DI 7 DI 6 DI 5 DI 4 DI 3 DI 2 DI 1 DI 0
DI 15 DI 14 DI 13 DI 12 DI 11 DI 10 DI 9 DI 8
DI 23 DI 22 DI 21 DI 20 DI 19 DI 18 DI 17 DI 16
DI 31 DI 30 DI 29 DI 28 DI 27 DI 26 DI 25 DI 24
OFF

Fig. 4-3


11:30 PM
ViewPAC I-87K DI Demo
Digital Input
Slot 1
Channels 4
Close
DI 7 DI 6 DI 5 DI 4 DI 3 DI 2 DI 1 DI 0
OFF OFF OFF OFF OFF ON OFF ON
DI 15 DI 14 DI 13 DI 12 DI 11 DI 10 DI 9 DI 8
OFF OFF OFF OFF OFF OFF OFF OFF
DI 23 DI 22 DI 21 DI 20 DI 19 DI 18 DI 17 DI 16
OFF OFF OFF OFF OFF OFF OFF OFF
DI 31 DI 30 DI 29 DI 28 DI 27 DI 26 DI 25 DI 24
OFF OFF OFF OFF OFF OFF OFF OFF

Fig. 4-4

4.2 I-87KW DO Modules in slots of VP-23(25)A1

Fig. 4-5

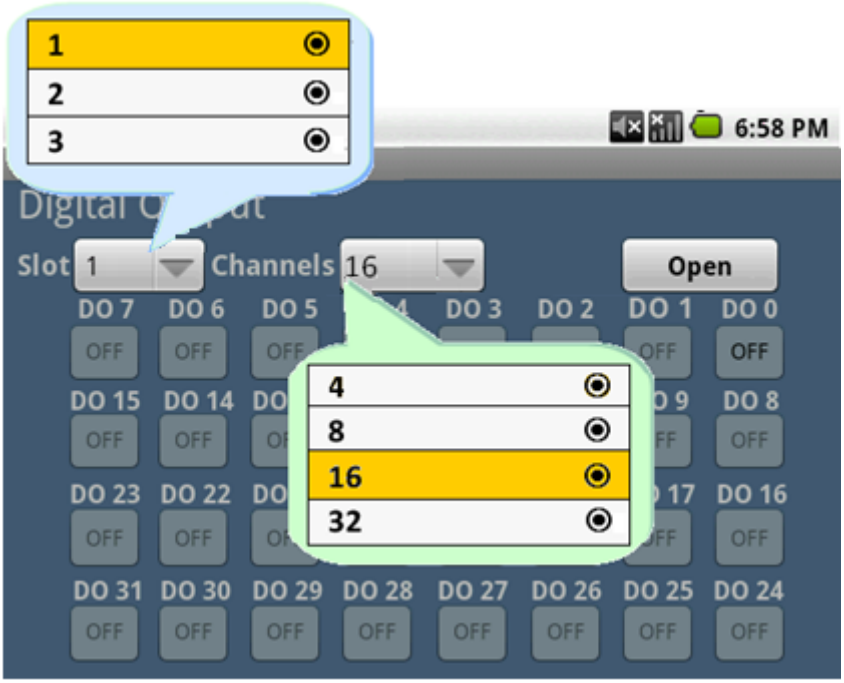

Fig. 4-6


6:58 PM
ViewPAC I-87K DO Demo
Digital Output
Slot 1
Channels 4
Open
DO 7 DO 6 DO 5 DO 4 DO 3 DO 2 DO 1 DO 0
OFF OFF OFF OFF OFF OFF OFF OFF
DO 15 DO 14 DO 13 DO 12 DO 11 DO 10 DO 9 DO 8
OFF OFF OFF OFF OFF OFF OFF OFF
DO 23 DO 22 DO 21 DO 20 DO 19 DO 18 DO 17 DO 16
OFF OFF OFF OFF OFF OFF OFF OFF
DO 31 DO 30 DO 29 DO 28 DO 27 DO 26 DO 25 DO 24
OFF OFF OFF OFF OFF OFF OFF OFF

Fig. 4-7


9:23 PM
ViewPAC I-87K DO Demo
Digital Output
Slot 1
Channels 16
Close
DO 7 DO 6 DO 5 DO 4 DO 3 DO 2 DO 1 DO 0
OFF OFF OFF ON ON ON ON OFF
DO 15 DO 14 DO 13 DO 12 DO 11 DO 10 DO 9 DO 8
ON OFF ON ON OFF OFF ON ON
DO 23 DO 22 DO 21 DO 20 DO 19 DO 18 DO 17 DO 16
OFF OFF OFF OFF OFF OFF OFF OFF
DO 31 DO 30 DO 29 DO 28 DO 27 DO 26 DO 25 DO 24
OFF OFF OFF OFF OFF OFF OFF OFF

