Prolin SDK Tutorial

V 2.8.0

PAX Computer Technology（Shenzhen）Co., Ltd.

{ This page intentionally left blank }

Revision History

Date	Version	Note	Author
2012.8.1	V1.0	1. SDK V1.0 release 2. First release.	Lu Xue
2012.8.31	V2.0	1. SDK V2.0 release 2. Add a lot of details for this document.	Lu Xue
2012.9.18	V2.1	1. SDK V2.1 release 2. Add instructions according to users' response for this document.	Lu Xue
2012.10.9	V2.2	1. SDK V2.2 release 2. Adjust the frame size of register and update interface. 3. Auto-scan dynamic libraries and add them to the link options. No longer relying on **PaxPayPro** software when check license.	Lu Xue
2012.10.24	V2.3	1. SDK V2.3 release 2. Way to download SDK is changed from FTP downloading to sending email to "support team". 3. Add a lot of details.	Lu Xue
2012.11.13	V2.4	1. SDK V2.4 release 2. Now support GDB debugging on the physical machine. 3. The app template does not use **minigui** library any more, but **exsoal** library.	Lu Xue
2012.11.28	V2.5	1. SDK v2.5 release 2. Now support both TCP and	Lu Xue

		serial port debugging on the physical machine. 3. Show current version and expired date in the eclipse title area. 4. Auto-check upgrade info and prompt users to do update operation when SDK is opened. 5. Auto-add standard C library path for the newly-created project.	
2012.12.12	V2.6	1. SDK V2.6 release 2. Update emulator to v1.5.	Lu Xue
2013.8.12	V2.7	1. SDK V2.7 release 2. New version of OSAL library. 3. Update to Eclipse Juno S2. 4. Increase SDK start-up speed. 5. Add “Show Line Number” chapter for this doc	Lu Xue
2013.11.6	V2.8.	1. SDK V2.7.06 release 2. Add XCB module. 3. Add XUI module. 4. SDK support displaying Logcat. 5. Support Crashreport.	Lu Xue

Contents

Figure List

1 Introduction

Prolin SDK is a software development kit customized for the Prolin pos machine, such as S300, S800, and S900. You can use it to develop and build pos application. It provides various functions such as code editing, debugging, building and emulating.

Users who have problems with the SDK are welcome to send an email to ProLin@paxsz.com.

This tutorial was originally written by the internal staff of PAX Company. It is still maintained by Lux (lux@paxsz.com).

1.1 Purpose

This manual provides instructions of how to use Prolin SDK for application developers.

1.2 Features

☑Base on Eclipse IDE platform and CDT plug-in. It is more convenient if you are familiar with Eclipse

☑Run on both Windows and Linux computers

☑Support C/C++ development, contains gcc/g++ compile environment

☑Identities registering and checking system based on C/S architecture; use RC4, RSA encryption algorithm

☑Integrated environment to develop and build application

☑You can quickly create an APP/LIB project via a step-by-step instruction

☑Colorful interfaces to show project architecture and source code. Easy to edit, debug and search for help

☑True-to-original emulator in which you can install you APP to instantly check the running result

☑Strong assisted back-stage management including detailed documents, online update and feedback

1.3 OS Supported

Both windows and Linux (Ubuntu) OS are supported.

OS	Version	32bit	64bit
Windows	Xp/Win7/Win8	✓	✓
Ubuntu	10.04 or higher	✓	✓

1.4 Environment Requirements

JDK (java development kit) or JRE (Java runtime environment) is required to be installed in your personal computer. Check if JRE exists in your PC by typing **java –version** command (P1-1).

Please note the SDK (specially the Eclipse) is only a 32bit version. So you should download 32bit JDK/JRE though your system is 64bit.

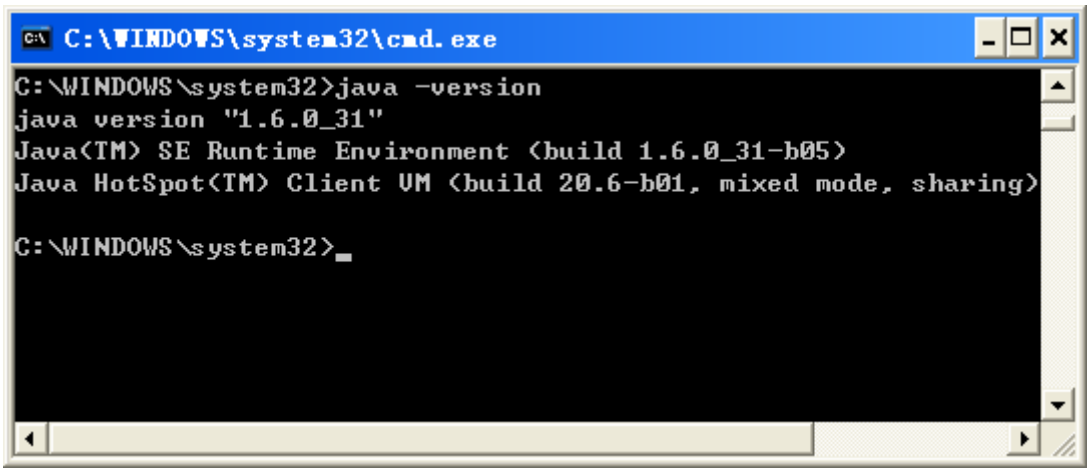

Figure 1.1 Windows console command

JDK/JRE downloading address:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

1.5 User Requirements

You should be familiar with C/C++ programming. And it would be better if you have experience in using the Linux OS and the Eclipse tool.

1.6 Abbreviation

Table 1 Abbreviation

Abbreviation	Description
ADB	Android Debug Bridge
APP	Application
CDMA	Code Division Multiple Access
CDT	C/C++ Development Tooling
COM	Cluster Communication Port
DOC	Document
FAQ	Frequently Asked Question
FTP	File Transfer Protocol
GCC	GNU Compiler Collection
GDB	GNU Project Debugger
GPRS	General Packet Radio Service
LCD	Liquid Crystal Display
IC Card	Integrated Circuit Card
IDE	Integrated Development Environment

JRE	Java Runtime Environment
LIB	Library
OS	Operating System
PC	Personal Computer
PED	PIN Entry Device
PIN	Personal Identification Number
POS	Point Of Sale
PVD	Prolin Virtual Device
RF Card	Radio Frequency Card
RTM	Release To Manufacturing
SDK	Software Development Kit
TCP	Transmission Control Protocol
TM	Terminal Manager
UDP	User Datagram Protocol
UI	User Interface
WAN	Wide Area Network
XCB	Xos Communication Bridge

2 Download and Install

2.1 Download SDK

To download SDK, please contact PAX support team or send an email to support@paxsz.com.

2.2 Install SDK

After downloading, extract the file **prolin_sdk_win-xxx.zip** or **prolin_sdk_linux-xxx.tar.gz** and directly use it. No additional installation or registration steps required.

It is not recommended that the absolute path of your SDK contains blank space or Chinese characters. It will cause unexpected error.

2.3 SDK Preview

Assume you have extracted the SDK to the folder **C:\prolin_sdk_win**. You will see the following files.

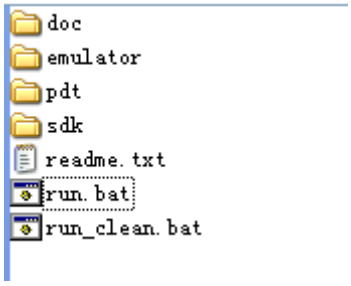

Figure 2.1 SDK files 1

The following is the brief directory tree.

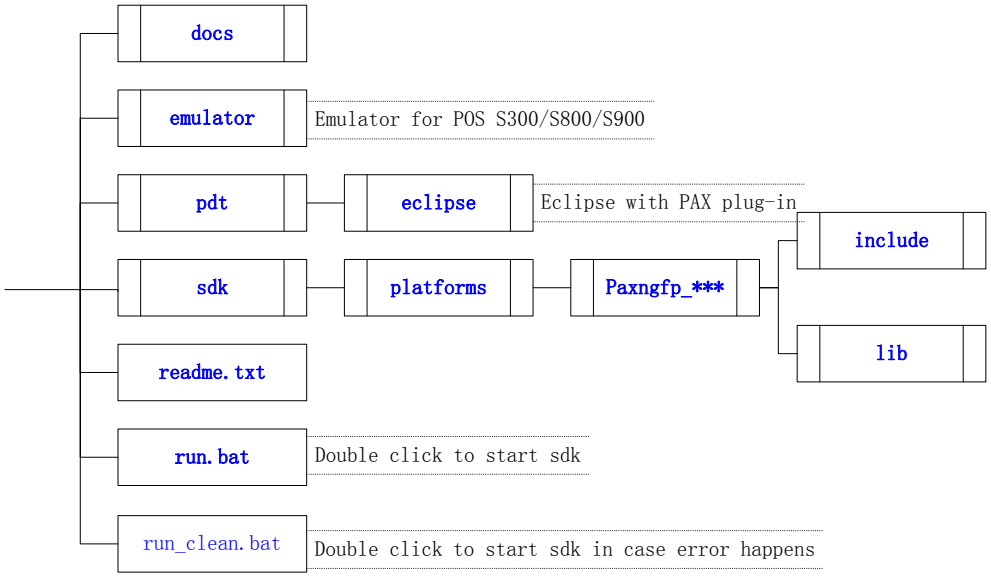

Figure 2.2 SDK file structure 1

3 Open and Register SDK

3.1 Activate SDK

Go to the root directory of the SDK and double click on **run.bat** or **run.exe** to activate (Figure 3.1).

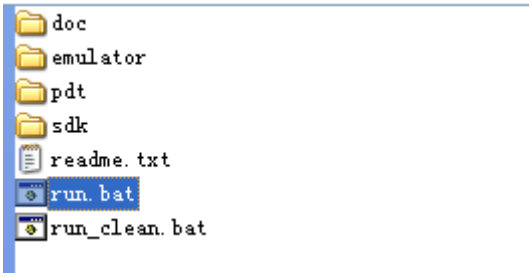

Figure 3.1 SDK files

3.2 Eclipse Launch

Wait for the eclipse to launch.

Figure 3.2 Eclipse launching

3.3 Set Workspace

Set folder of the workspace as you wish (Figure 3.3).

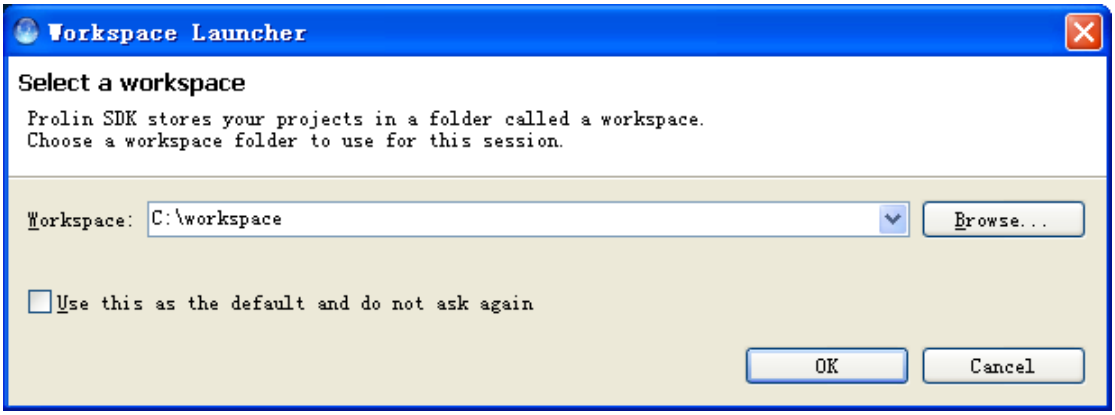

Figure 3.3 Set workspace

Press **OK** button and wait for the SDK to start up.

3.4 Register SDK

If you have not registered, register dialog will auto popup after eclipse start-ups (Figure 3.4).

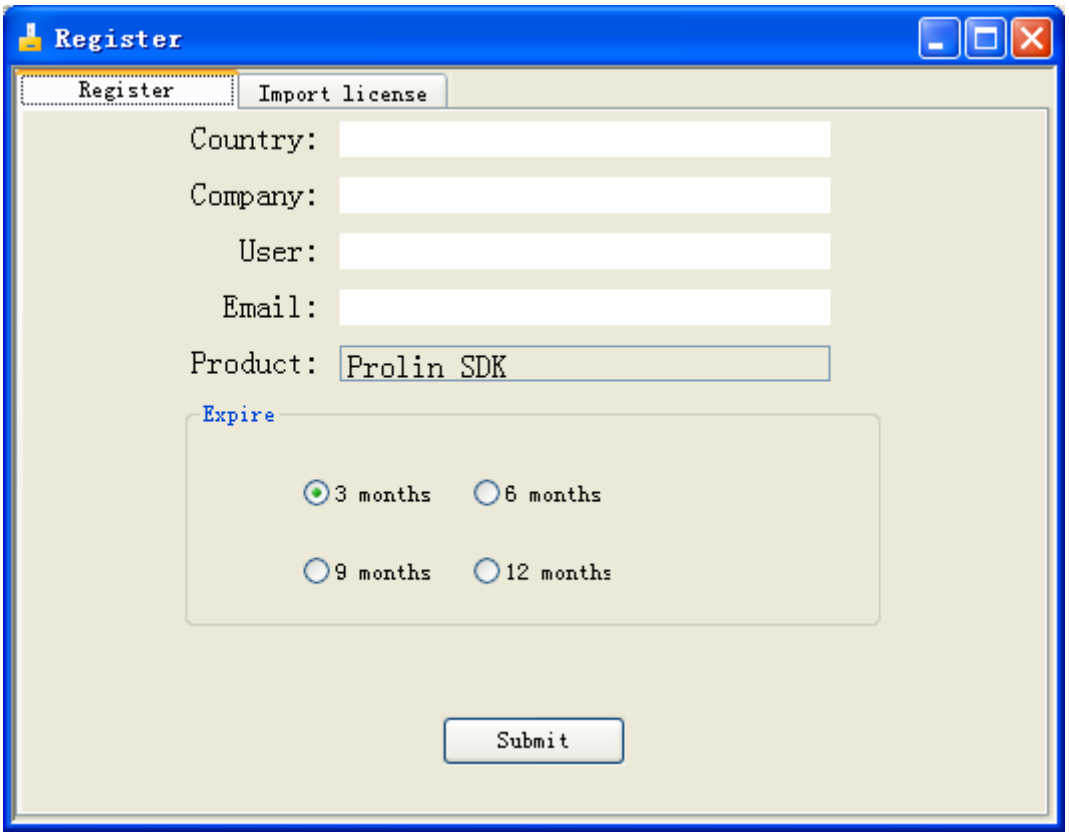

Figure 3.4 Register Dialog

Fill in Register information (P3-5). Click on Button **Submit** (P3-6). This action will send your request to the server.

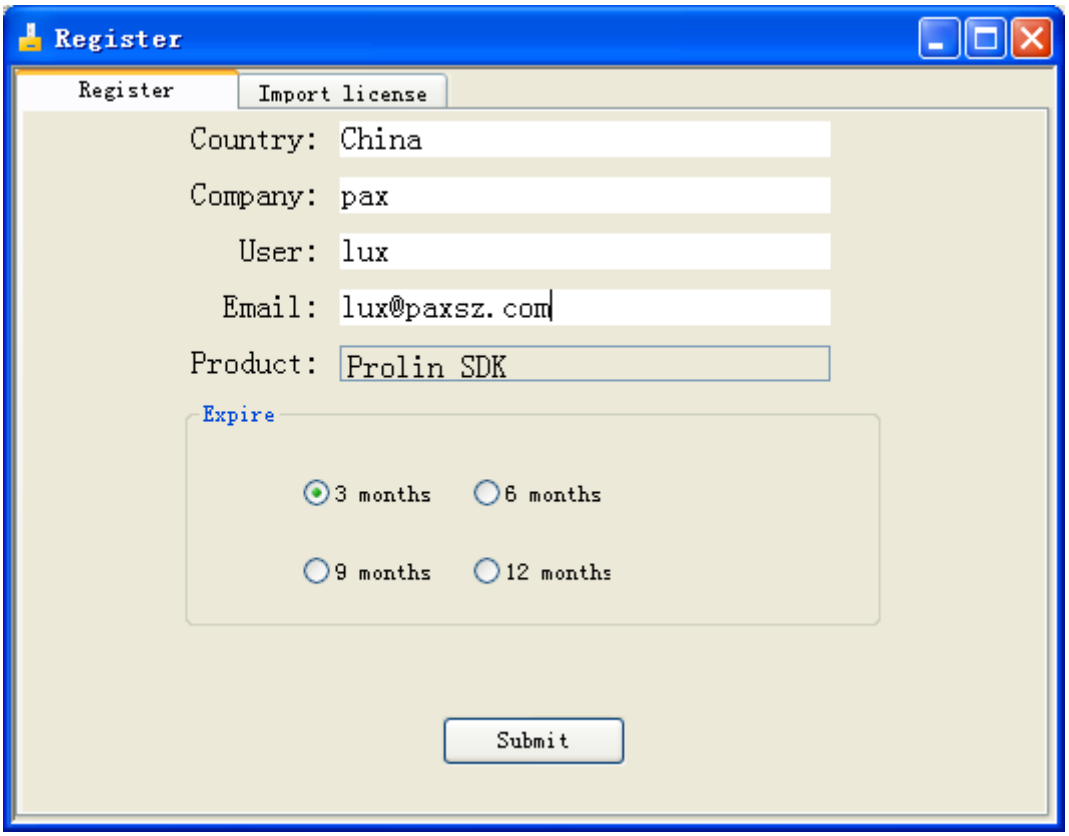

Figure 3.5 Register information

CAUTION

There is a bug in the SDK v2.6 or earlier. If your registering information contains blank space, you will always get an error message "Send request failed".

NOTE

Input effective e-mail address from which you can receive the license.

NOTE

Keep smooth network when registering.

Now close eclipse and wait. Within 24 hours, you will receive an email attached with a license key named "license_sig.key".

Download the license key to your local machine, and then import the license to complete register.

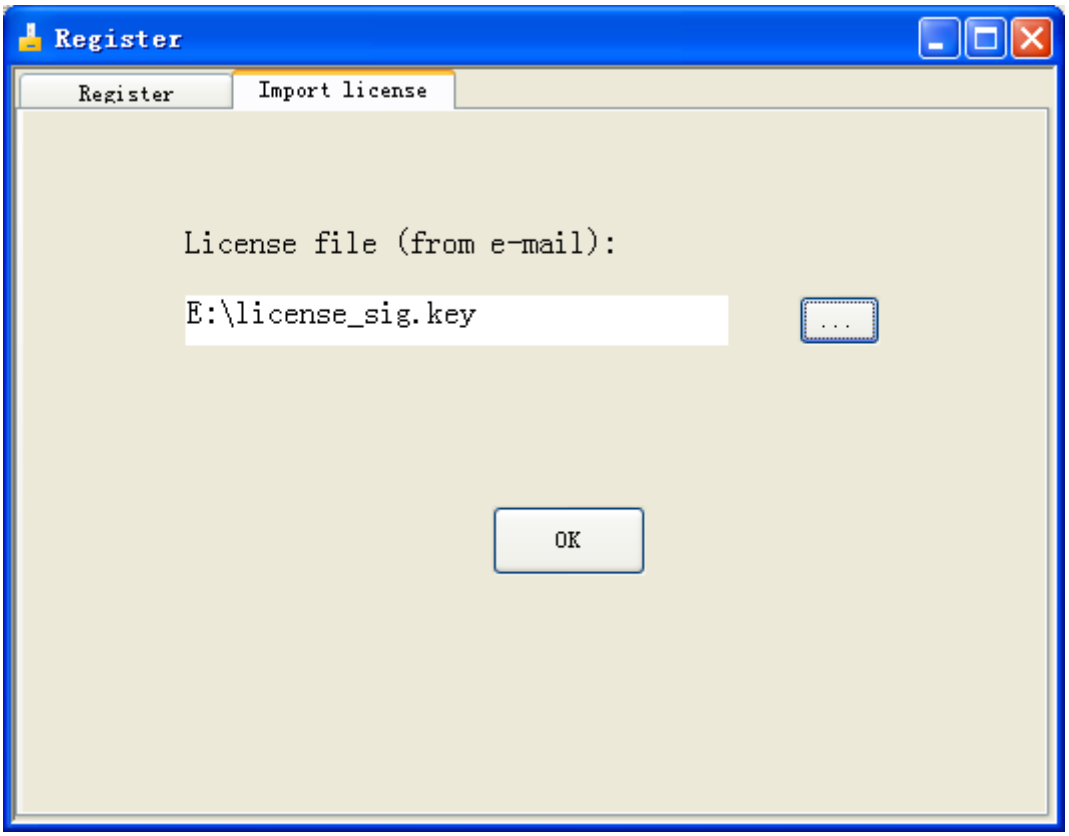

Figure 3.6 Import license

{ This page intentionally left blank }

4 Eclipse Preview and Setting

4.1 Eclipse Main Interface

As integrated development software, Eclipse is customized to support **Prolin** APP/LIB development. The following is the main interface of customized eclipse (Figure 4.1).

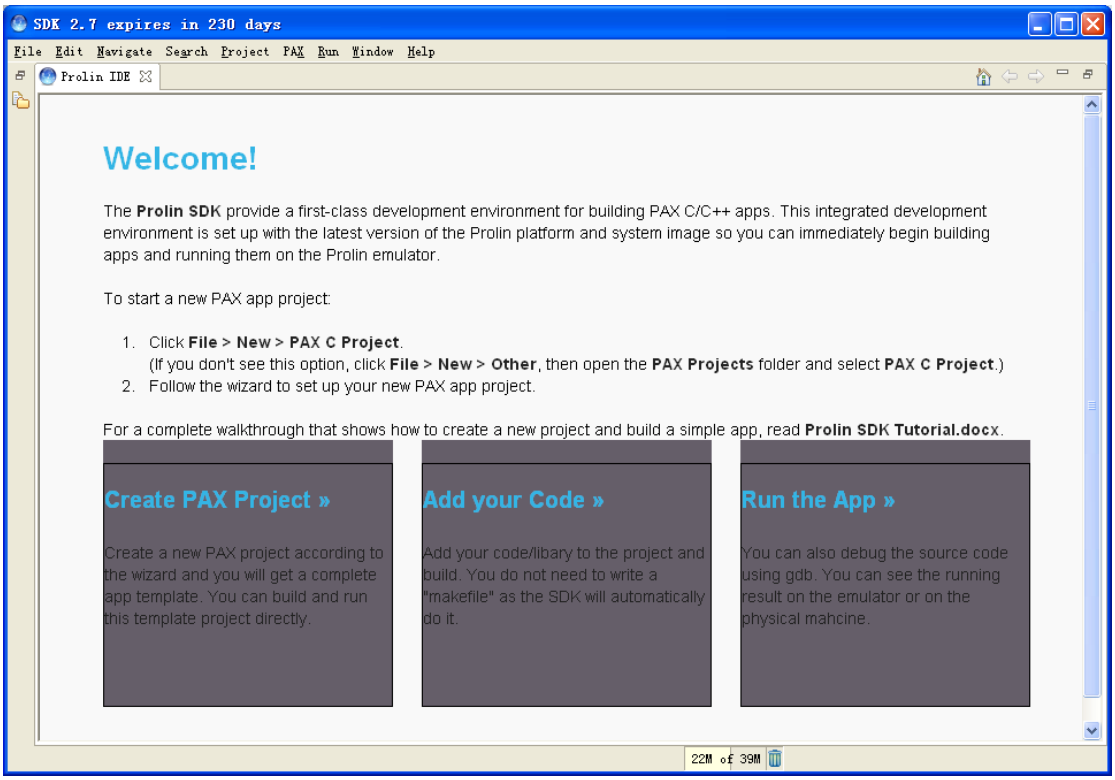

Figure 4.1 Eclipse main interface

4.1.1 Main Menu of PAX

Under the menu **PAX**, there are 4 sub menus: **Generate Package**, **Install Package**, **Emulator Manager**, and **Help**. And Under the **Help** sub menu, there are 4 children: **About SDK**, **License Info**, **Feedback**, and **Update SDK** (Figure 4.2).

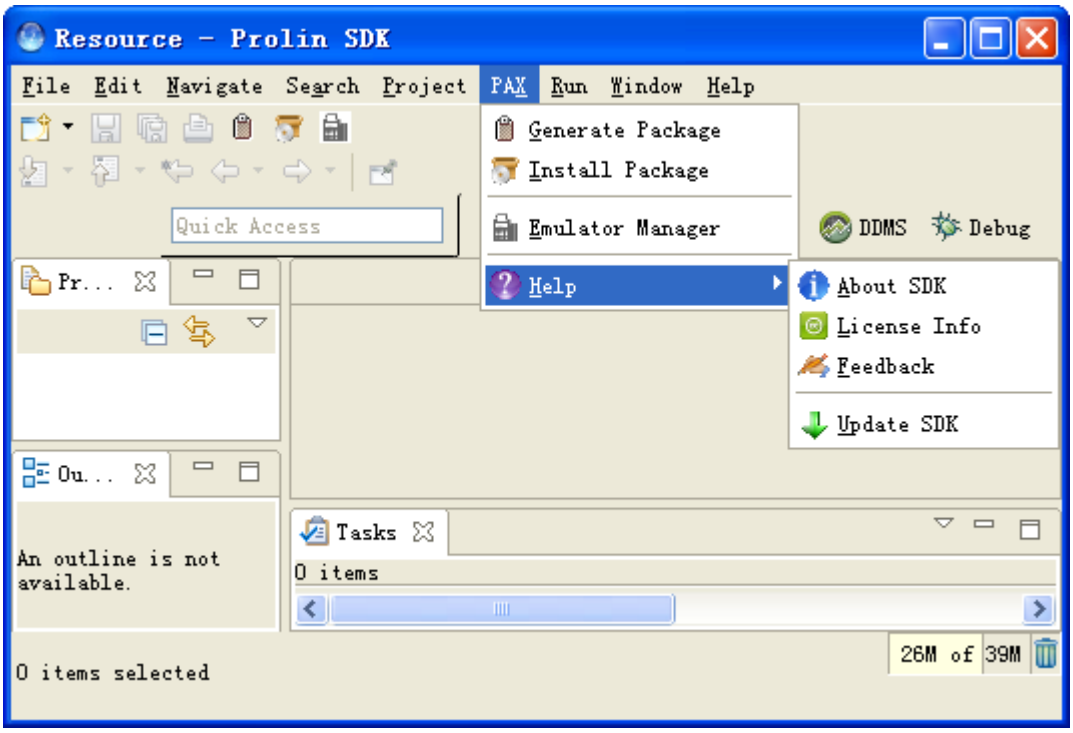

Figure 4.2 PAX menu

4.1.2 Toolbar of PAX

There are 3 toolbar icons about PAX: **Generate Package**, **Install Package**, and **Emulator Manager.** (Figure 4.6)

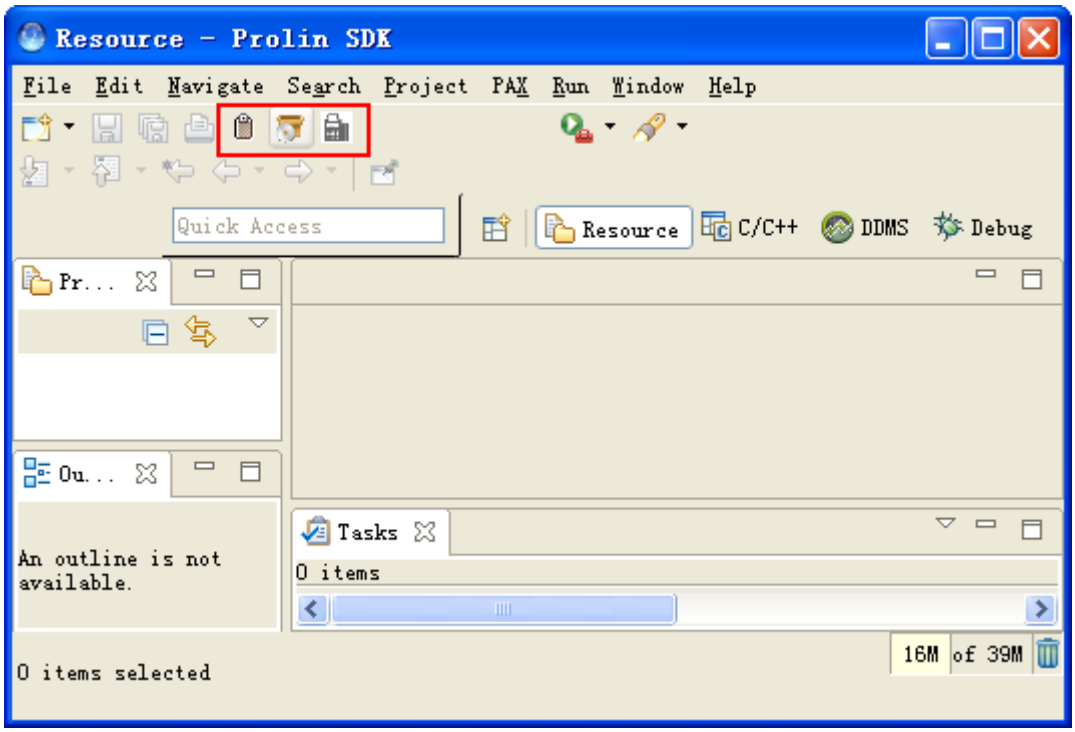

Figure 4.3 PAX toolbar

NOTE

Move the mouse pointer above the toolbar icon, tips of function will popup.

4.2 Settings

Choose menu **Windows→Preferences**, **Preferences** page will be open (P4-4). Then you can set documents language, emulator path and SDK (library and tool-chains) path.

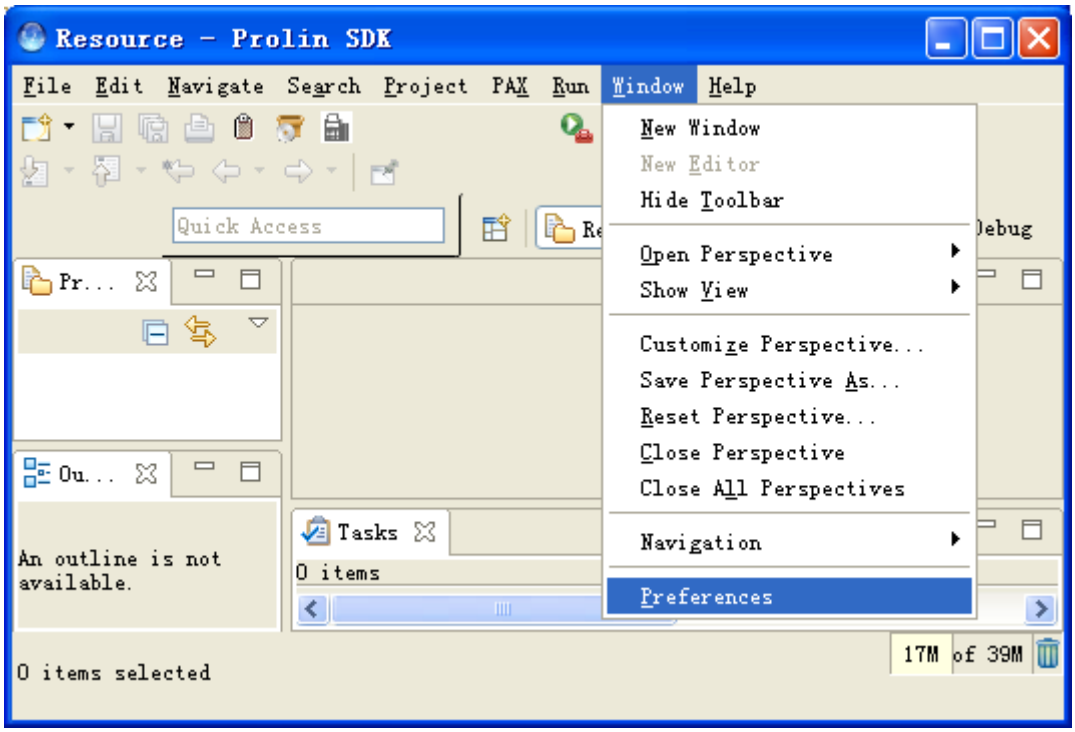

Figure 4.4 Open preferences

4.2.1 Set Docs Language

Choose English or Chinese Language to set documents language (Figure 4.5).

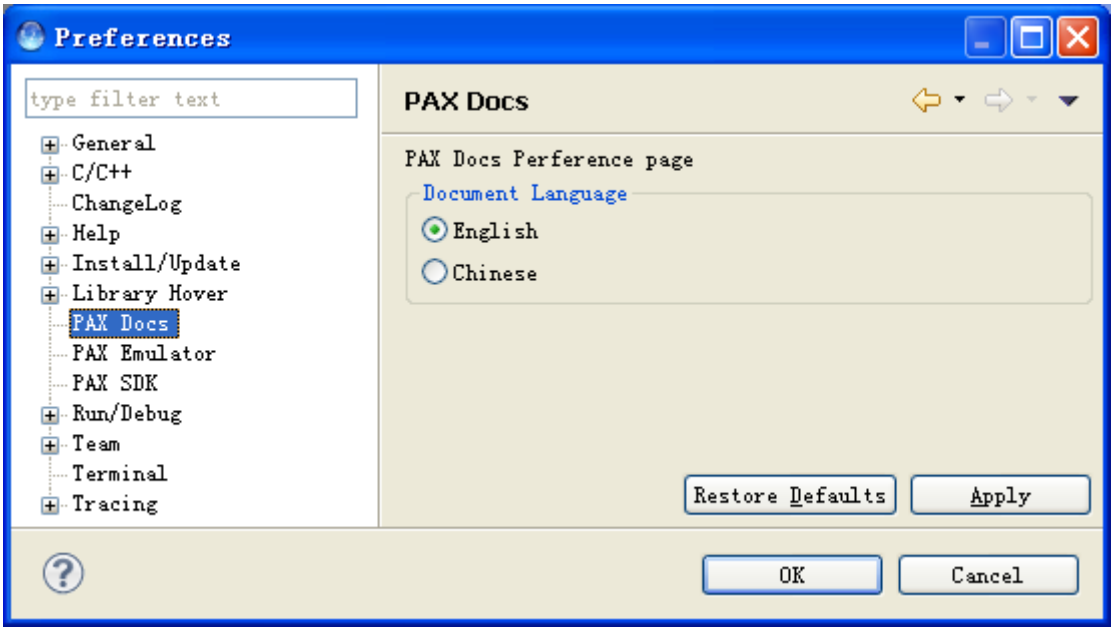

Figure 4.5 Set PAX Docs

4.2.2 Set Emulator Path

Emulator is in the current SDK directory. (Figure 4.6)

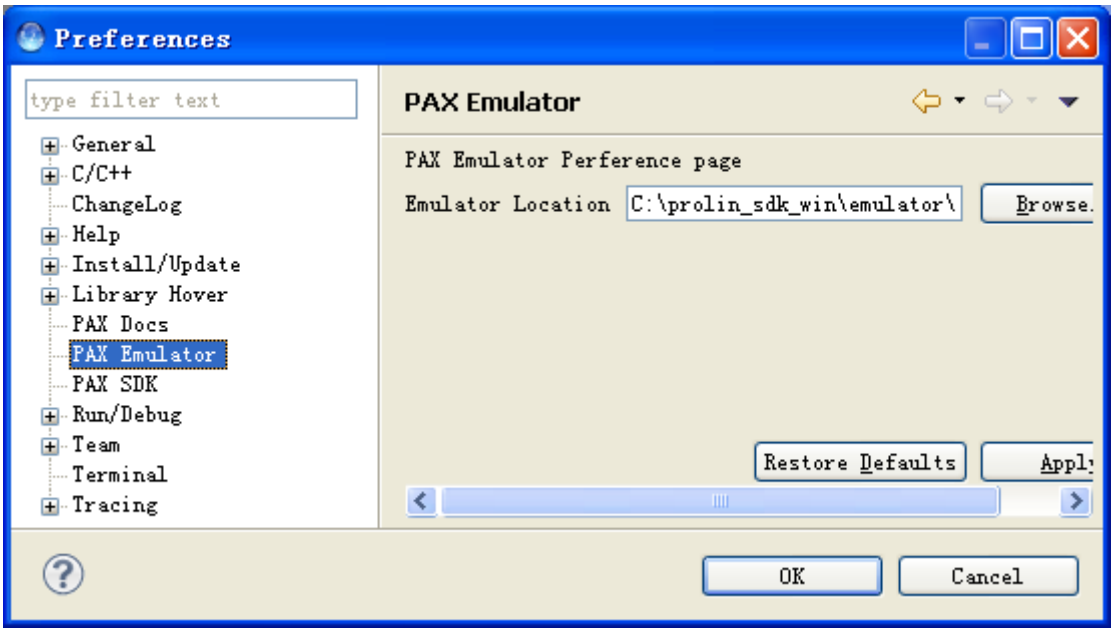

Figure 4.6 Set PAX Emulator

4.2.3 Set SDK path

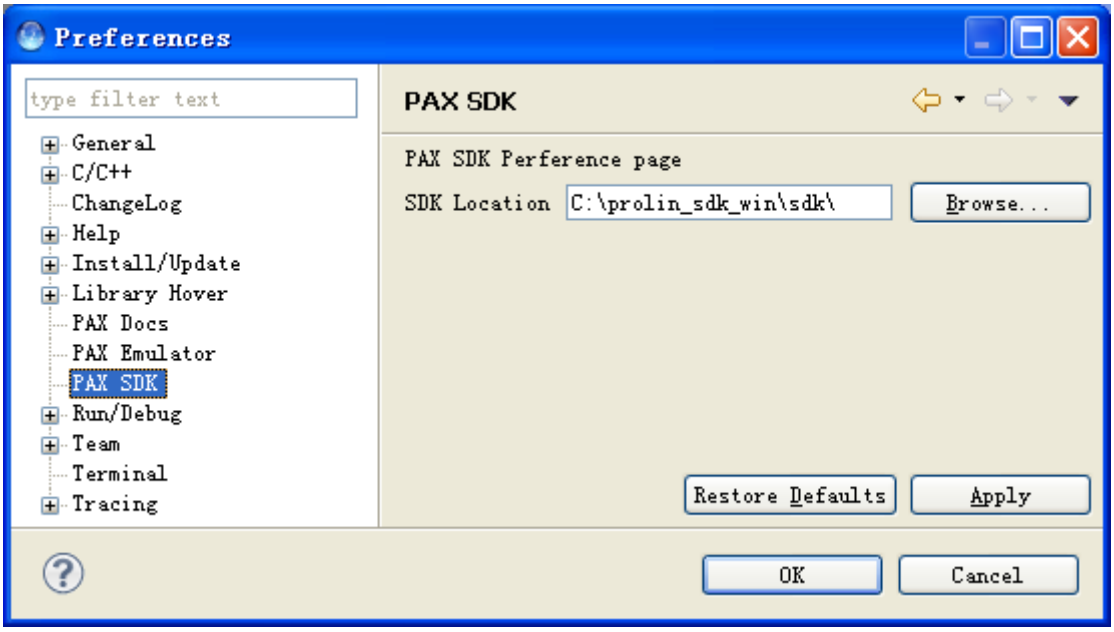

Figure 4.7 Set PAX SDK

4.2.4 Set default encoding

Choose menu **Windows→Preferences**, preferences page will be open. Choose **General→Workspace**, set **Text file encoding** to be **UTF-8(**Figure 4.8). Then when you create a new project, its code will be automatically set to UTF-8.