Fig. 4-8

4.3 I-87KW DIO Modules in slots of VP-23(25)A1

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

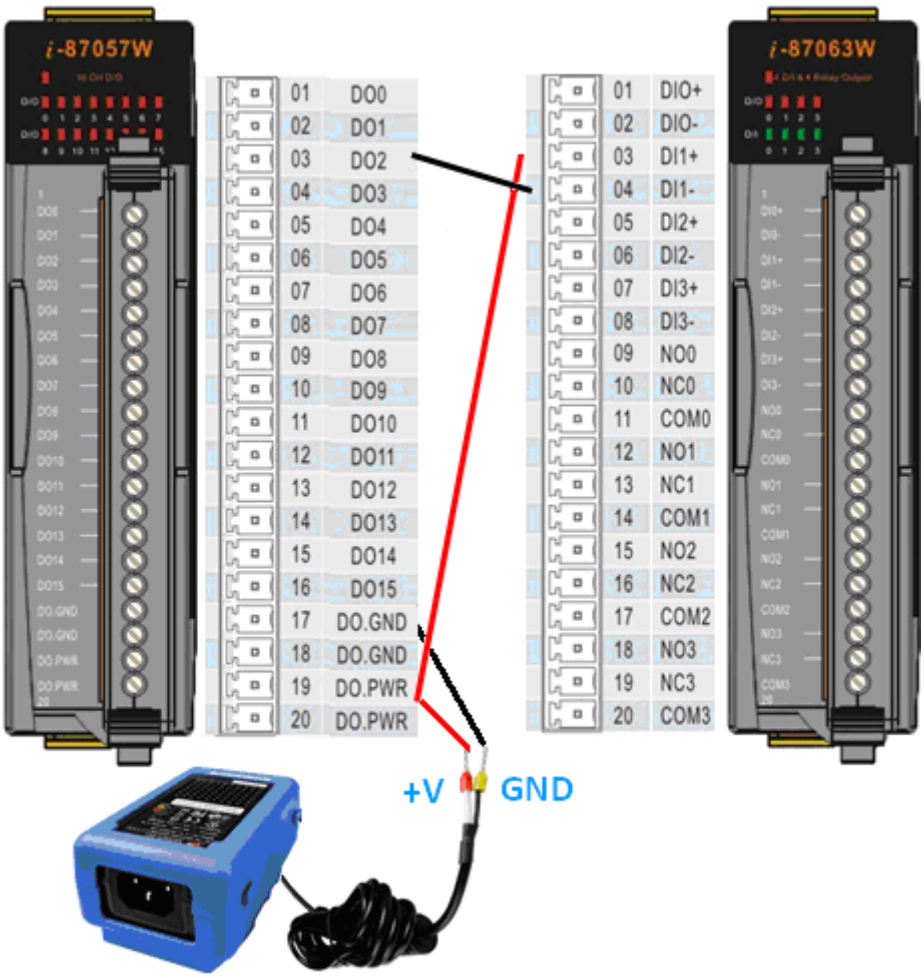

Fig. 4-9

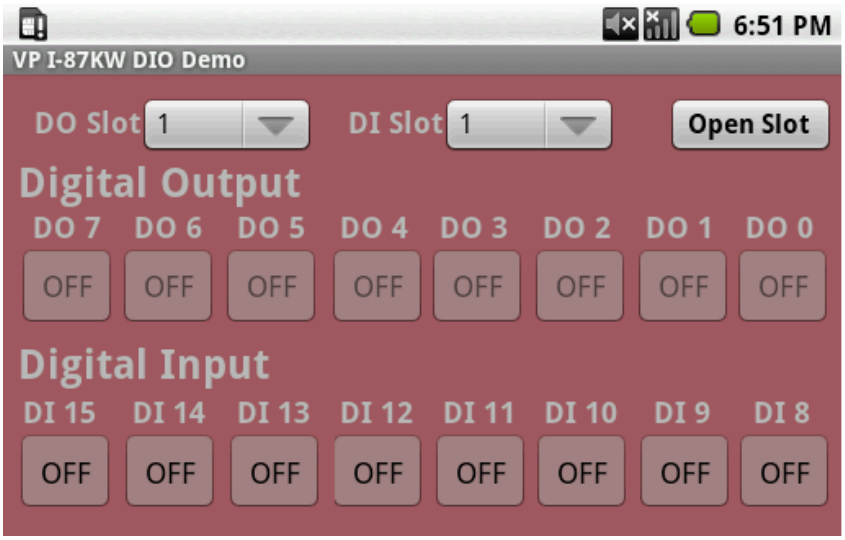

Fig. 4-10

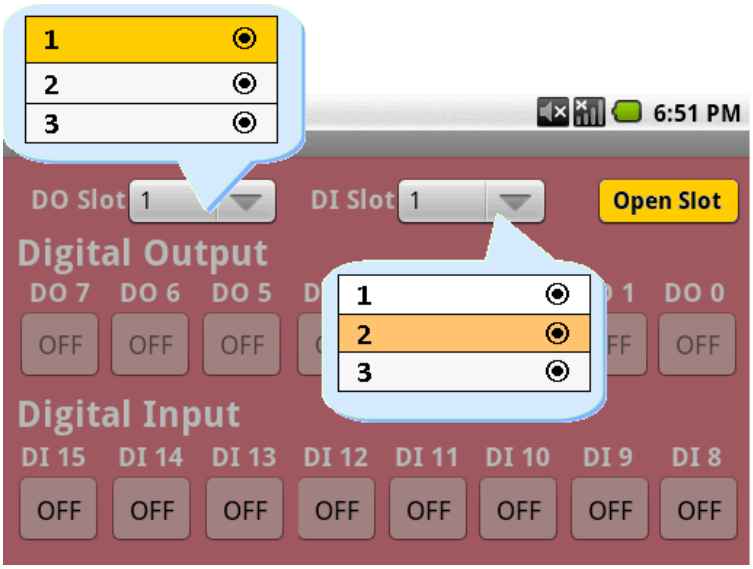

Fig. 4-11

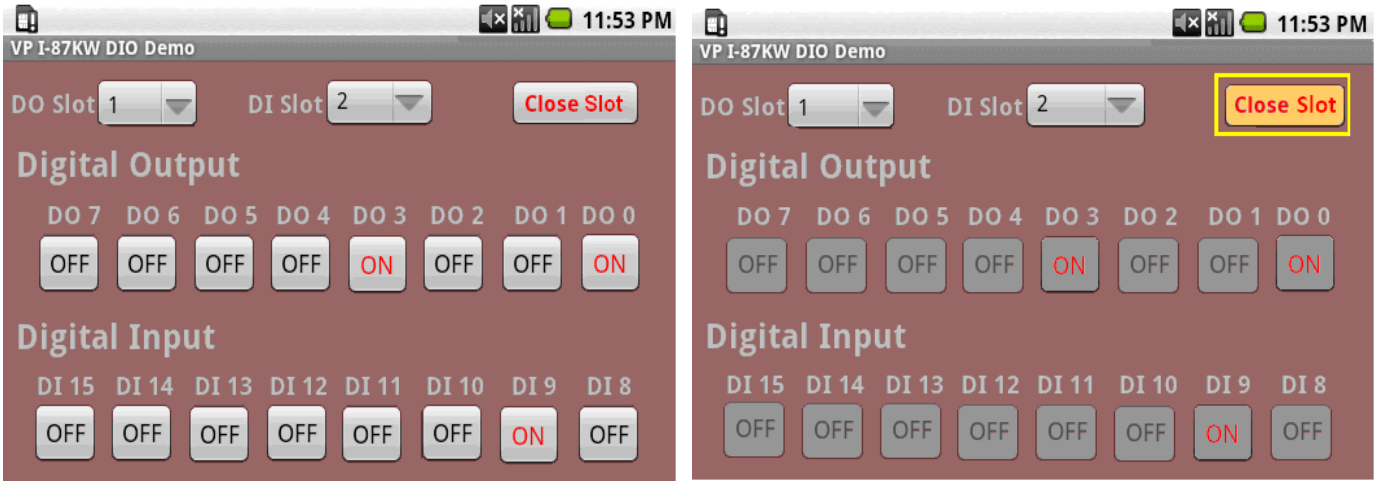

Fig. 4-12

5. I-87KW Modules AIO Control Demo

When using I-87KW modules for I/O control of the VP-23(25)A1, the program will be a little different depending on the location of I-87KW modules. There are three conditions for the location of I-87KW modules：

(1) When I-87KW AI modules are **in the VP-23(25)A1 slots**, please refer to the demo in the section 5.1.

(2) When I-87KW AO modules are **in the VP-23(25)A1 slots**, please refer to the demo in the section 5.2.

(3) When I-87KW AIO modules **in the VP-23(25)A1 slots**, I-87KW modules will be regarded as I-8KWW modules. In this case users can refer to I/O control demo of I-8KW modules in the section 5.2