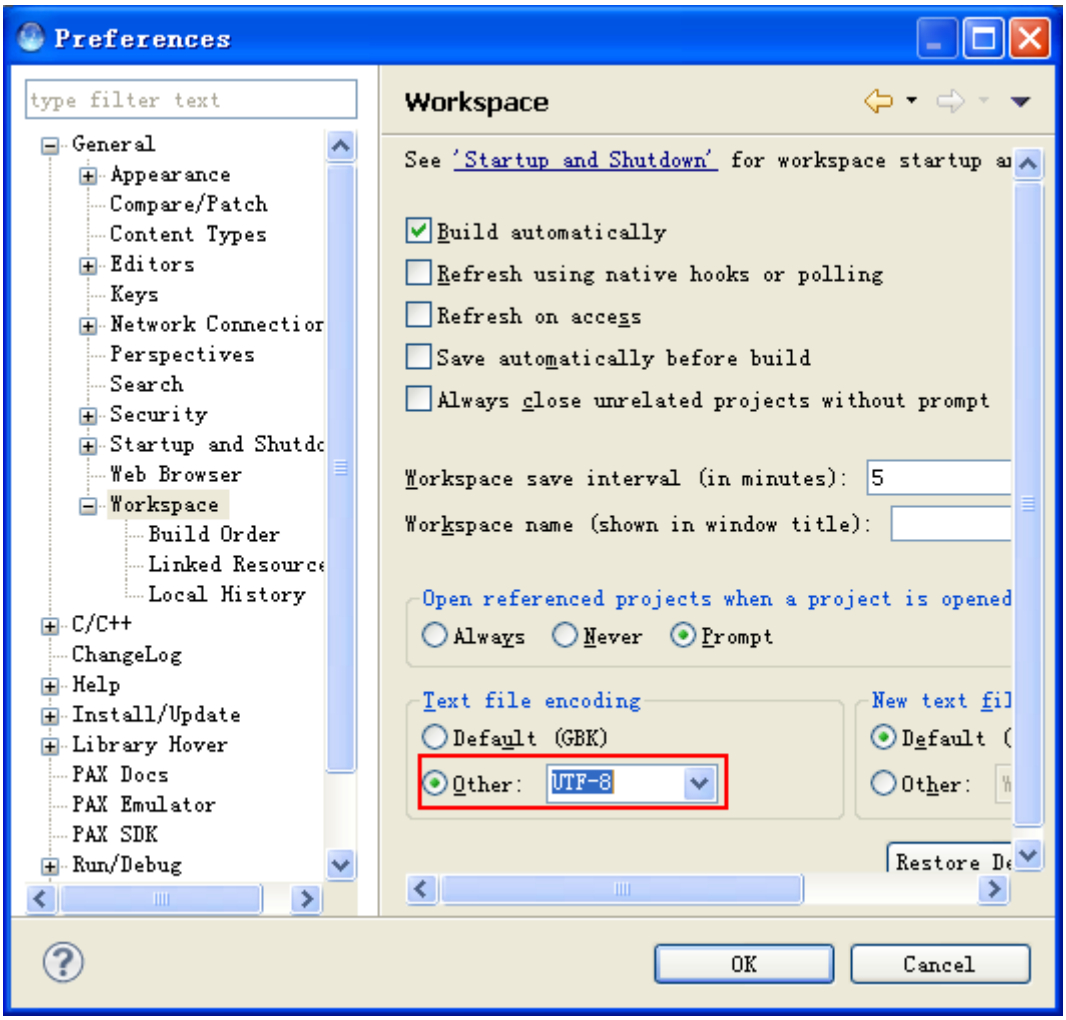

Figure 4.8 Set encoding

4.2.5 Set Shortcuts

Choose Menu **Windows→Customize Perspective**, then choose **Shortcuts** sub page, then set **PAX Projects** as below (Figure 4.9).

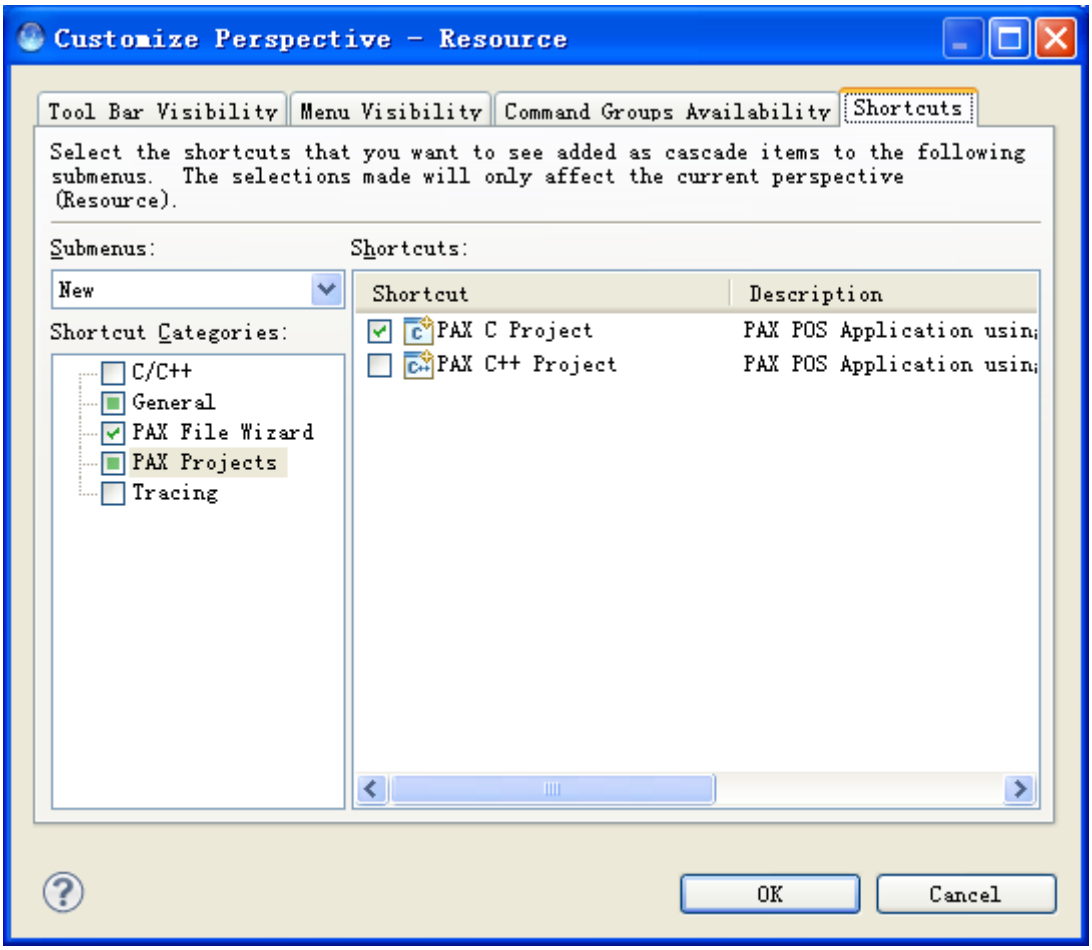

Figure 4.9 Customize Shortcuts

Now **PAX Projects** has been added to the **New Shortcuts**. Let's see the result. Go to eclipse main page, choose menu **New**, you will see **PAX Projects** has been added (Figure 13.7).

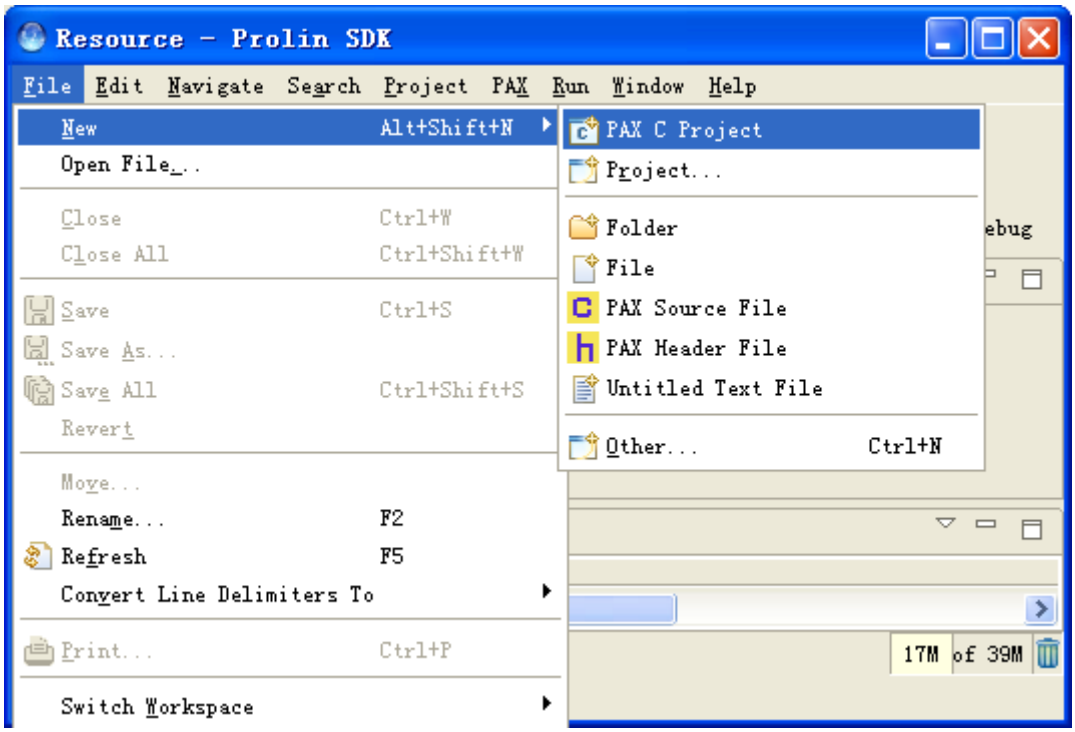

Figure 4.10 Shortcut Setting Result

5 Application Developments

You can use SDK to develop your own apps. Remember that the apps built by SDK are not signed. So they can only run on the emulator or the POS not requests signatures.

If you want to sign the app, please contact PAX support team for help.

5.1 Create Project

Choose menu **File→New→PAX C Project**, Wizard **New Project** will be open (Figure 5.1)

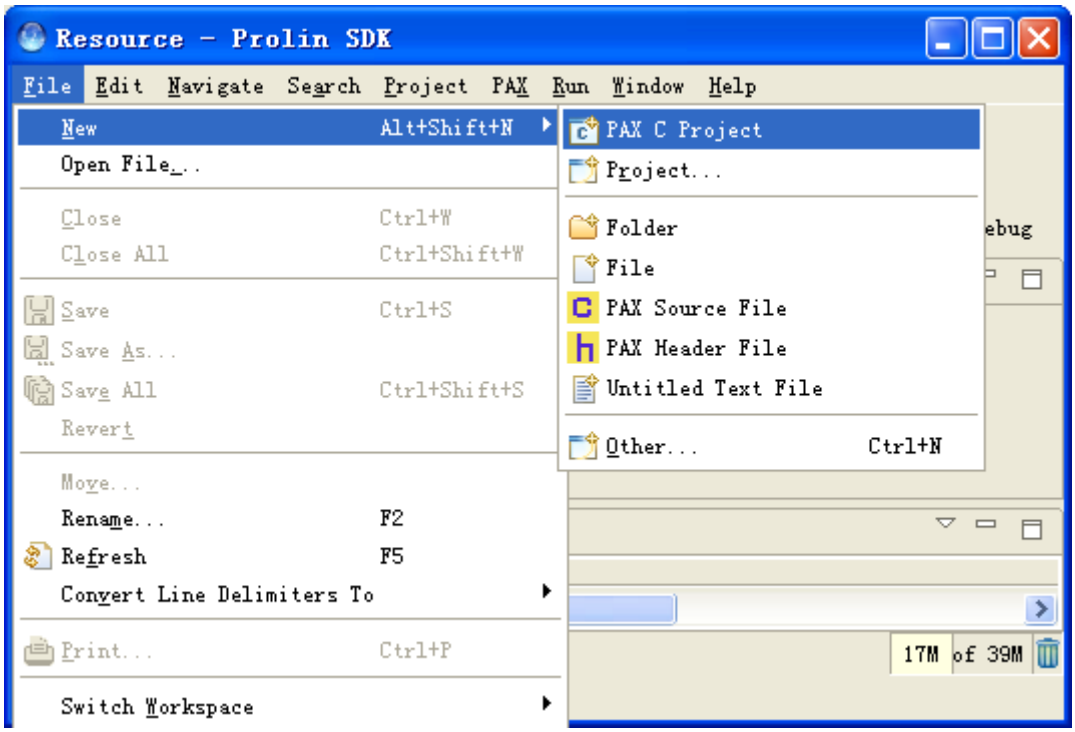

Figure 5.1 Create Project

Input **Project name**, choose **Project type**, and then click on **Next** button (Figure 5.2).

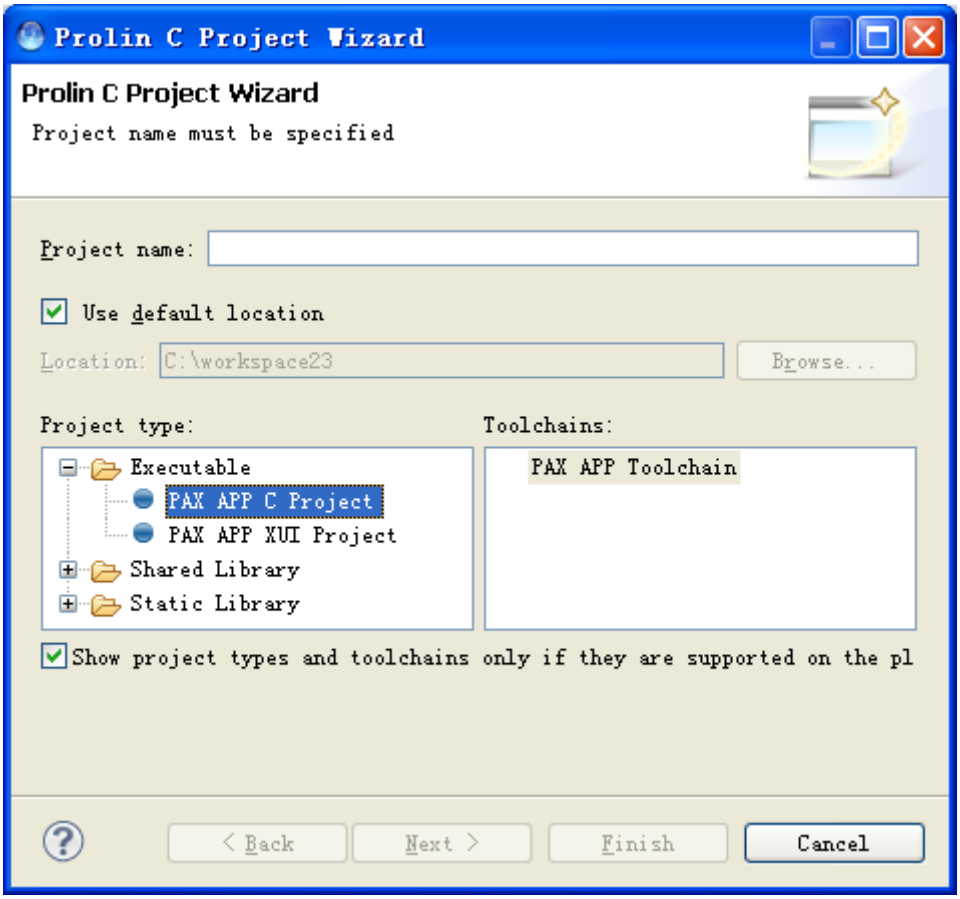

Figure 5.2 Project wizard

Keep default settings, and click on **Finish** button (Figure 5.3).