5.1 I-87KW AI Modules in slots of VP-23(25)A1

Fig. 5-1

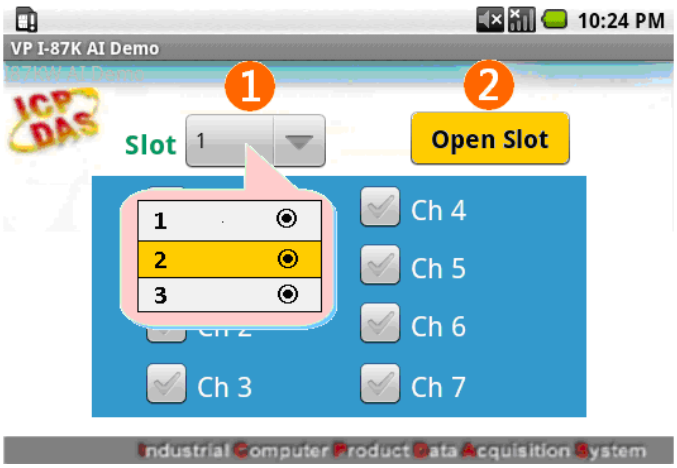

Fig. 5-2

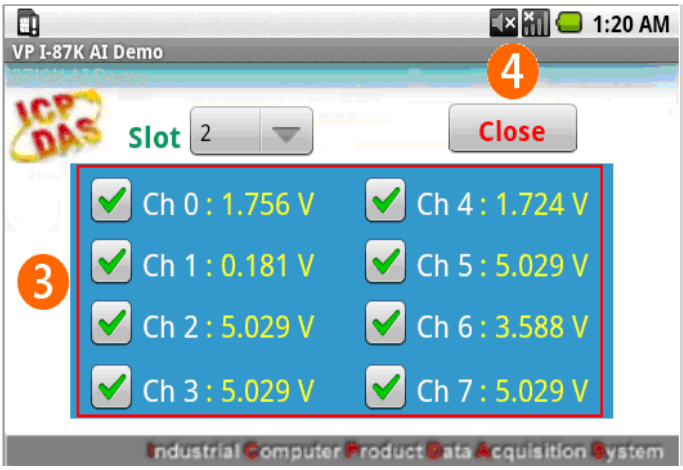

Fig. 5-3

5.2 I-87KW AO Modules in slots of VP-23(25)A1

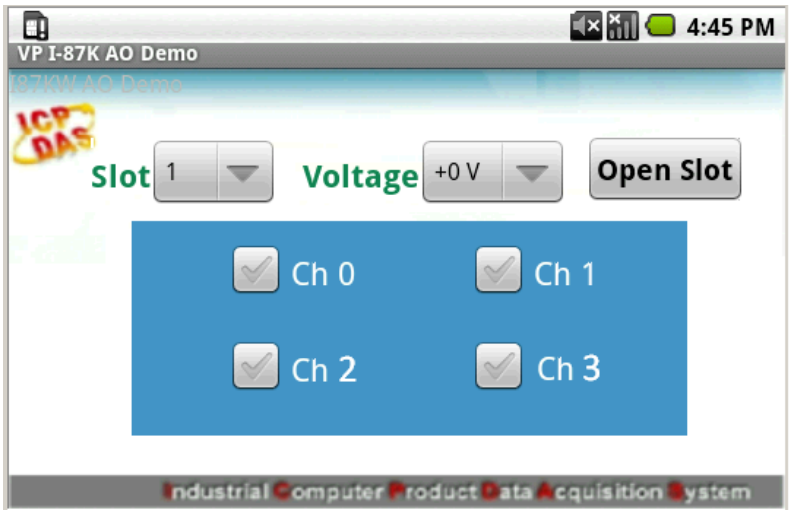

Fig. 5-4

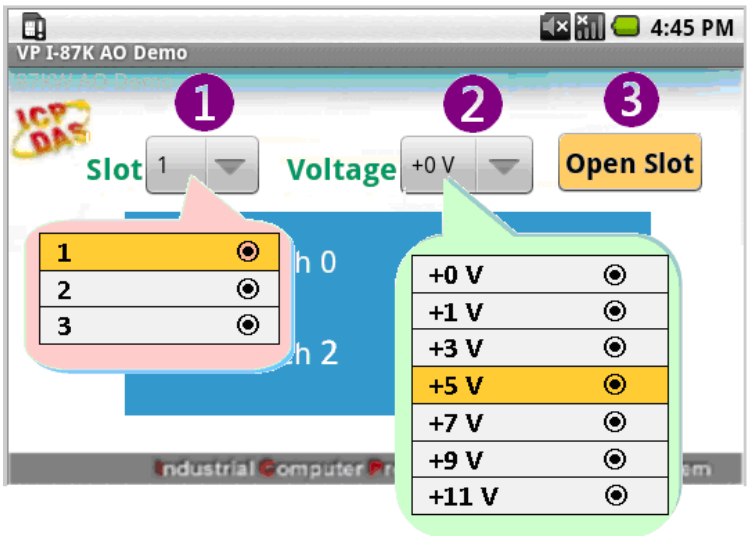

Fig. 5-5

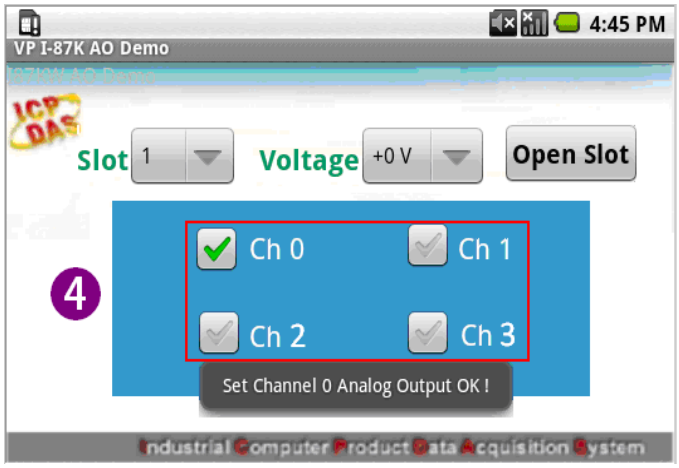

Fig. 5-7

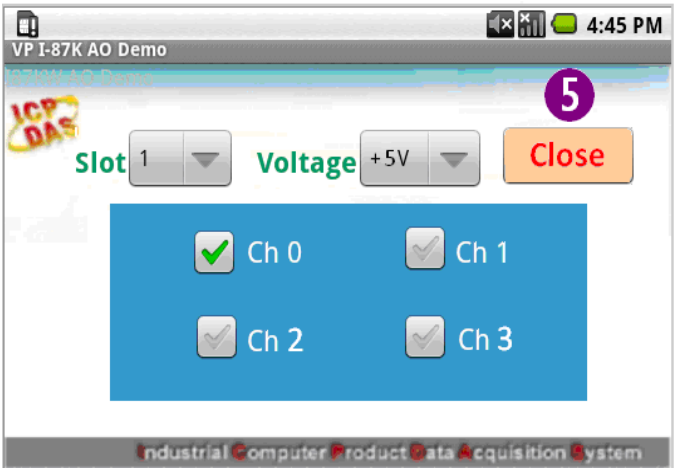

Fig. 5-6

6. I-8KW Modules DIO Control Demo

When using I-8KW modules for I/O control of the VP-23(25)A1, the program will be a little different depending on the location of I-8KW modules. There are three conditions for the location of I-8KW modules：

(1) When I-8KW **DI** modules are in the VP-23(25)A1 slots. Please refer to I/O control demo of I-8KW DI modules in the section 6.1

(2) When I-8KW **DO** modules in the VP-23(25)A1 slots. Please refer to I/O control demo of I-8KW DO modules in the section 6.2

(2) When I-8KW **DIO** modules in the VP-23(25)A1 slots. Please refer to I/O control demo of I-8KW DIO modules in the section 6.3

6.1 I-8KW DI Modules in slots of VP-23(25)A1

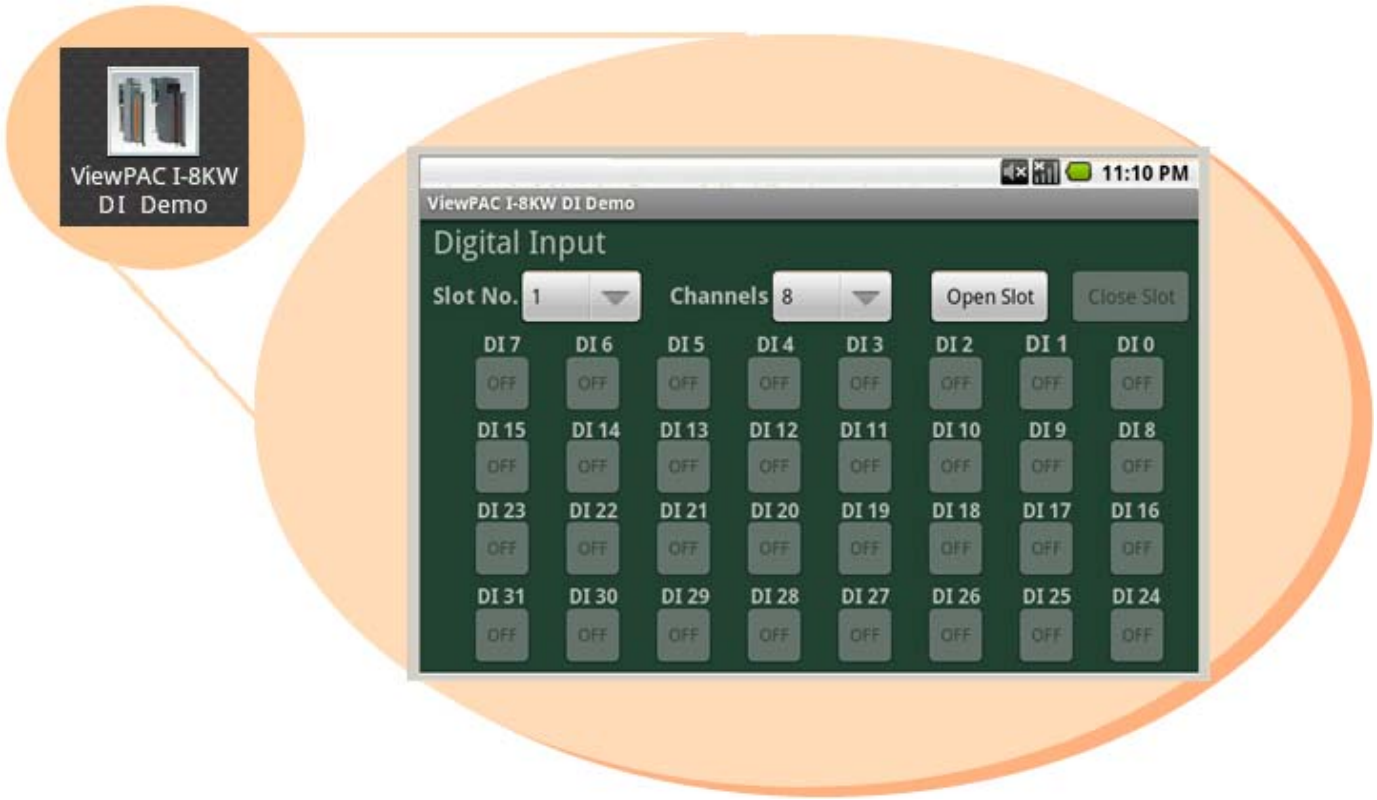

Fig. 6-1

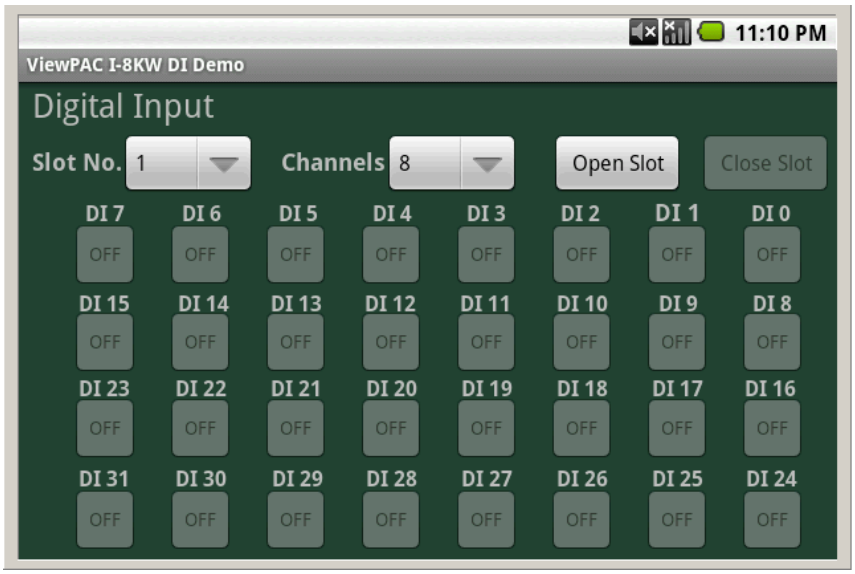

Fig. 6-2

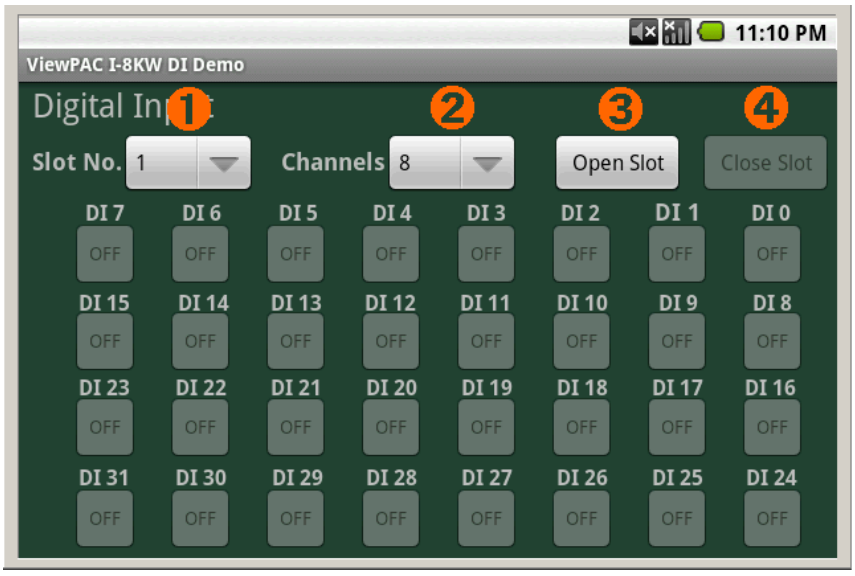

Fig. 6-3

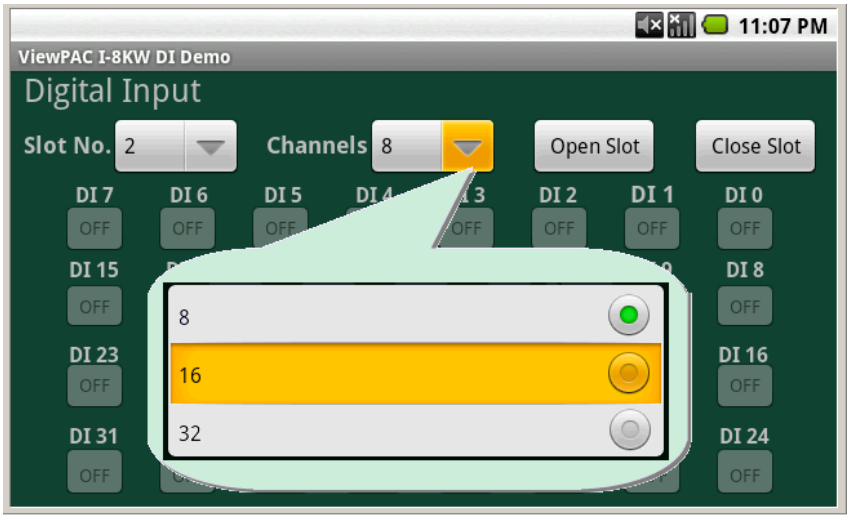

Fig. 6-4

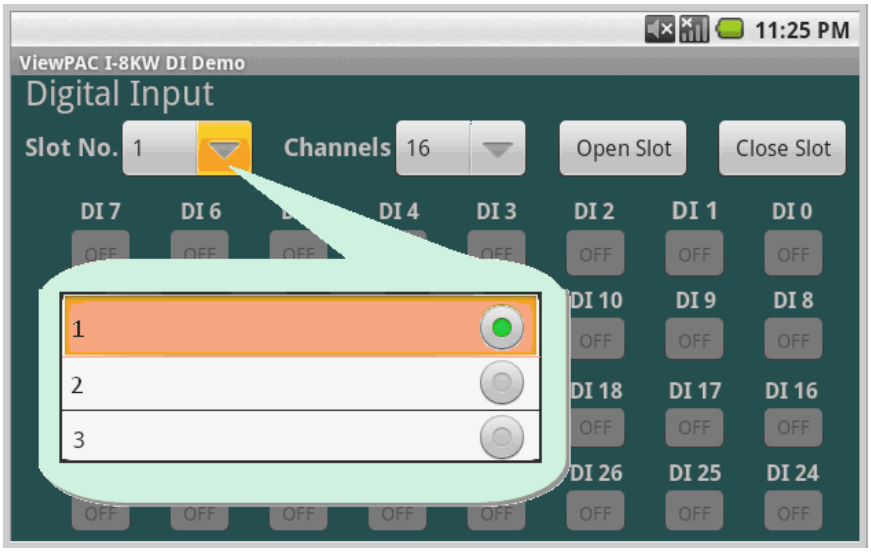

Fig. 6-5

Fig. 6-6

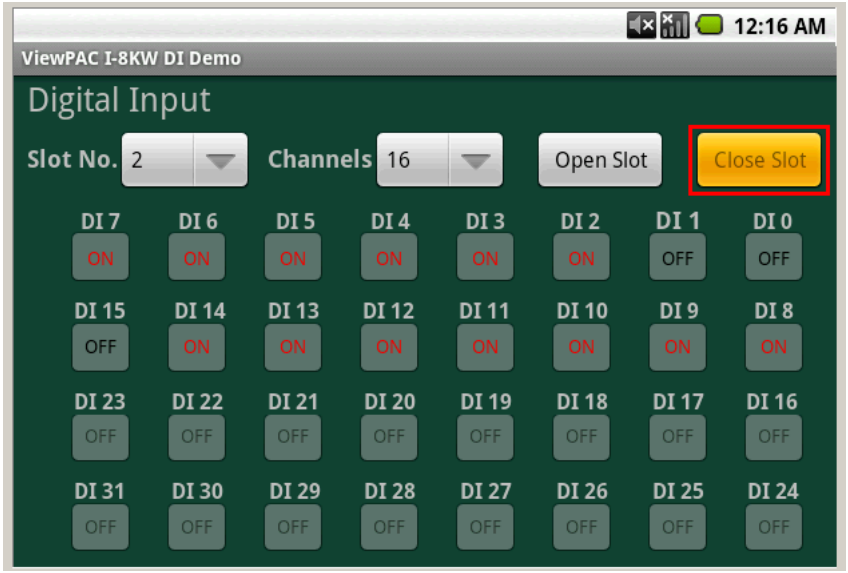

Fig. 6-7