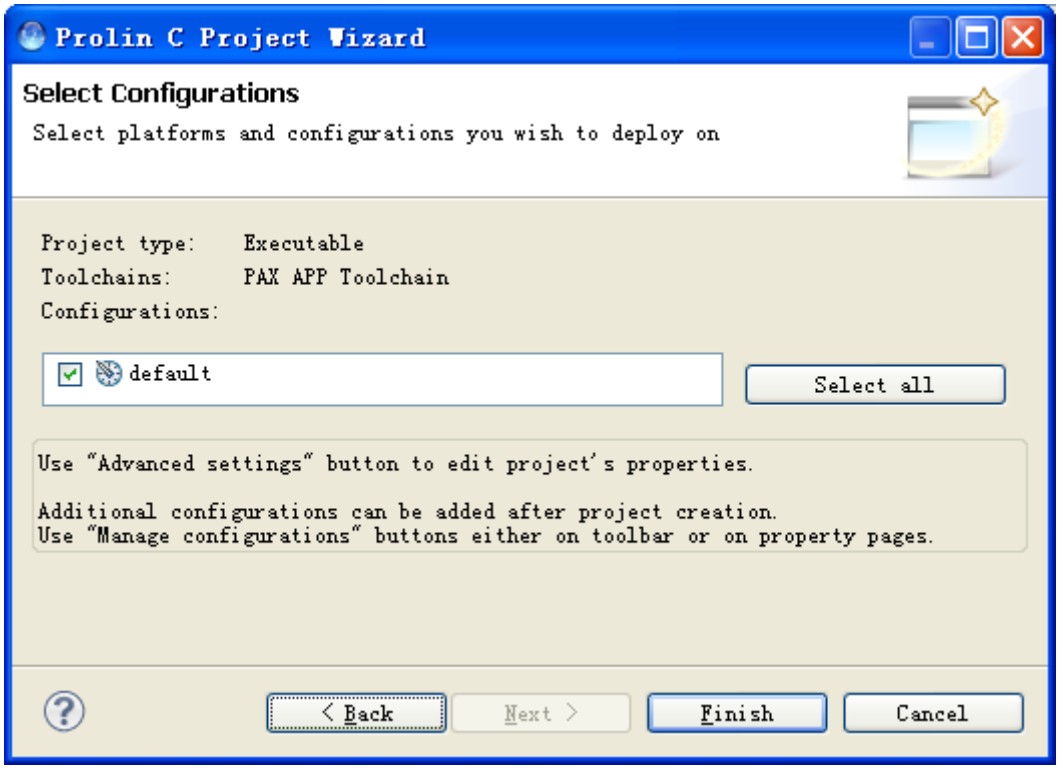

Figure 5.3 Project wizard

If current page is **Welcome** page, close the page or restore to the **Project Explorer** page (Figure 5.4).

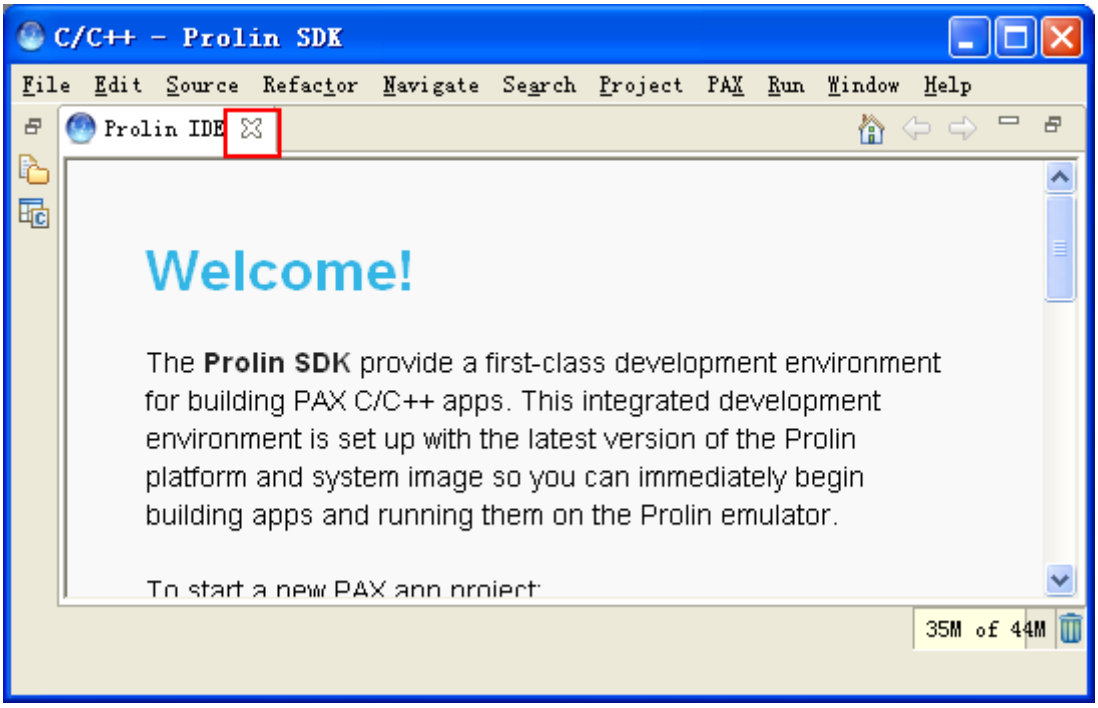

Figure 5.4 Switch to project explorer

Now the whole view looks like this (Figure 5.5):

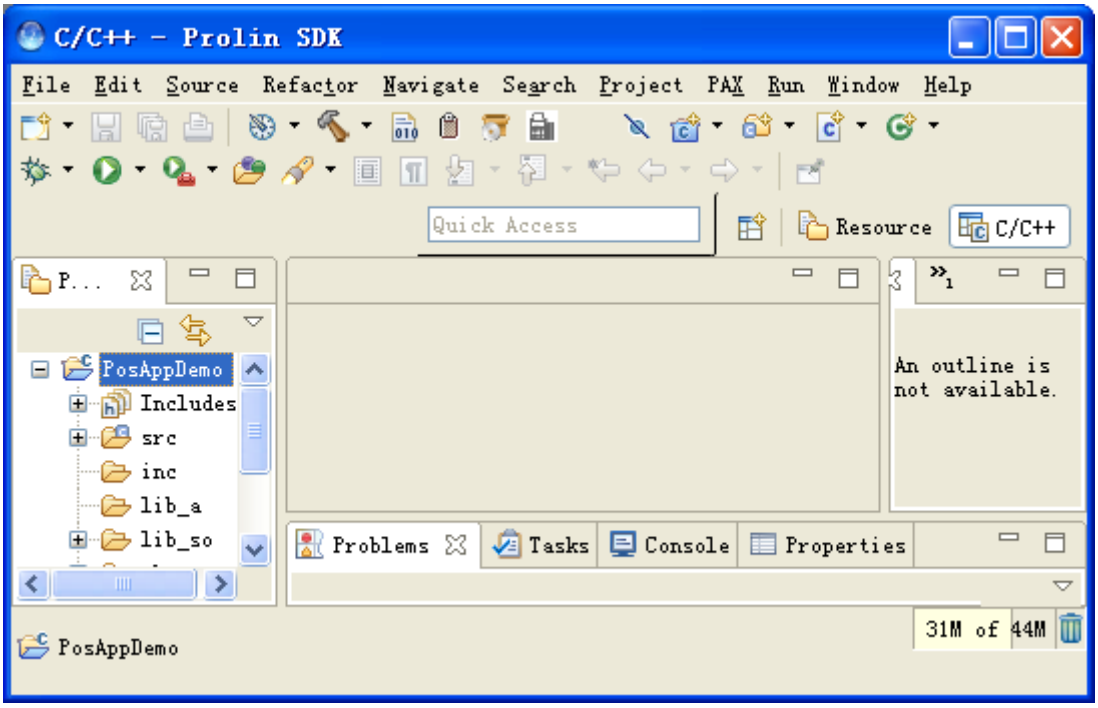

Figure 5.5 Project explorer interface

5.2 Build Project

There are 3 common ways to build a project.

1st, Choose target project, then click on **Build** button in toolbar.

2nd, Choose target project, then choose menu **Project→Build Project**.

3rd, Choose target project, then right click on the mouse, from the popup context menu, choose **Build Project** (Figure5.6, Figure5.7)**.**

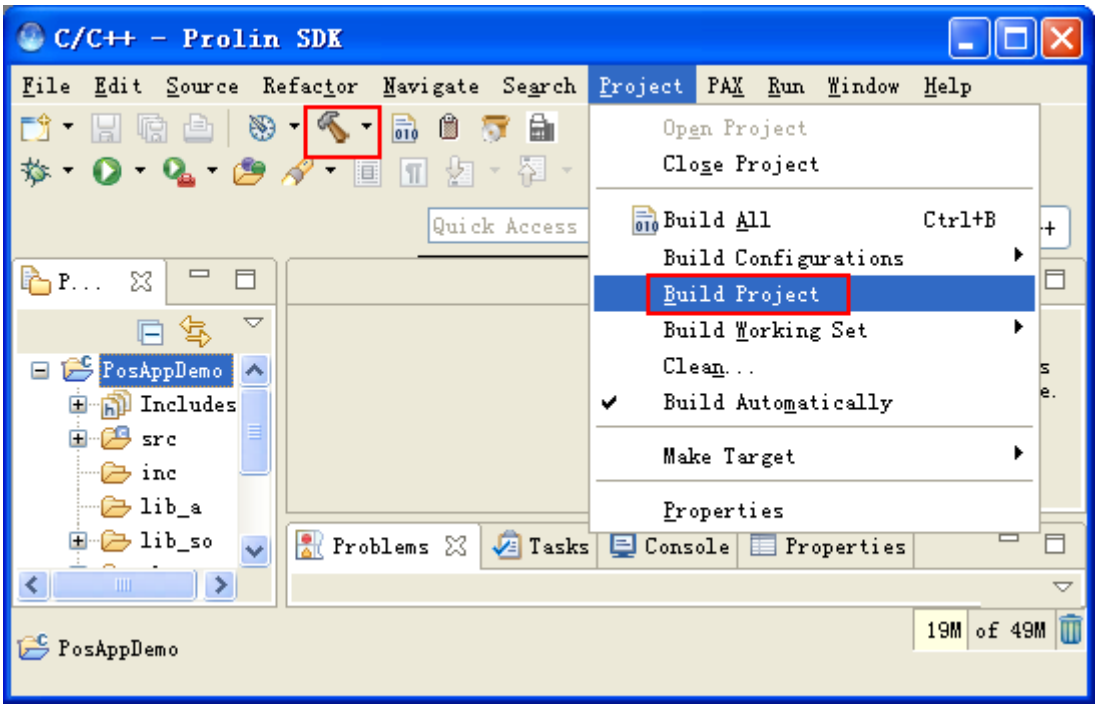

Figure 5.6 Build project

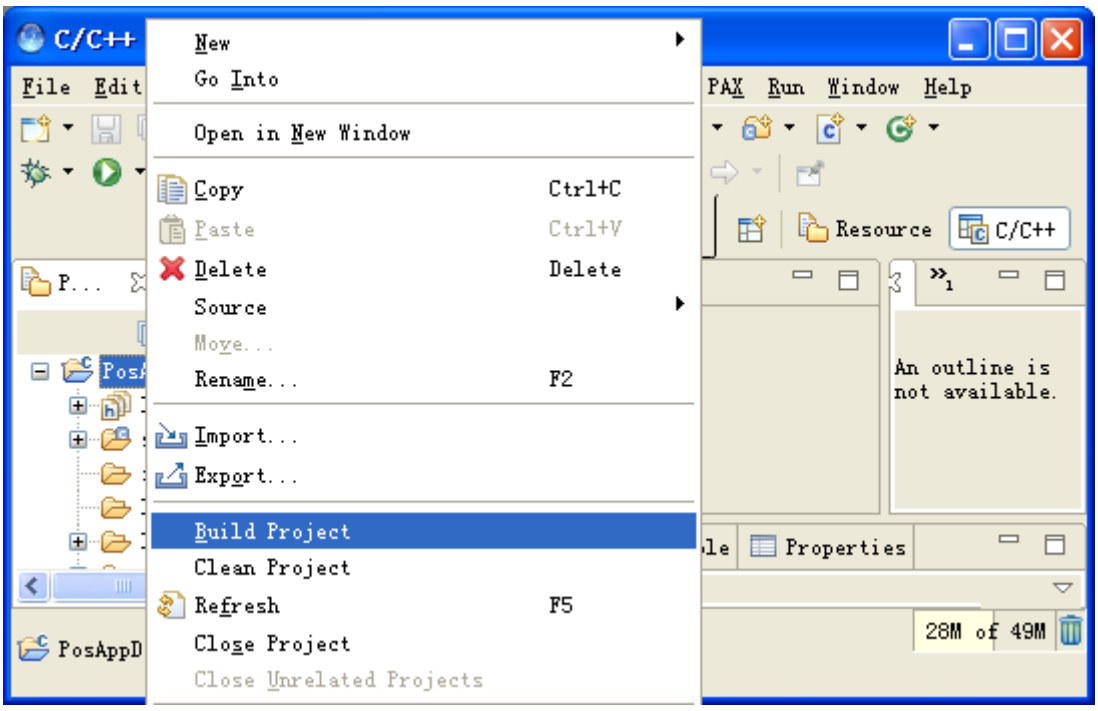

Figure 5.7 Build Project

Building process interface (Figure 5.8):

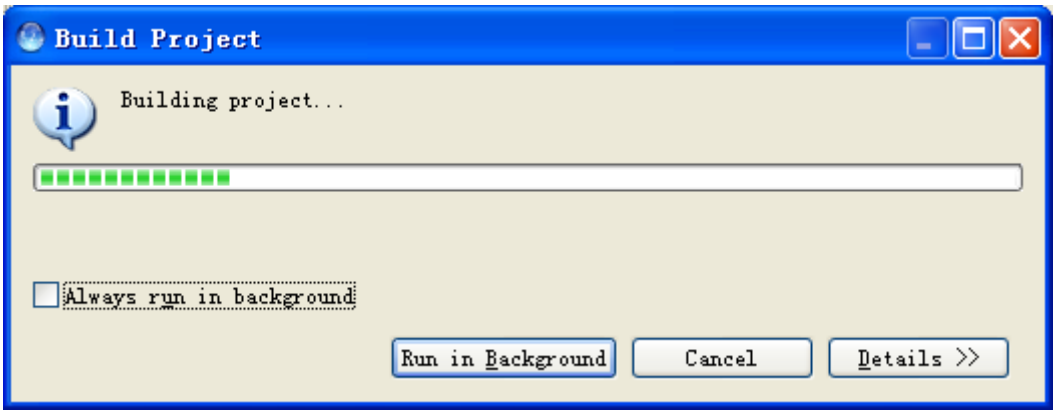

Figure 5.8 Build Project

After successfully building, the **binaries** will be created (Figure 5.9).

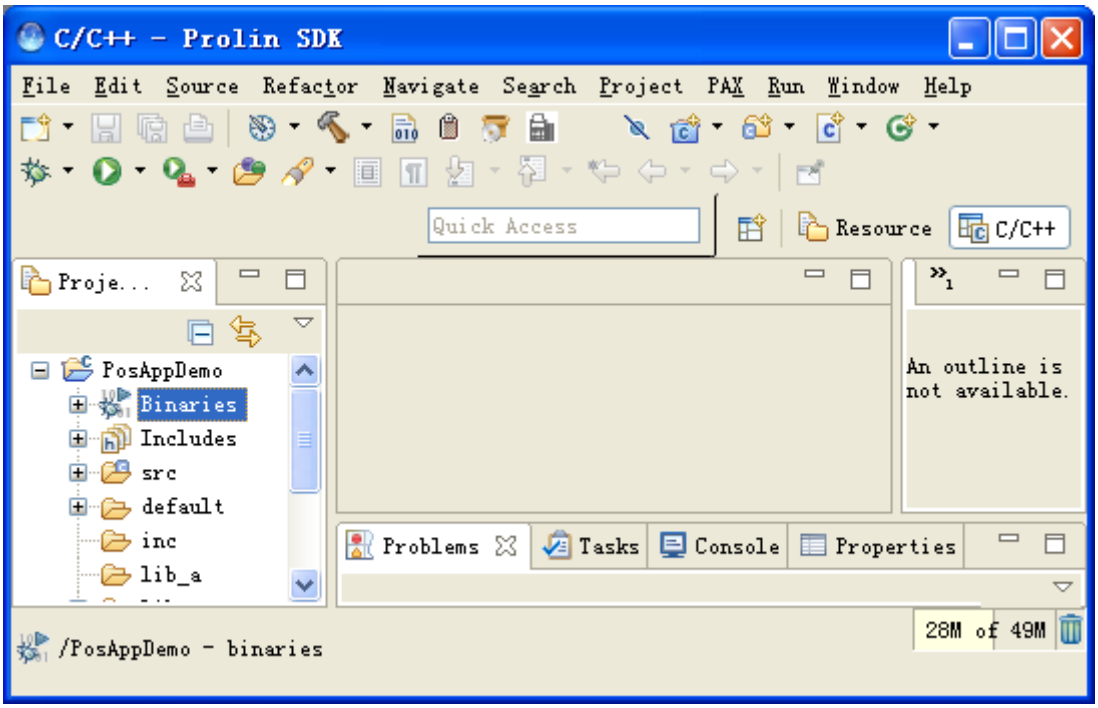

Figure 5.9 Building result

5.3 Generate Package

There are also 3 ways to generate package. Choose menu **PAX→Generate Package** or click on toolbar **Generate Package** or choose context menu **Generate Package**, .aip file will then be created in pkg folder (Figure5.10).

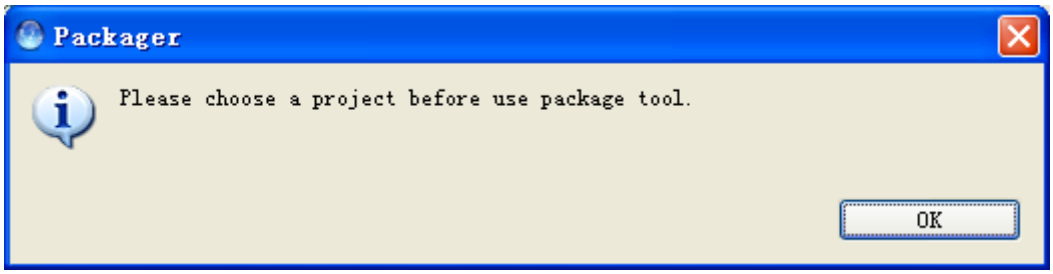

Figure 5.10 Generate package

5.4 Install Package

You can install the package to the emulator or physical pos machine by SDK. You can also use U-Disk or TermAssist to install the package.

Before installing, open xcb service for the POS side from the TM. Choose **Main Menu → 1.System Config → 4.XCB Service**. Set the connection type of XCB Service to be **COM**, **USB**, or **Network** (Figure5.11).

Figure 5.11 XCB Service

Click on the **Install Package** button, the installer dialog will pop up. There is a list box to show all the devices connected, a **Console** box to show XCB running result, a **user input** box for users to type all the XCB commands, An **Add Device** button to add new device, and an **Install** button to run install package command (Figure 5.12).