6.2 I-8KW DO Modules in slots of VP-23(25)A1

Fig.6-8

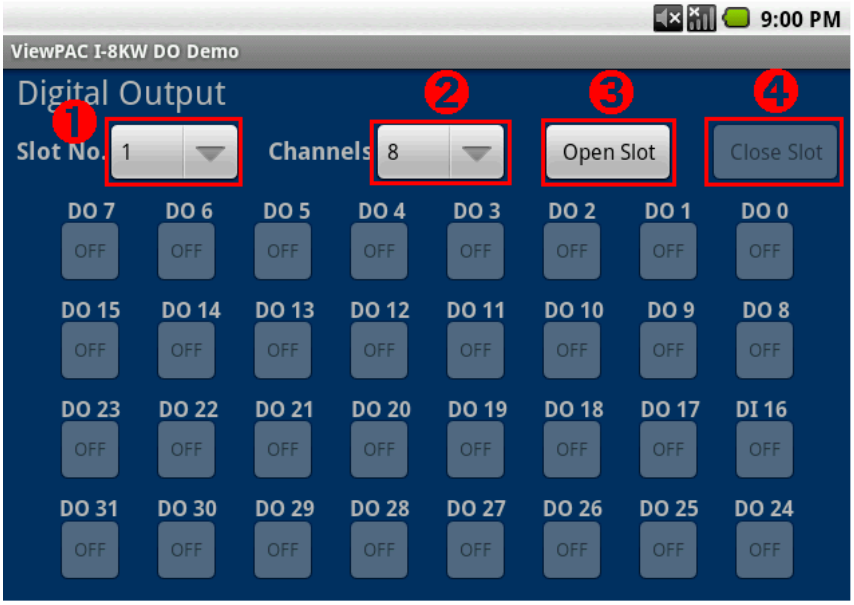

Fig.6-9

Fig.6-10

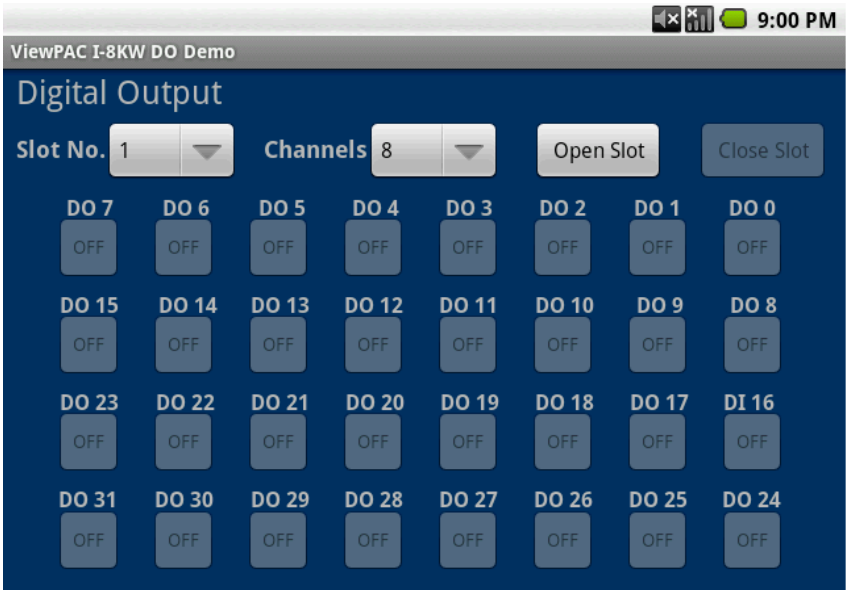

Fig.6-11

6.3 I-8KW DIO Modules in slots of VP-23(25)A1

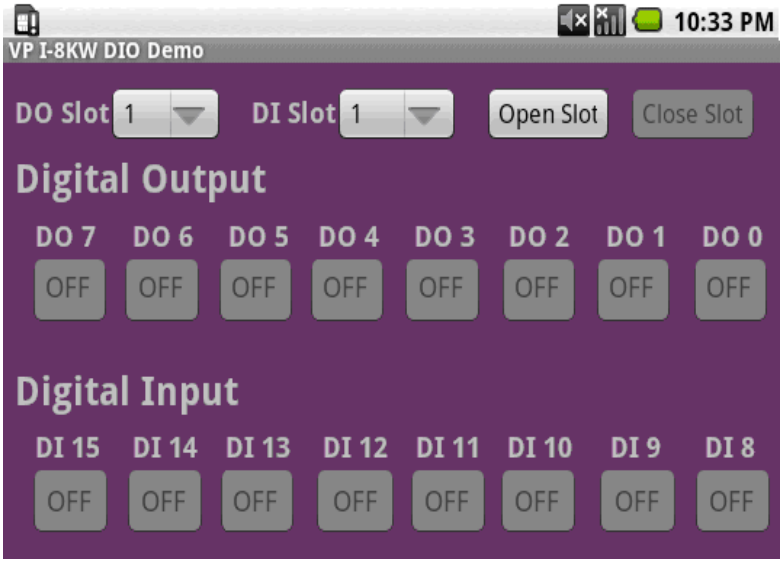

Fig. 6-12

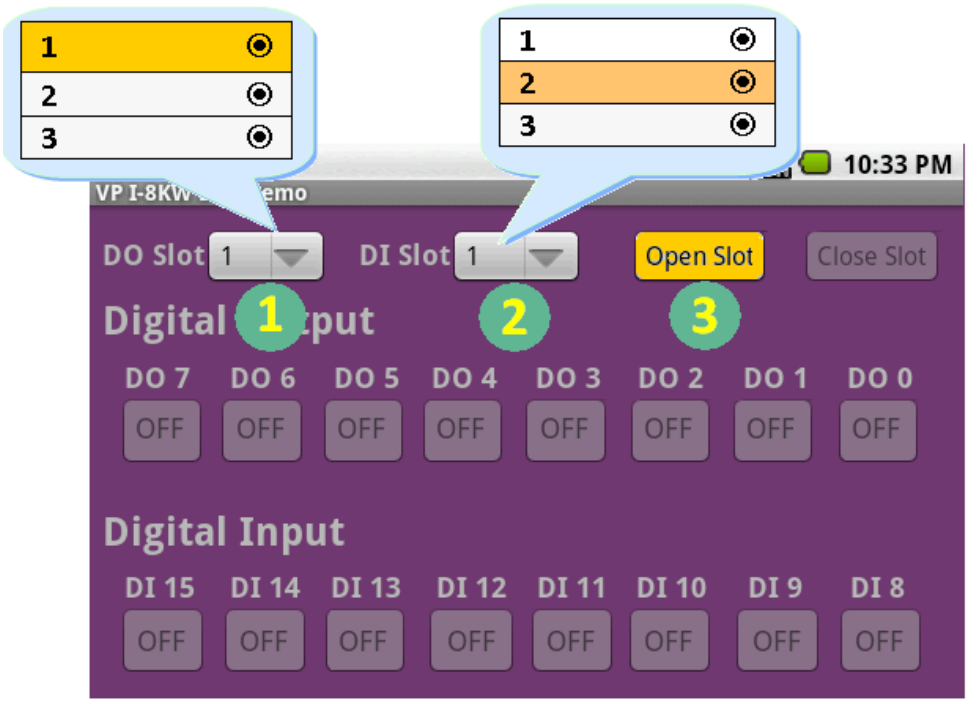

Fig. 6-13

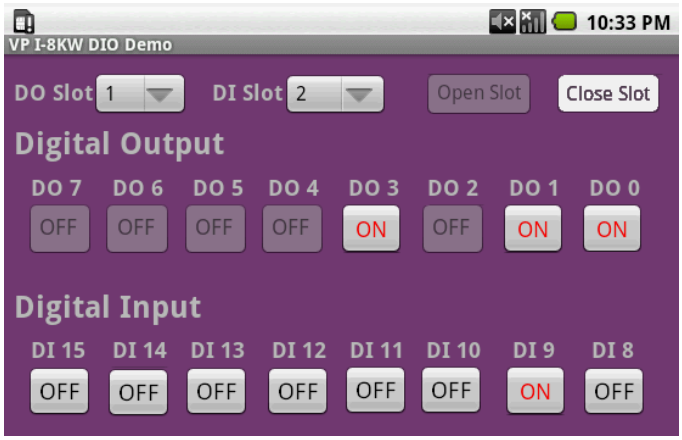

Fig.6-14

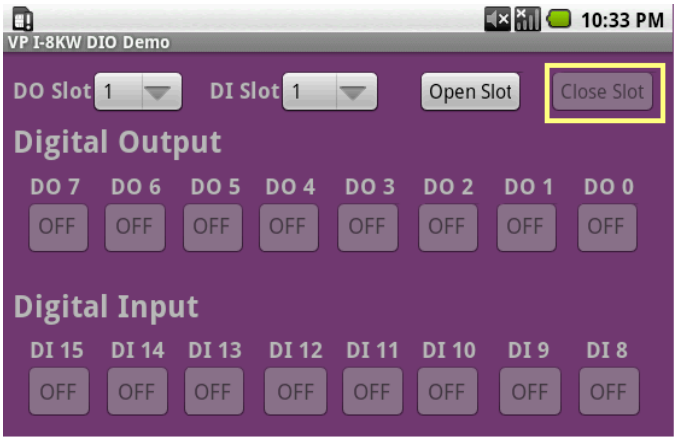

Fig. 6-15

7. I-8KW Modules AIO Control Demo

When using I-8KWW modules for I/O control of the VP-23(25)A1, the program will be a little different depending on the location of I-8KWW modules. There are two conditions for the location of I-8KW modules：

(1) When I-8KWW AIO modules are **in the VP-23(25)A1 slots.** Please refer to I/O control of I-8KW AIO modules in the section 7-1

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

7.1 I-8KW AI Modules in slots of VP-23(25)A1

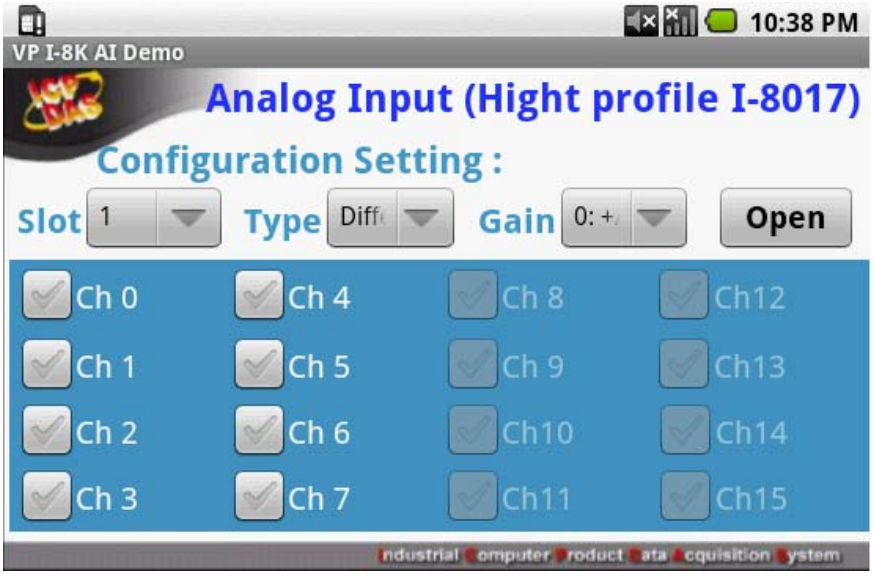

Fig. 7-1

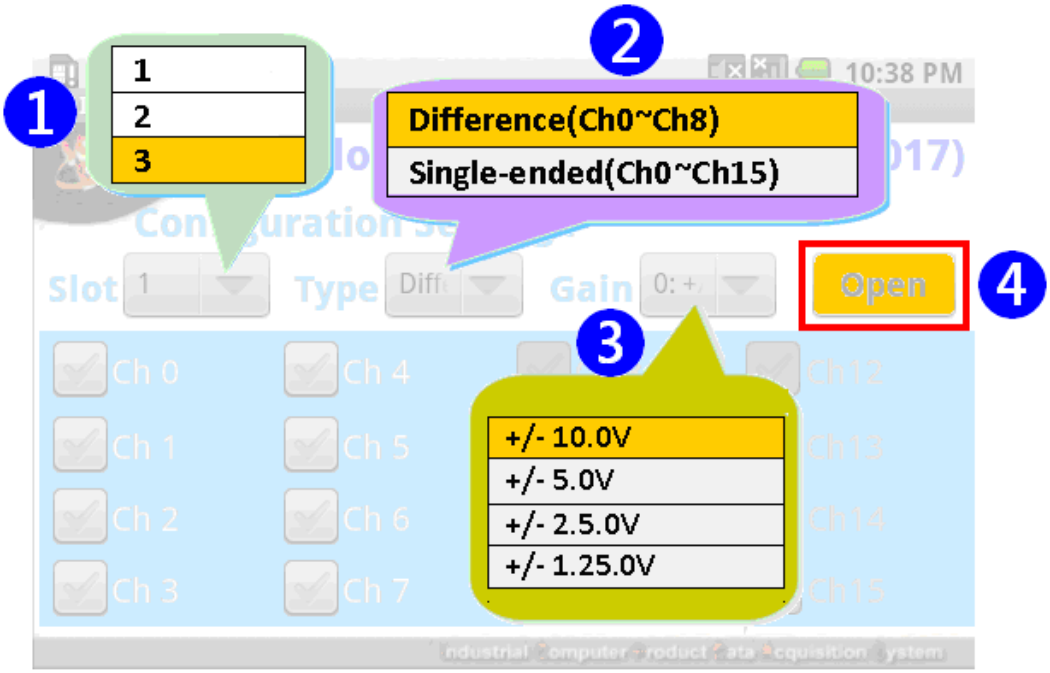
1
2
3
2
Difference(Ch0~Ch8)
Single-ended(Ch0~Ch15)
10:38 PM
Slot
Type
Gain
Open
4
3
+/- 10.0V
+/- 5.0V
+/- 2.5.0V
+/- 1.25.0V
Ch 0
Ch 1
Ch 2
Ch 3
Ch 4
Ch 5
Ch 6
Ch 7

Fig. 7-2


1:41 AM
VP I-8K AI Demo
Analog Input (Hight profile I-8017)
Configuration Setting :
Slot 3
Type
Gain +/-1
Close
Ch0
Ch4 :5.0476074
Ch8
Ch12
Ch1 :5.0476074
Ch5 :5.0463867
Ch9
Ch13
Ch2 :5.050049
Ch6
Ch10
Ch14
Ch3
Ch7 :5.045166
Ch11
Ch15

Fig. 7-3

Fig. 7-4

User can use a third party JAR(ex: icpdas.jar) in your application by adding it to your Android project as follows (refer to Fig.7-5):

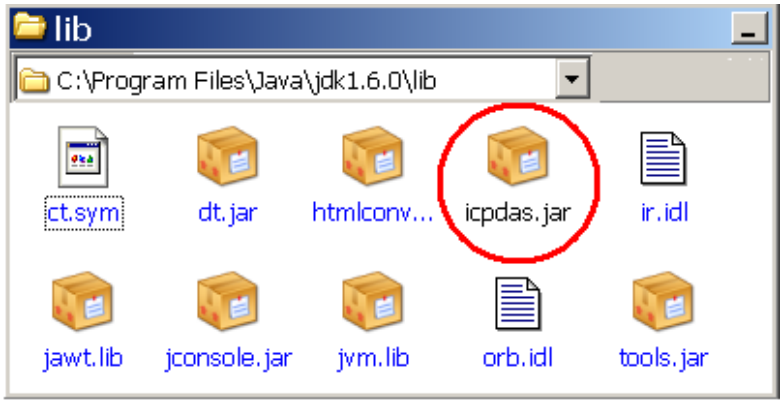

Fig. 7-5

Open a "**ViewPAC-2000 Working Environment**" which is "DOS Command Prompt".