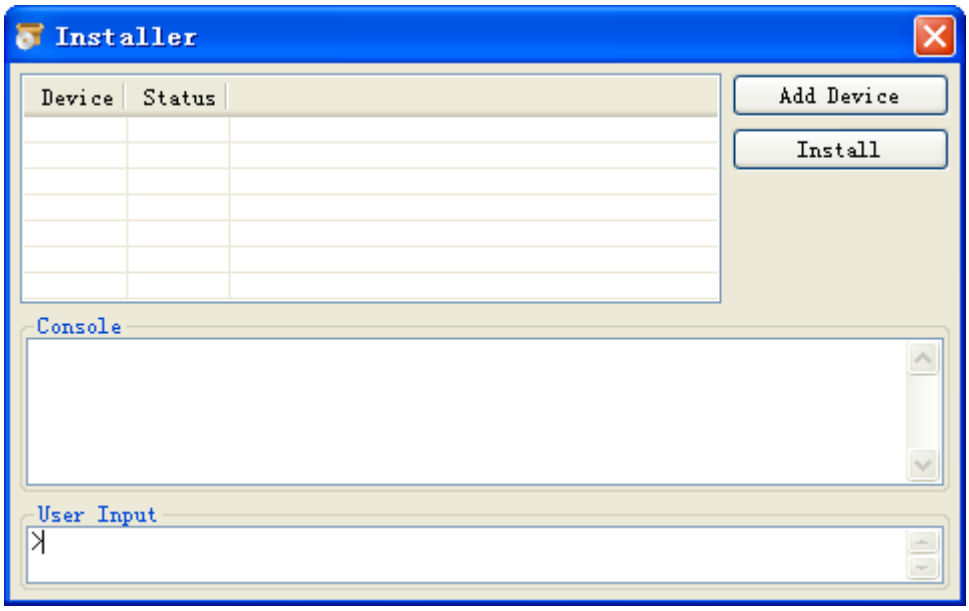

Figure 5.12 Installer

You must choose a device before installing. Click on **Add Device** button to add a new device. When the **Add Device** dialog shows itself, just set the connection parameters and then press **Connect** button (Figure 5.13).

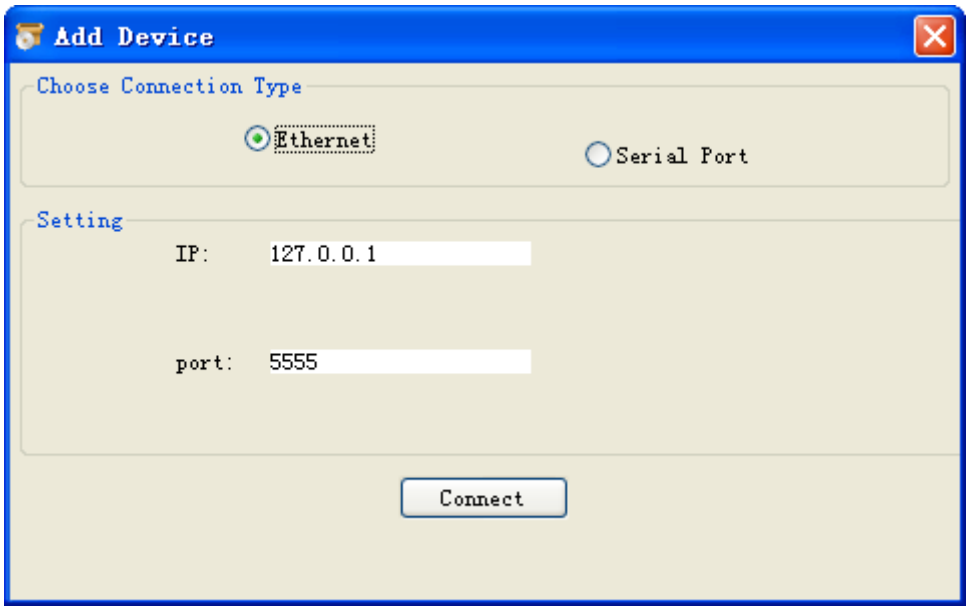

Figure 5.13 Add Device

If the connection is established, the connected device will be show in the list of the installer dialog (Figure 5.14).

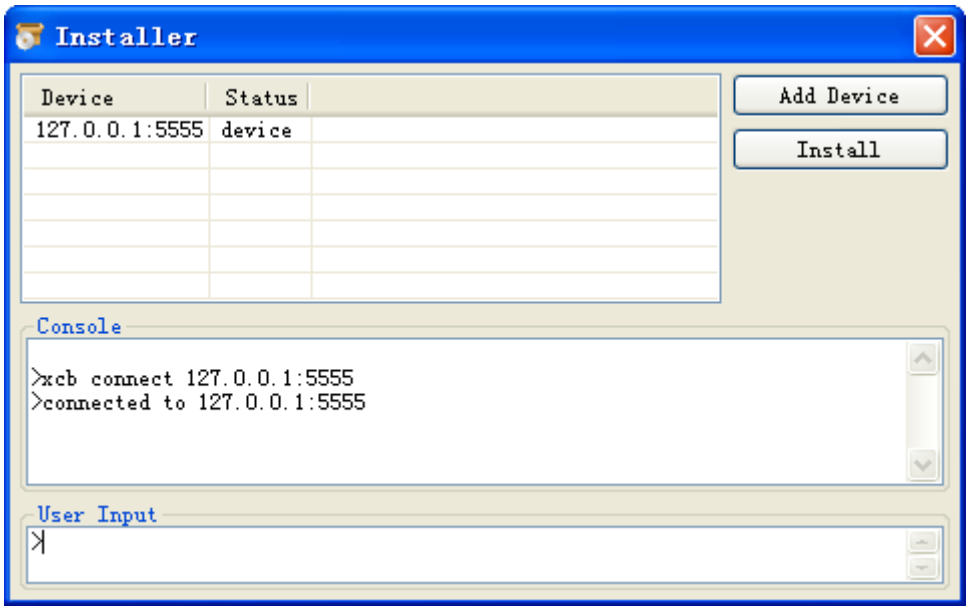

Figure 5.14 Installer

You can also type XCB command "xcb connect ***" in the **User Input** box to connect devices (Figure 5.15). And also, you can type any XCB commands in this box, such as "xcb devices", "xcb logcat", etc.

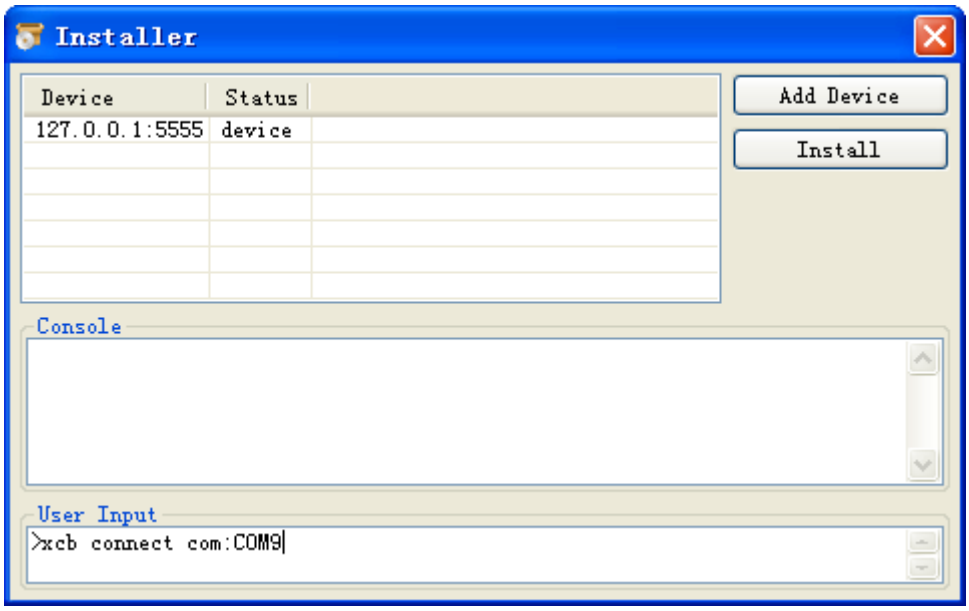

Figure 5.15 Installer

Choose the device and press **Install** button. See the log in the **console** to see the installing result.

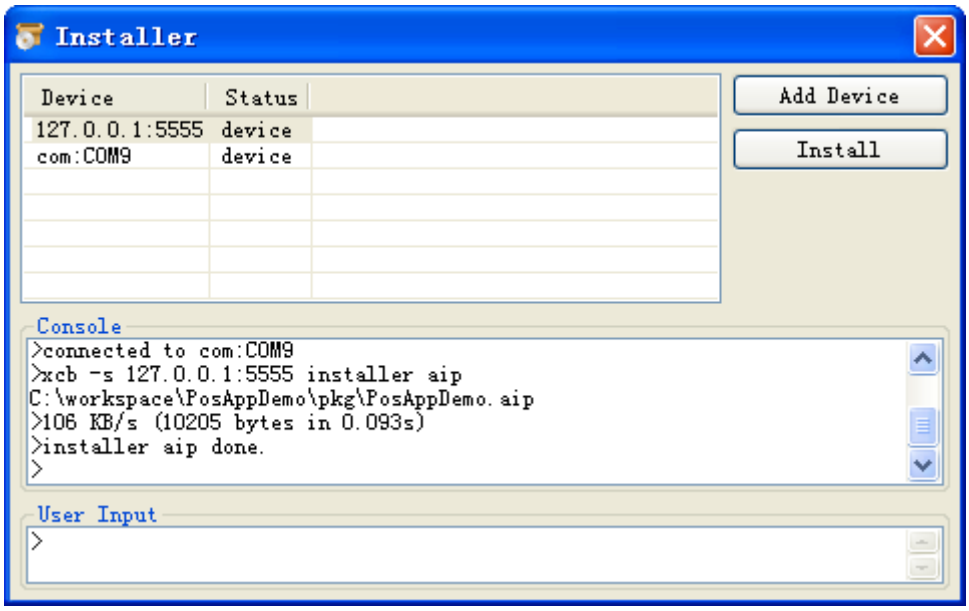

Figure 5.16 Installer

You can see the running result now (Figure 5.17).

Figure 5.17 Running result

5.5 Import Project

You can import Existing project into the **Project Explorer**. Choose menu **File→Import** (Figure 5.18) then the **Import** wizard page will pop up.

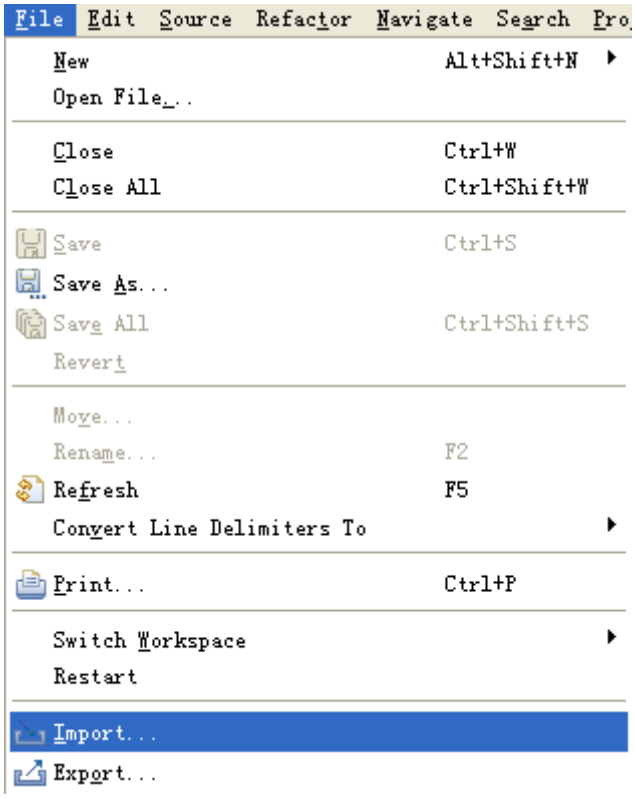

Figure 5.18 Import project

From the **Import** wizard page, choose **General→Existing Projects into Workspace**, and then click on **Next** button (Figure 5.19).

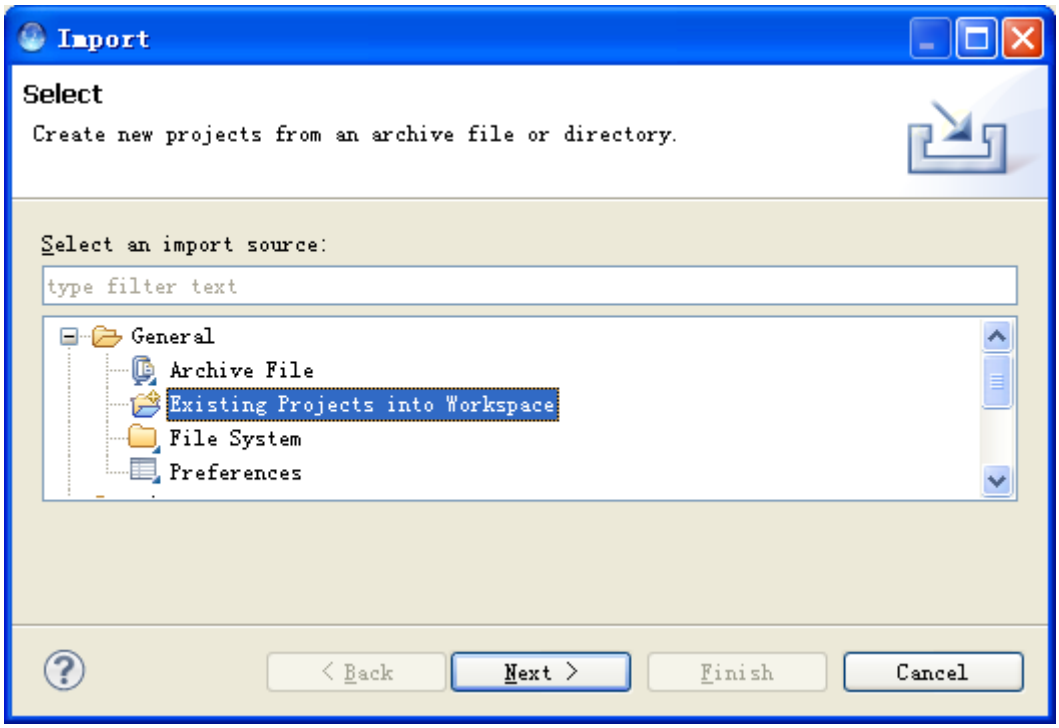

Figure 5.19 Import wizard

From the next **Import** wizard page, set the root directory of the project, and then click on **Finish** button (Figure 5.20).

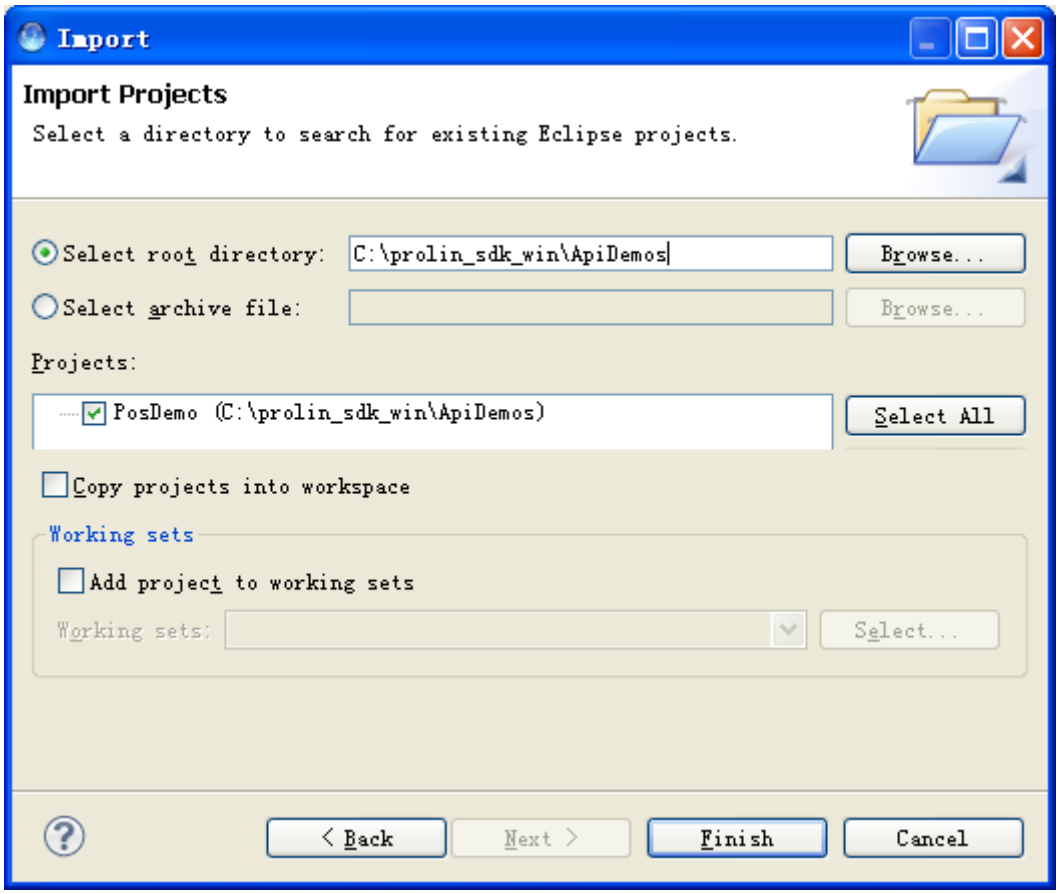

Figure 5.20 Import wizard

Now, the project is added to the **Project Explorer** (Figure 5.21)**.**

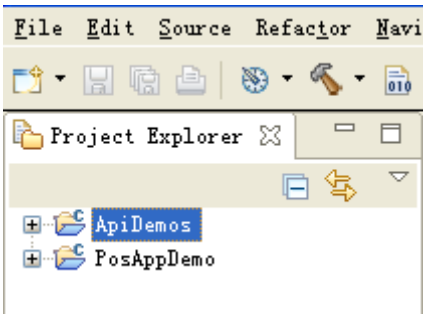

Figure 5.21 Check projects

5.6 Project Environment

By default, the project environment meets most of development requirements. But you can also edit it according to your own needs.

Click on menu **Project→Properties**. From the pop-up properties page, choose **C/C++ Build→Environment**. Now there are several environment variables to set (Figure 5.22).

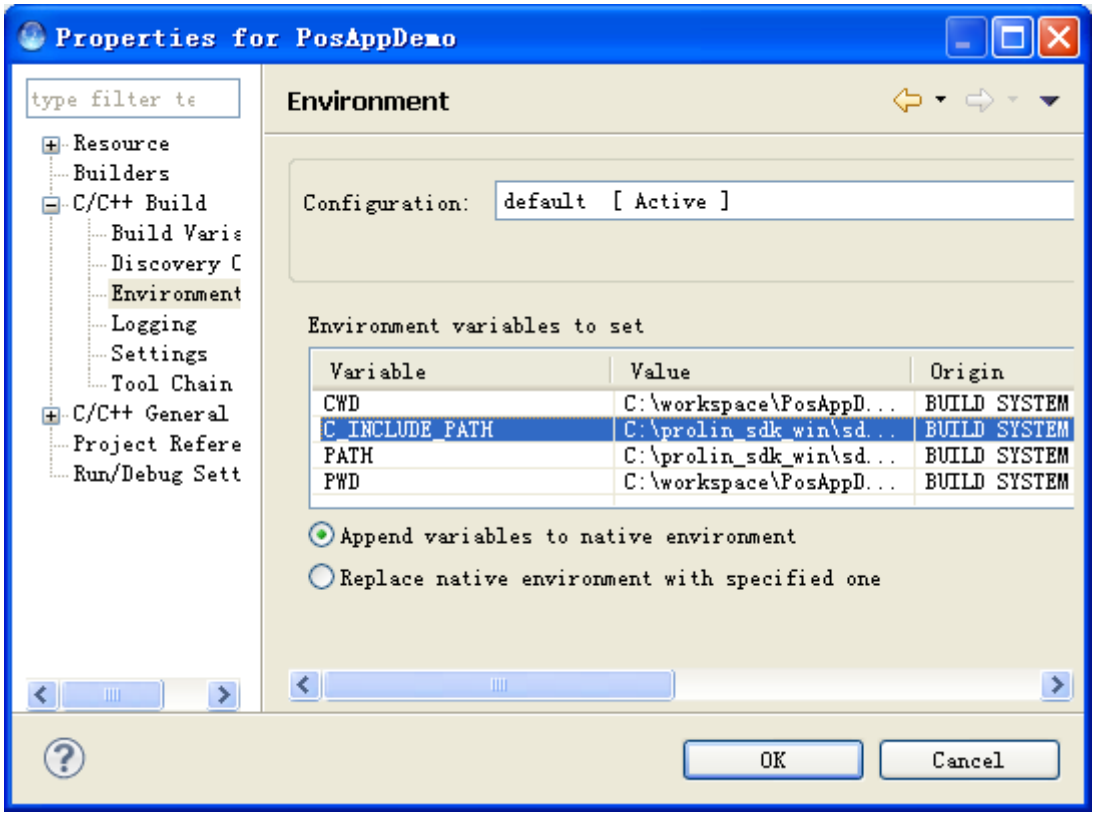

Figure 5.22 Project environment page

For example, you can edit **C_INCLUDE_PATH** to add new header file directory. Just click on button **Edit** and edit the value (Figure 5.23).

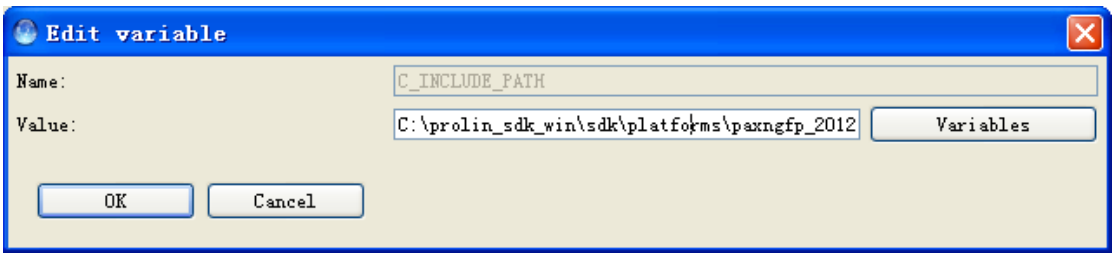

Figure 5.23 Edit variable

Then rebuild the project, you will see new including path in the project explorer (Figure 5.24).

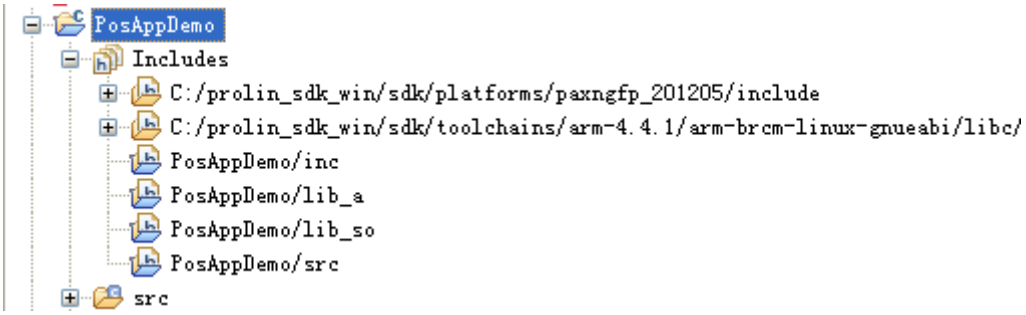

Figure 5.24 Project Explorer

5.7 Project Settings

You can also set project compiler and linker options. From the properties page, choose **C/C++ Build→Settings** (Figure 5.25). There are several settings. Here mainly introduce **GCC Compiler** and **GCC Linker**.

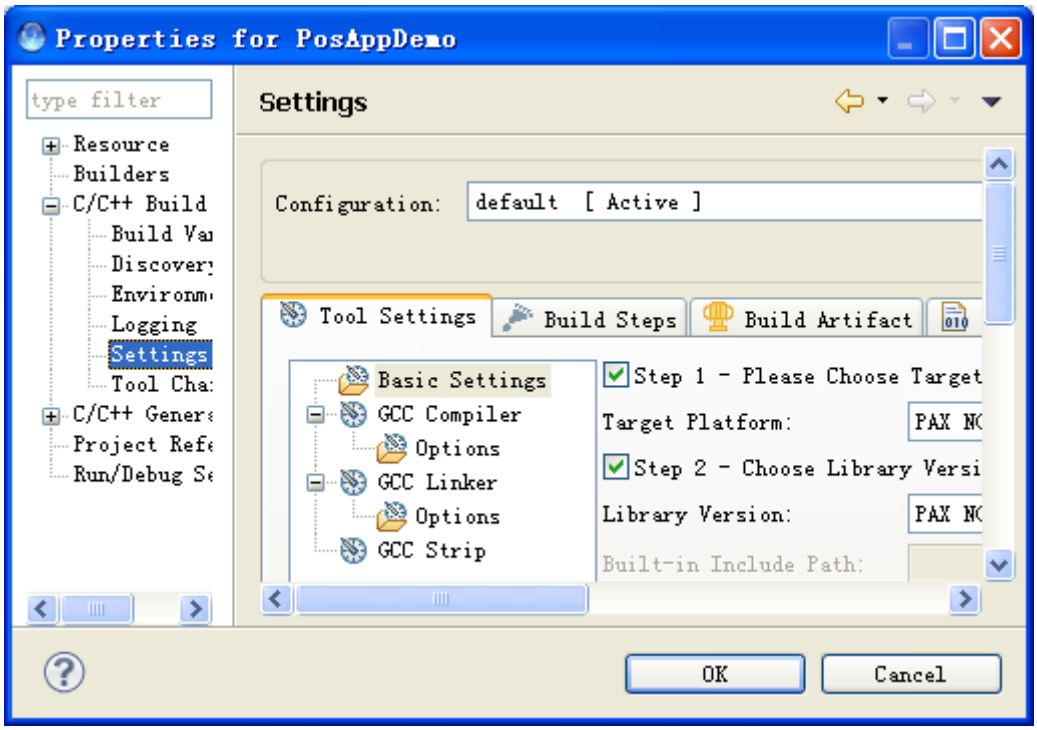

Figure 5.25 Project settings

Choose **GCC Compiler**, you can see all compile options. Click on the **Options** below to set as you wish (Figure 5.26). You can set **Optimization Level**, **Debug Level**, **Compiler misc flags**, etc.

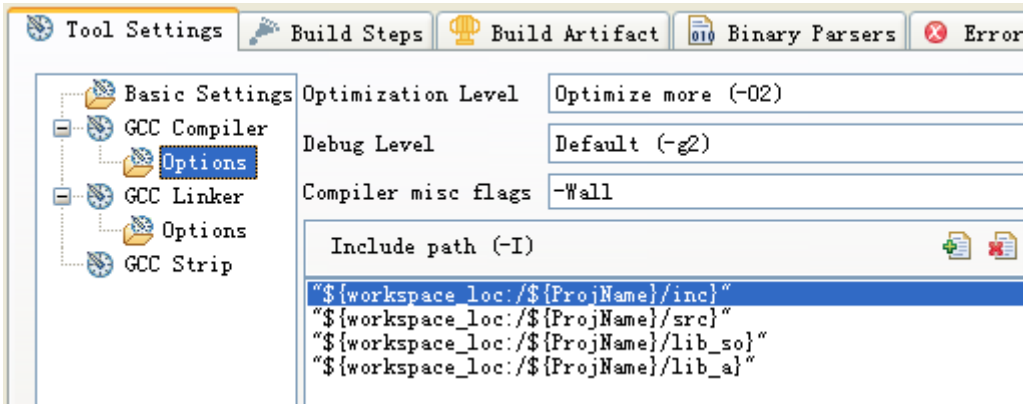

Figure 5.26 Compile options

Also click on the Options below the **GCC Linker** to customize link commans (Figure 5.27).

Figure 5.27 Link options

{ This page intentionally left blank }

6 Library Developments

Both shared LIB and static LIB development are available. The following is about how to build a shared LIB.

6.1 Create Project

Create a new **PAX C Project**, and in the **Prolin C Project Wizard** page, write **Project name** with **PosLibDemo** and choose **Project type** as **PAX Shared Library**. Press **Finish** to complete the wizard (Figure 6.1).

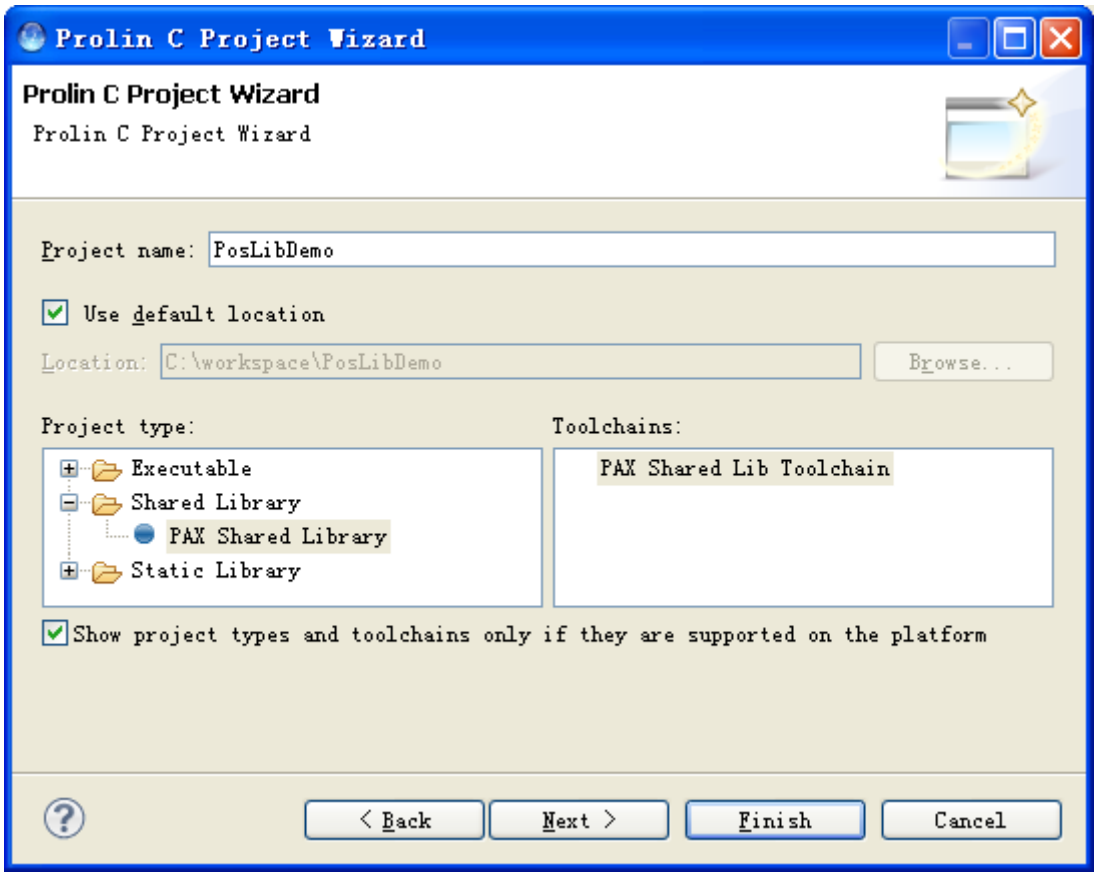

Figure 6.1 Project wizard

6.2 Build Project

Build project and **Binaries** will be created. The target library is **default/LibPosLibDemo.so** (Figure 6.2).

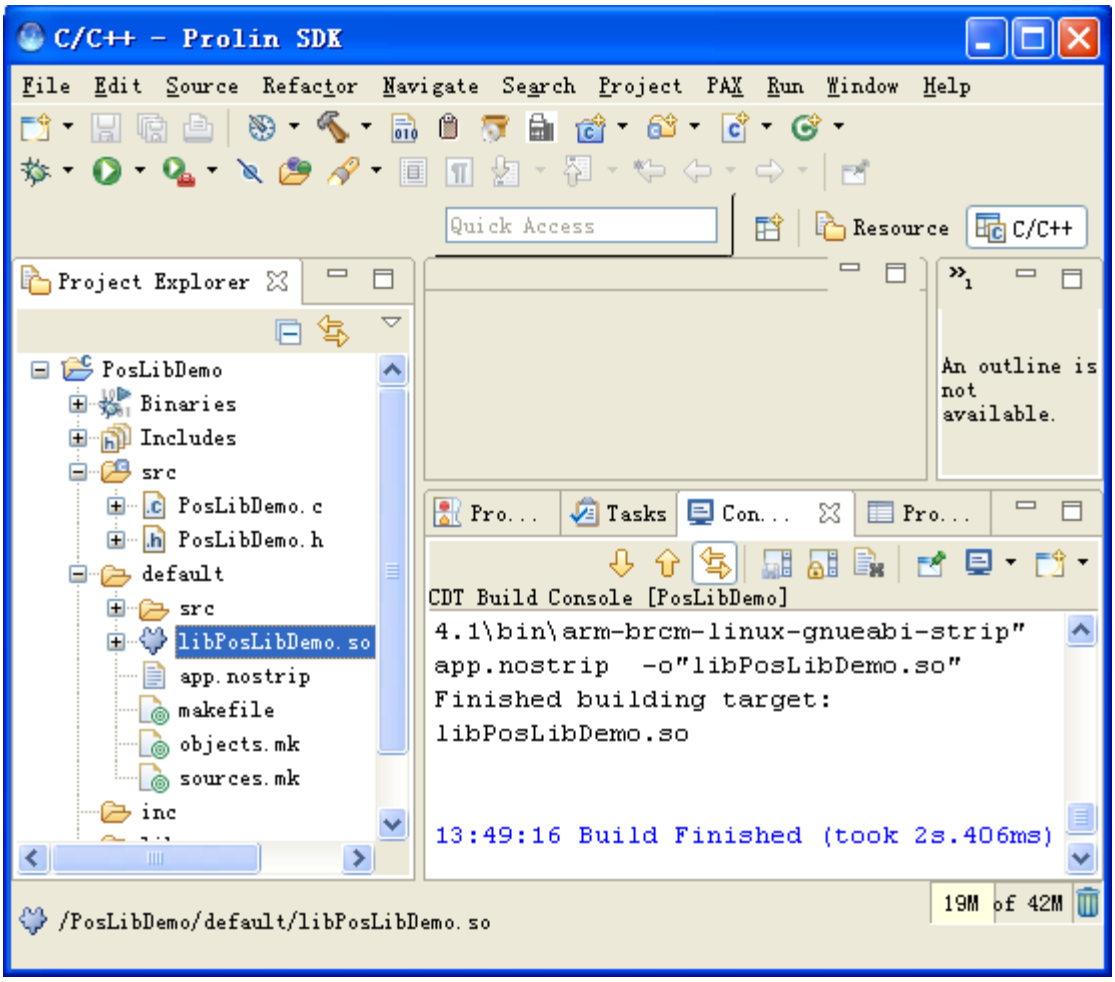

Figure 6.2 Building result

6.3 Use Library

You can use **LibPosLibDemo.so** as common .so file. For example, we'll add **LibPosLibDemo.so** to project **PosAppDemo**.

6.3.1 Copy LIB to Project

Copy **LibPosLibDemo** into folder **lib_so** (Figure 6.4).

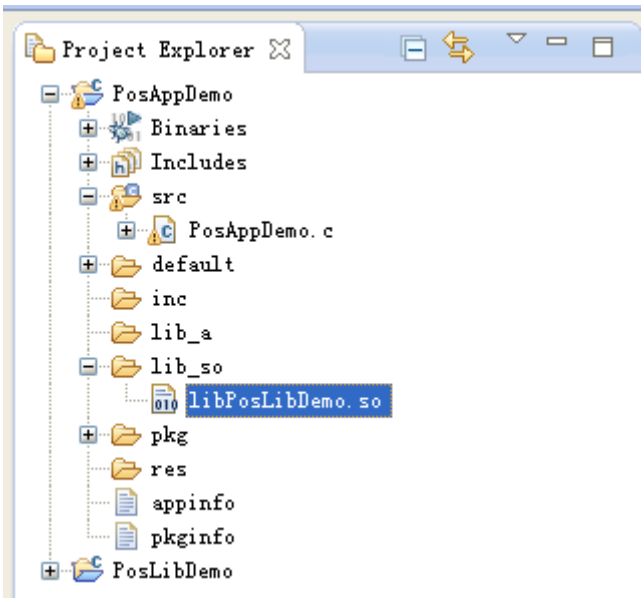

Figure 6.3 Copy library

NOTE

Copy shared library to **lib_so** folder. Copy static library to **lib_a** folder.

6.3.2 Modify Link Option

Right click on the project and from the popup context menu choose **Properties**.

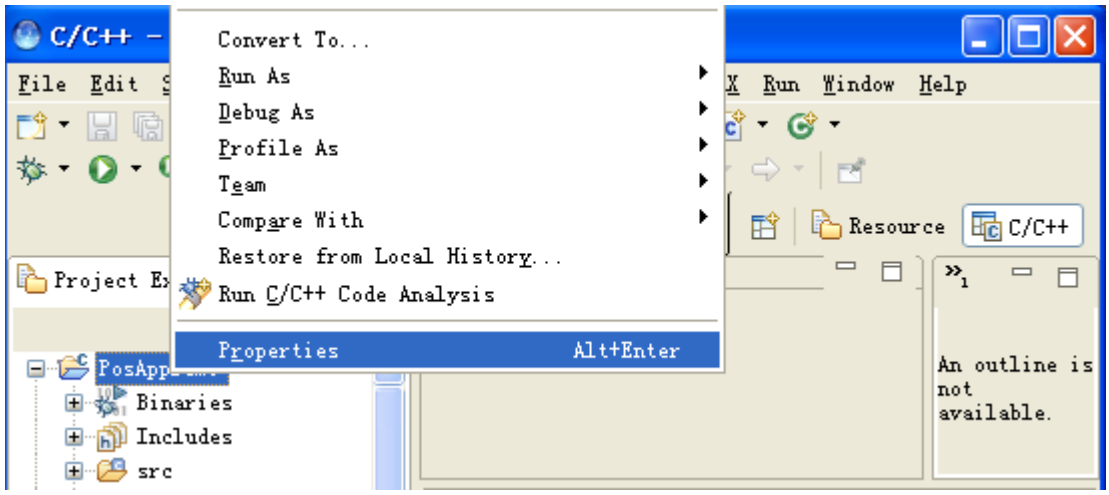

Figure 6.4 Choose properties menu

In the **Properties** page, choose **C/C++ Build→Settings**. In the **Settings** sub page, choose **GCC Linker→Options**. From **Libraries (-l)** list click on (Add) icon (Figure 6.5).