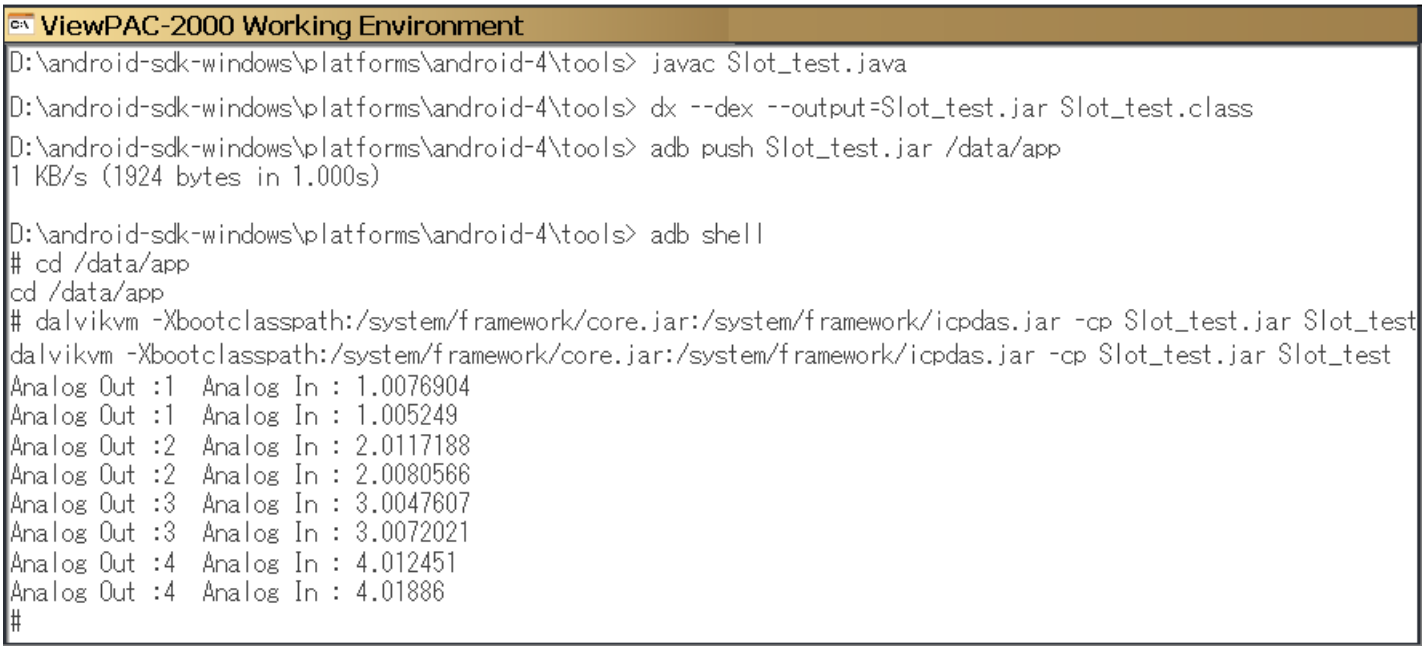

Fig. 7-6

7.2 I-8KW AO Modules in slots of VP-23(25)A1

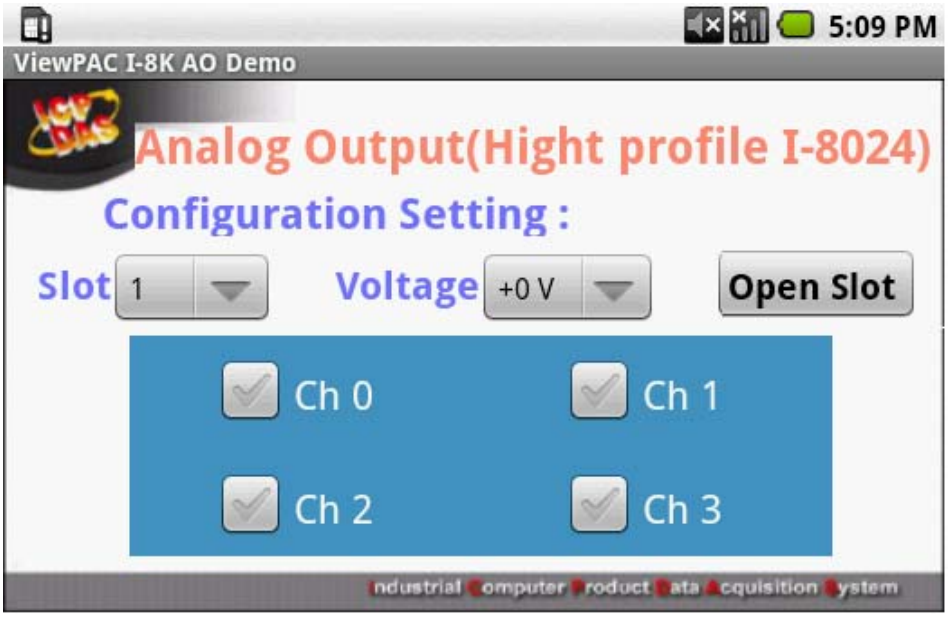

Fig.7-7

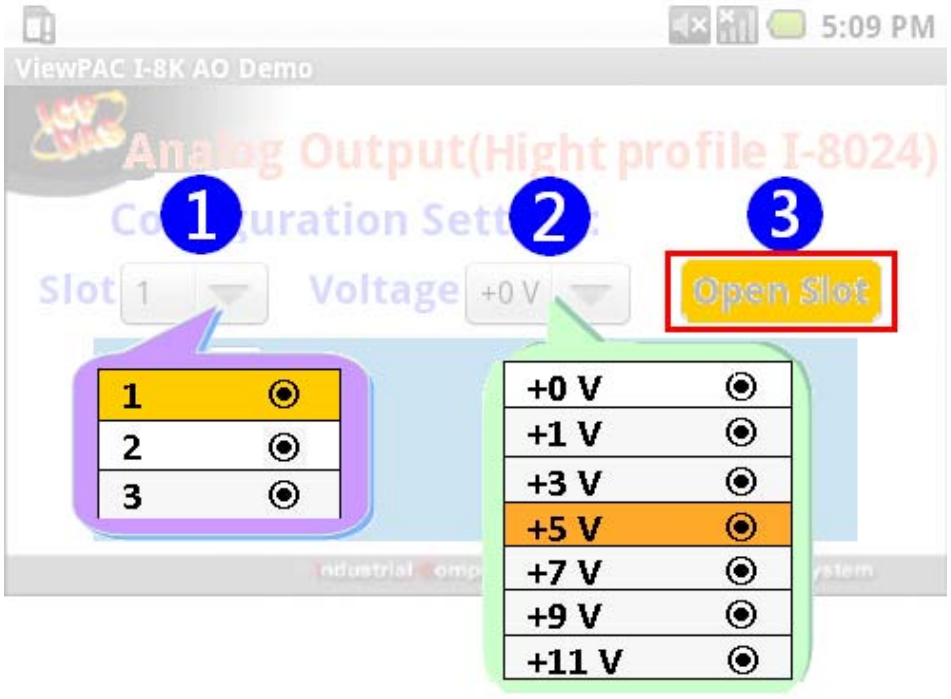

Fig.7-8

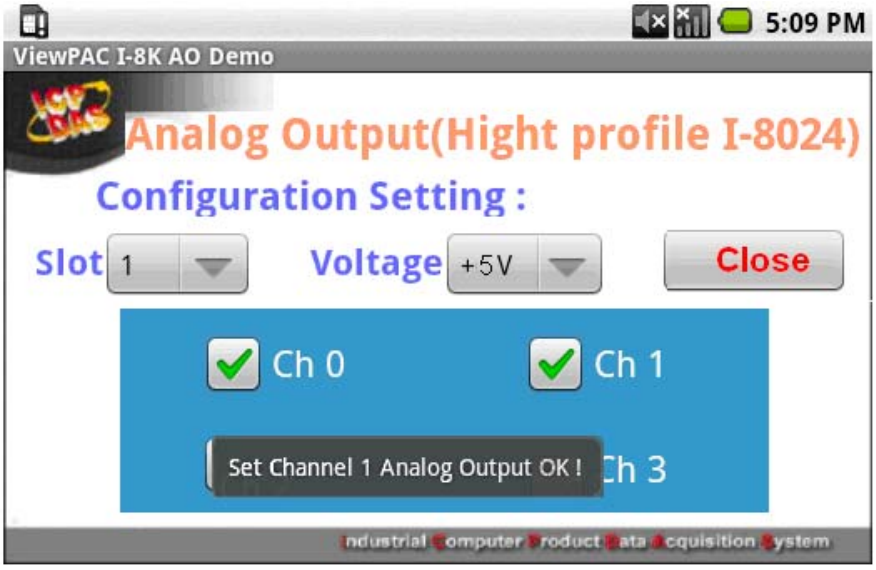

Fig.7-9

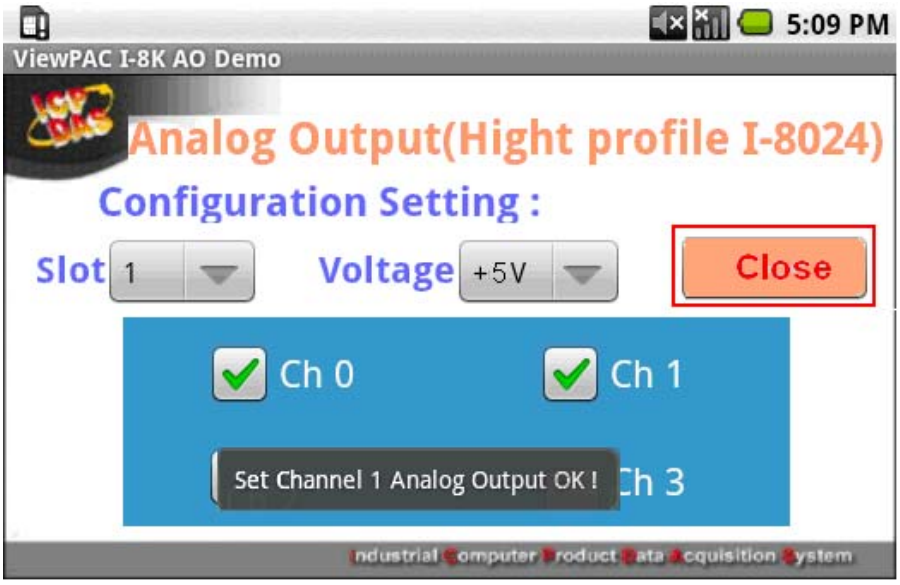

Fig.7-10