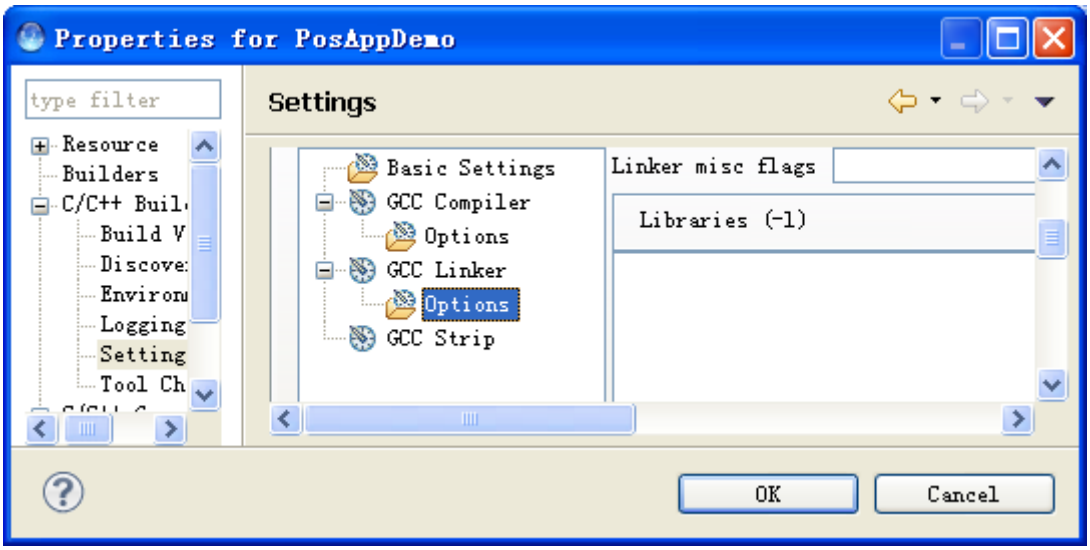

Figure 6.5 Properties Setting

From the popup dialog **Enter Value**, input library name **PosLibDemo**.

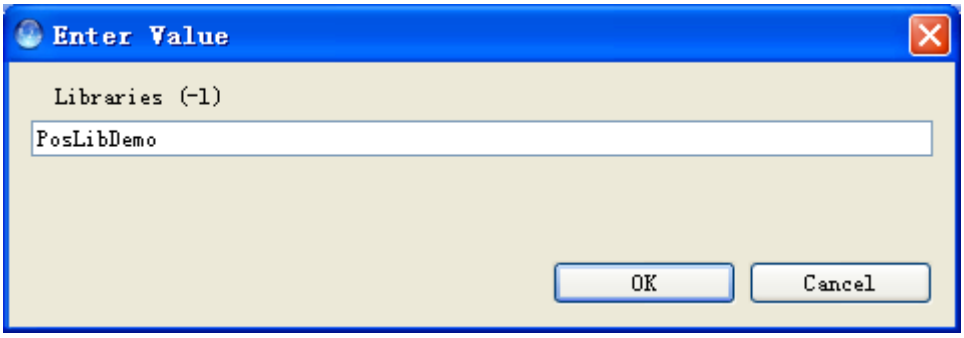

Figure 6.6 Input Library

CAUTION

Needs not to input prefix (**Lib**) and suffix (**.so**) of the library.

NOTE

For SDK v2.7 and later, the folder name “lib_so” is changed to “lib”.

6.3.3 Rebuild Project

Clean the whole project and rebuild (Figure 6.7).

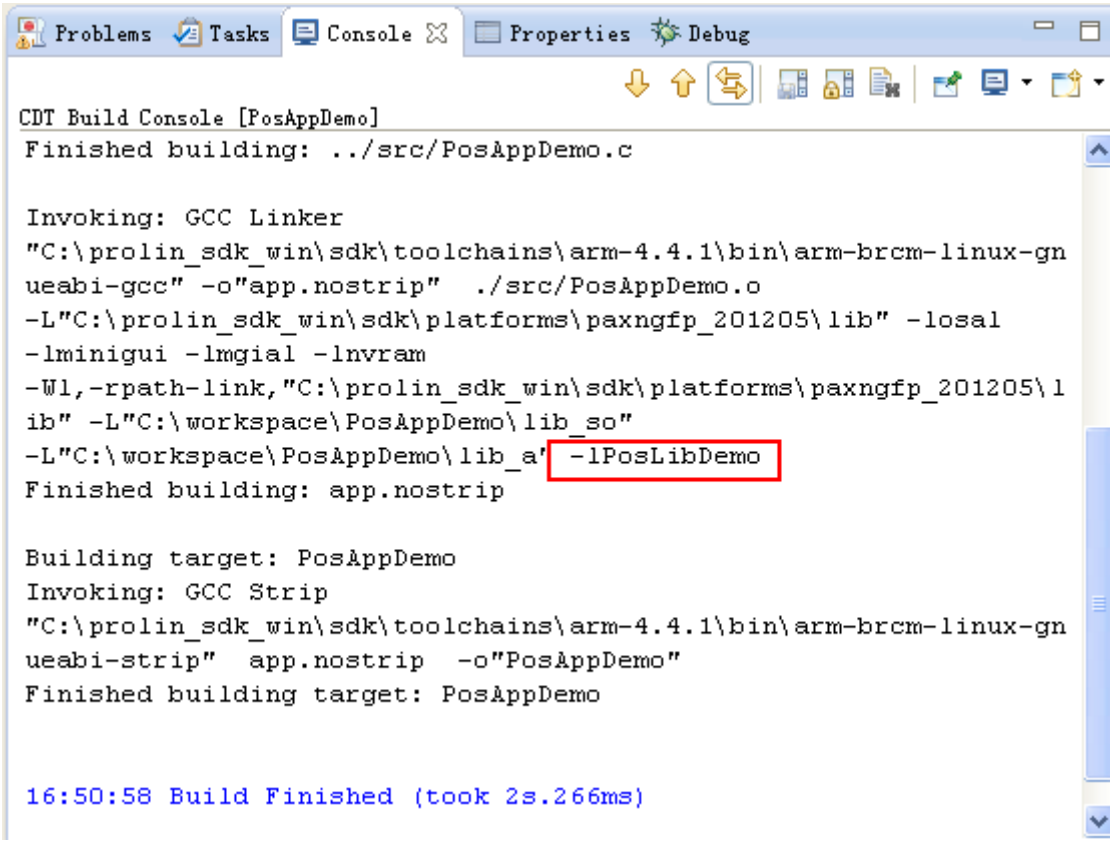

Figure 6.7 building log

7 Code Edit Help

7.1 Query Functions or Keywords

Put the mouse pointer above the function or keyword, then prototype will be shown (Figure 7.1).

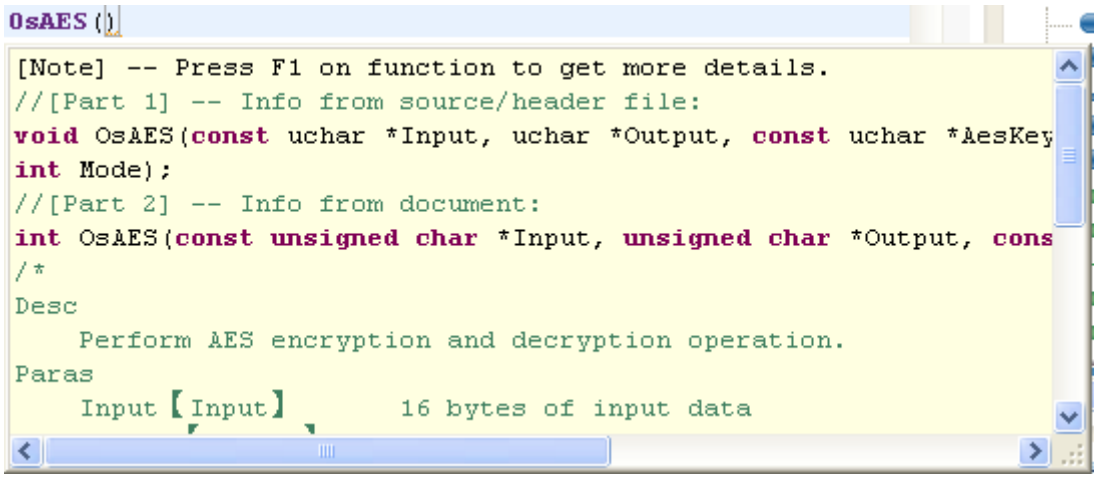

Figure 7.1 Query result

7.2 Query Details from Documents

Put the mouse pointer above the function or keyword, and then press **F1** on the keyboard. Then search result will be shown (Figure 7.2).

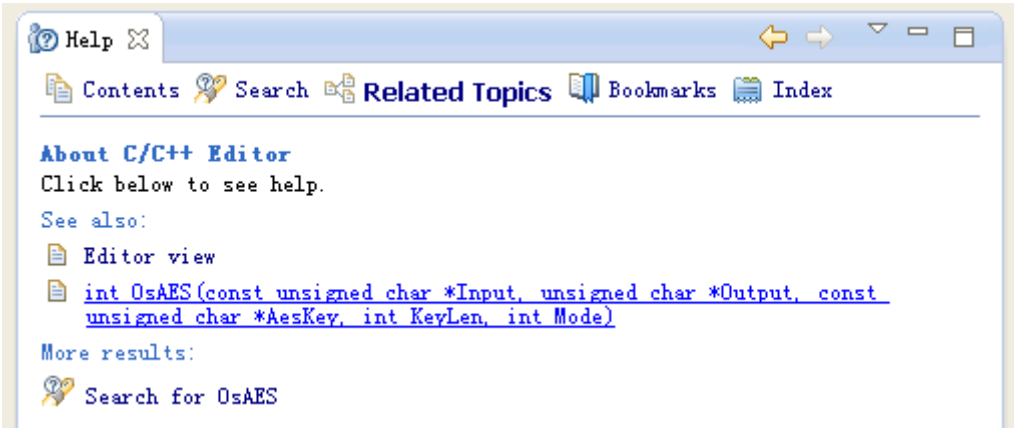

Figure 7.2 Query result

Go to the Link of What your need, and you will get more details (Figure 7.3).

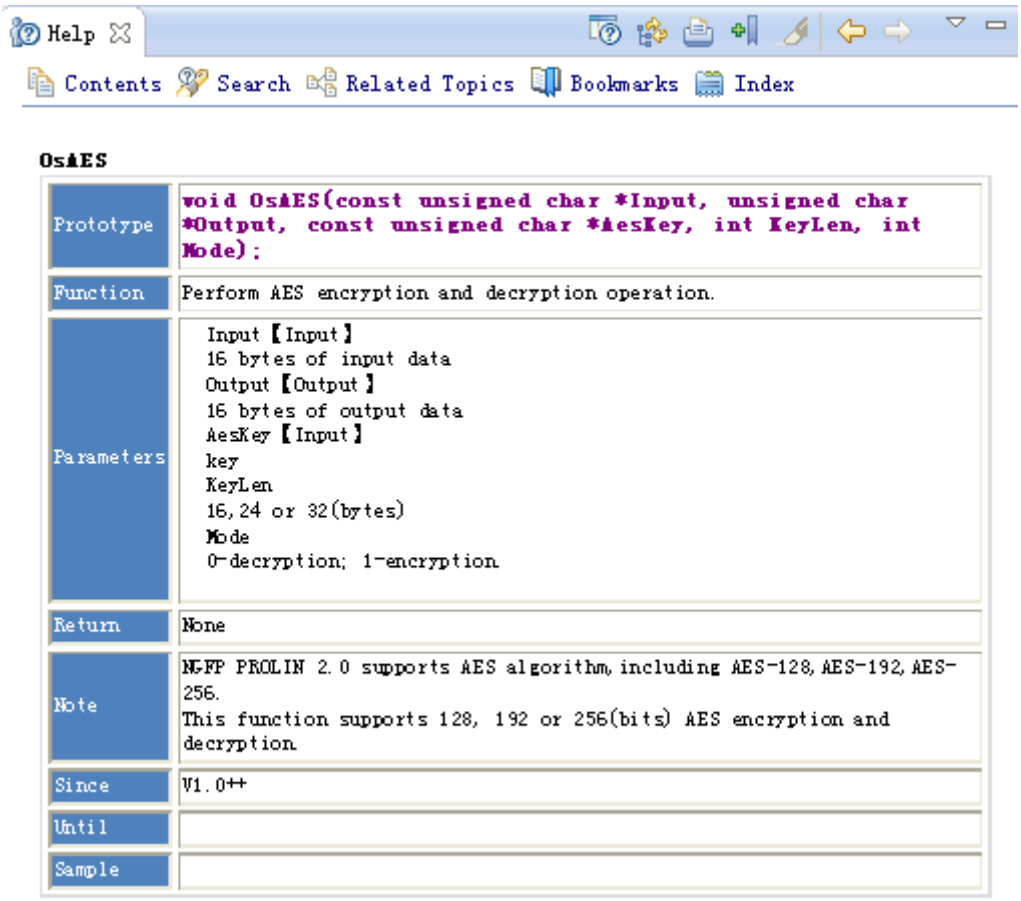

Figure 7.3 Function detail

7.3 Input Prompts

Input prefix of the word, then type Alt+/, then all the functions with the prefix will be shown (Figure 7.4).

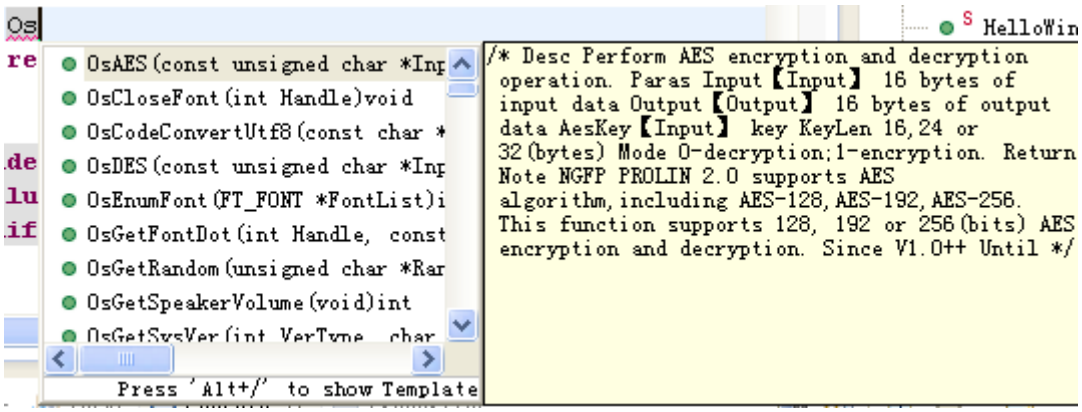

Figure 7.4 Input prompts

8 Emulator

The Prolin emulator is a Linux virtual machine which emulates the Prolin system environment. With the emulator, users can install the unsigned APP to have a quick view of how the APP runs. However, the emulator cannot simulate all the functions as a real machine. So, the final running result must be tested and verified on the real POS machine, including S300, S800, and S900.

For more information, please refer to **Prolin Emulator User Guide** manual.

8.1 Emulator functions

The following table lists what the emulator can do and what cannot.

Function	Support(YES) or not(NO)	Comment
Encryption and Decryption	YES	
LCD	YES	
Keyboard	YES	
Touch Screen	YES	For S300/S900
Signature board	YES	
Printer	YES	
Font	YES	
Encoding	YES	

Magnetic card	YES	
Network communication	YES	
File system	YES	
System message	YES	
Audio	YES	
PED	YES	
Serial port	YES	
Network Configuration	YES	Not include OsNetPing, PPPoE
IC card	NO	
RF card	NO	
Modem	NO	
GPRS /CDMA	NO	
WIFI	NO	
Power management	NO	

8.2 Open Emulator Manager

The emulator is integrated in the Prolin SDK. Please just check the SDK root directory (Figure 8.1).

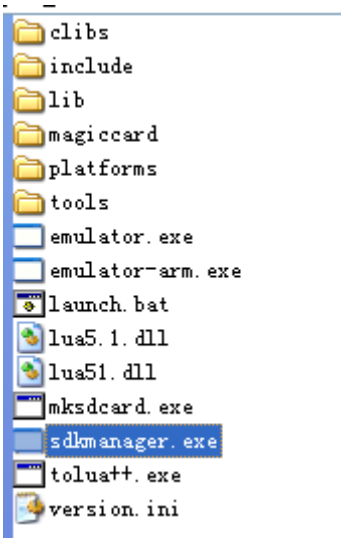

Figure 8.1 Emulator directory

You can directly open sdkmanager.exe. Or you can choose menu **PAX→Emulator Manager** or toolbar **Emulator Manager** from the SDK to open the emulator manager (Figure 8.2, Figure 8.3).

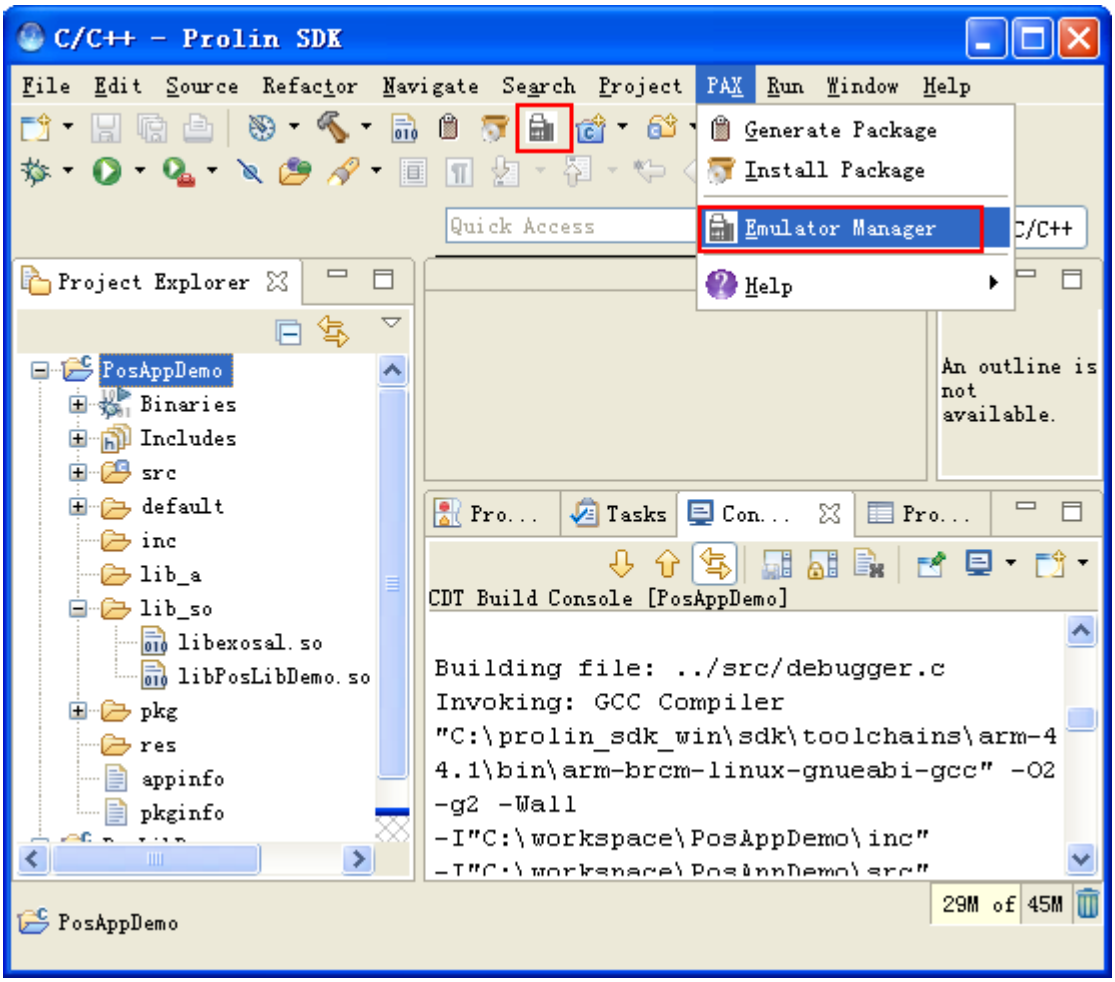

Figure 8.2 Open Emulator Manager

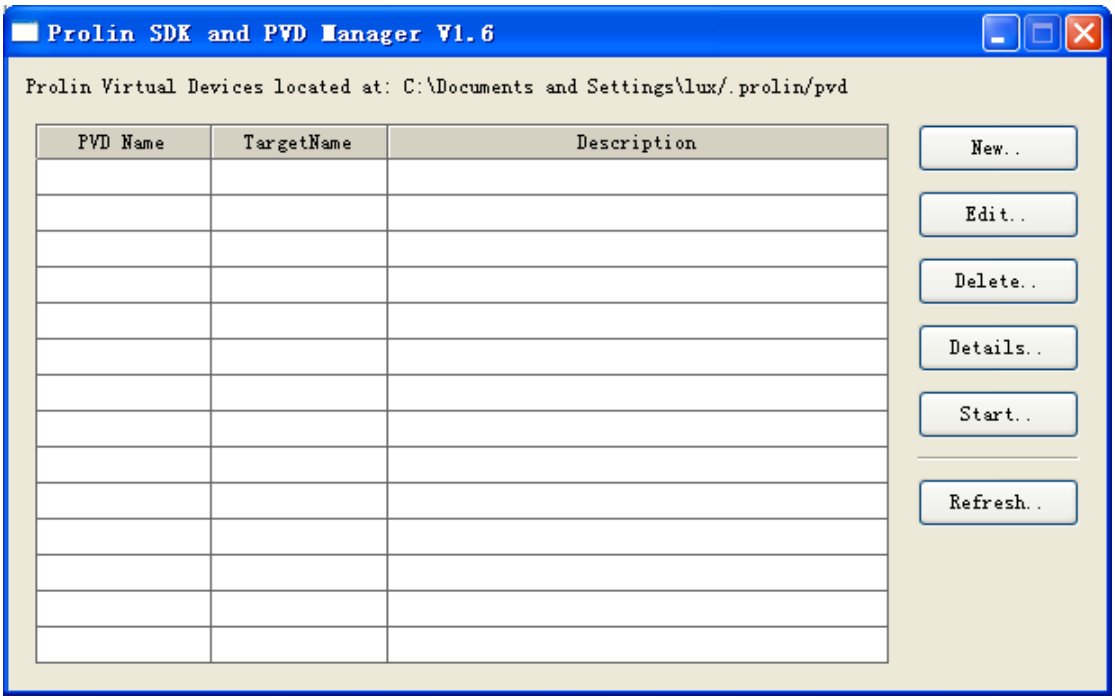

Figure 8.3 Emulator Manager

Click on **New** button to create a new PVD. In the creating page, input device name. Choose Target **prolin-2** and Built-in **POS-S800** (Figure 8.4).

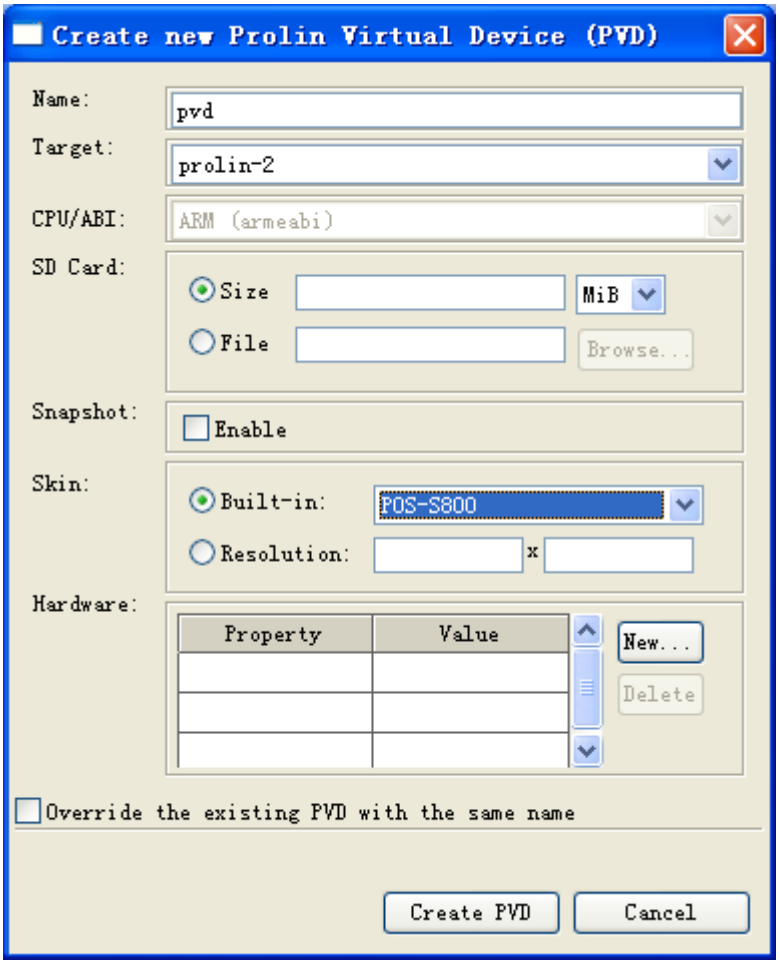

Figure 8.4 Set Emulator

After creating successfully, the device will be list in the **Prolin SDK and PVD Manager** page (Figure 8.5).

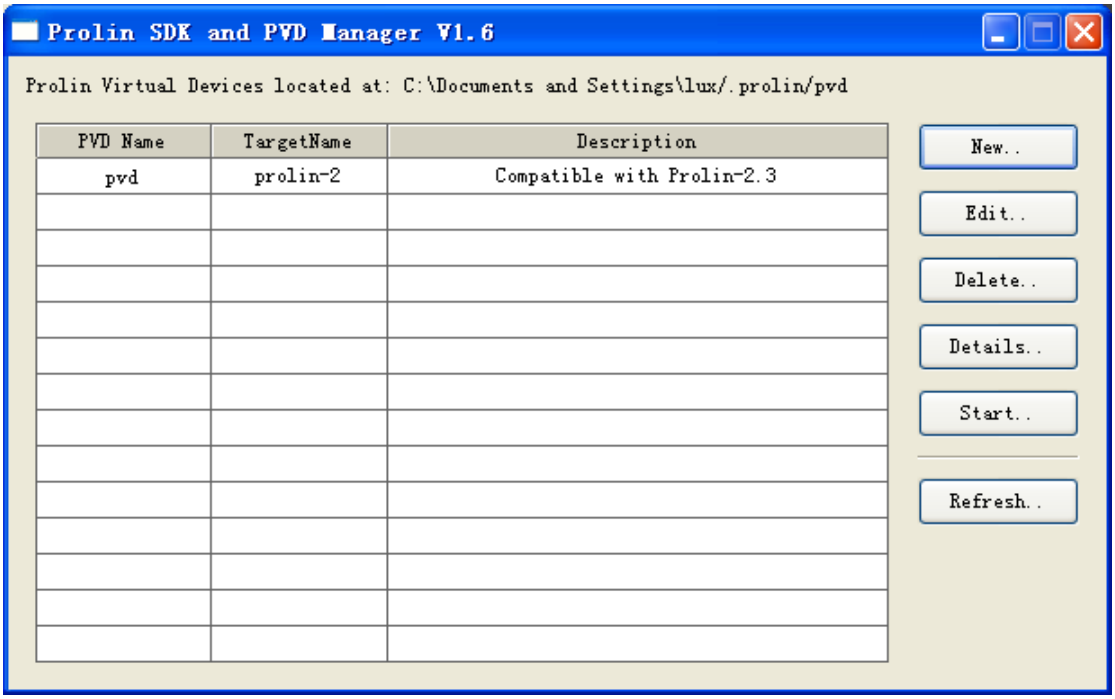

Figure 8.5 Emulator List

8.3 Open Emulator

In the **Prolin SDK and PVD Manager** page, choose **PVD** and click on **Start** button to start emulator (Figure 8.6).

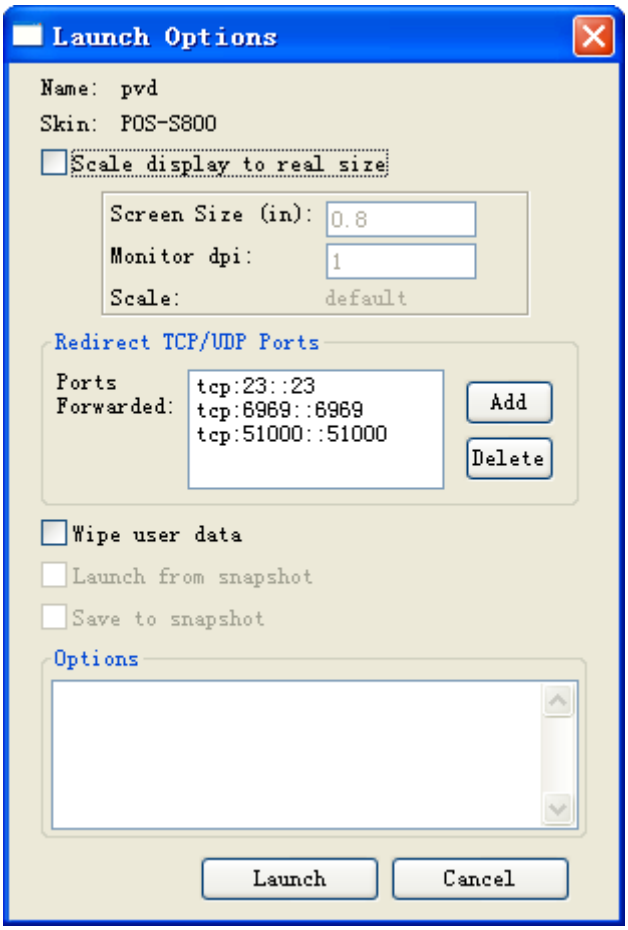

Figure 8.6 Launch options

In the **Launch Options** page, click on **Launch** button to start (Figure 8.7, Figure 8.8). If it is the first time to start up the emulator. You need to do some extra operation. For S800, you are asked to input the password. Type "**9876**" and click on the "Enter" button.

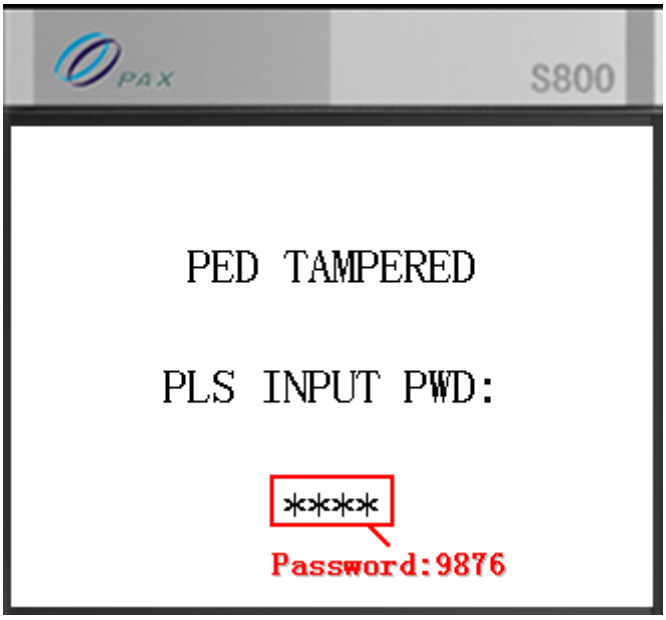

Figure 8.7 Emulator

Figure 8.8 Emulator

9 Logcat

You can use SDK to retrieve the log. It is very helpful when you develop the app. This chapter shows you how to write log in the source code and how to use logcat to show the log. So please make sure to know how to use it.

9.1 Add log

There are 2 main APIs, OsLogSetTag and OsLog. Use them as the following example (P9-1). Then reinstall the built package and run the app.

```
OsLogSetTag("PosAppDemo");
OsLog(LOG_DEBUG, "Prolin log debug, w=%d, h=%d", 320, 240);
OsLog(LOG_INFO, "Prolin log info");
OsLog(LOG_WARN, "Prolin log warn");
OsLog(LOG_ERROR, "Prolin log error");
```

Figure 9.1 Add log

9.2 Show Log

Run the app with log code. Then from Eclipse, click on menu **Window→ Open Perspective→ Other…**-> **DDMS**. The DDMS perspective will be open. In this perspective, choose a connected device and its log displays (Figure 9.2).

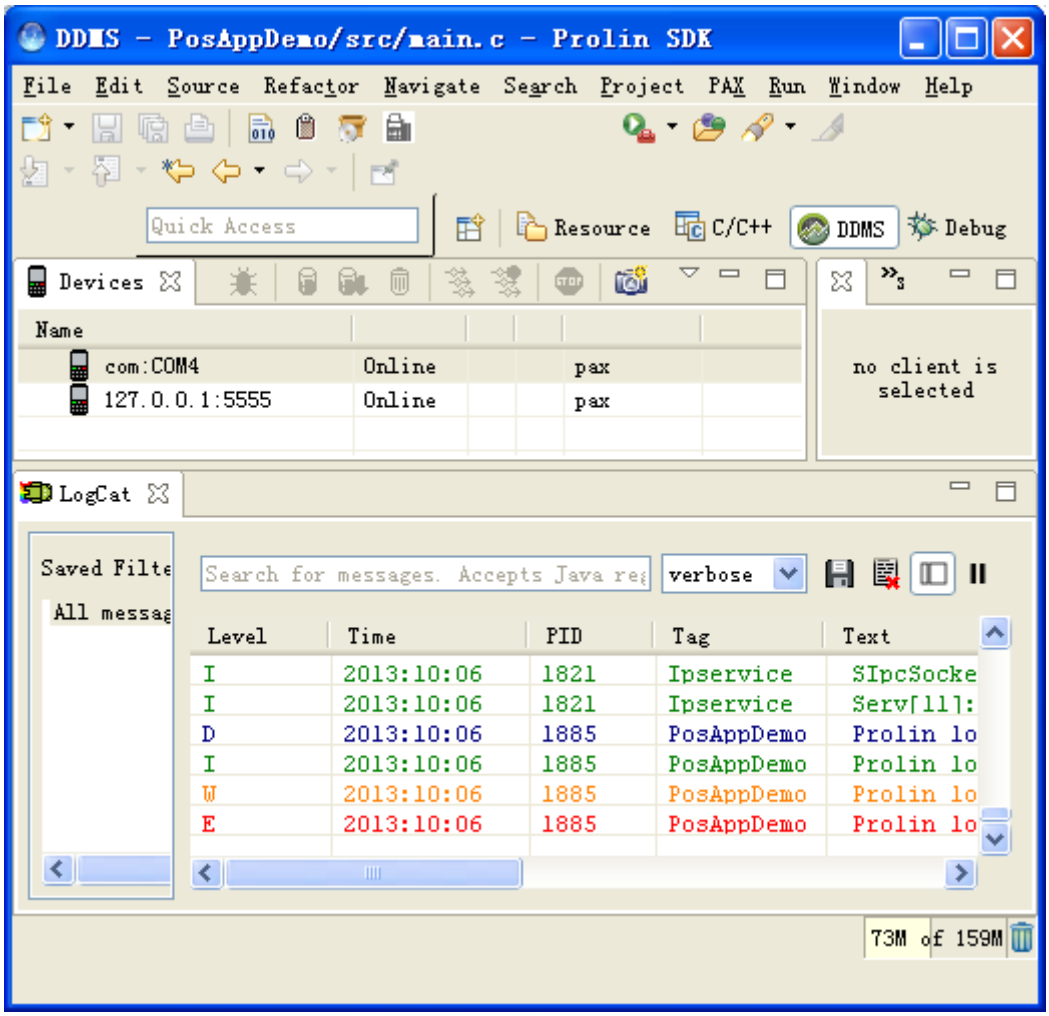

Figure 9.2 Logcat view

10 Debug

Debugging code by emulator and physical machine is supported. If you want to debug your source code please read this chapter carefully. Also, as always, feel free to ask questions to Prolin support team or you can ask me during lab hours.

10.1 Preparation

Before debugging, you should toggle breakpoint to source code in order to run single-step debug. Then generate and install aip package, and open the emulator or physical machine.

10.1.1 Set Debug Content

Add **start_debug** function in the program entry (Figure 10.1).

```
int main()
{
    start_debug();
    Init();
    HelloWorld();
    Finish();
    return 0;
}
```

Figure 10.1 Add debug code

10.1.2 Toggle Breakpoint

Select the **Toggle Breakpoint** command to create a new breakpoint at the given location (Figure 10.2, Figure 10.3).

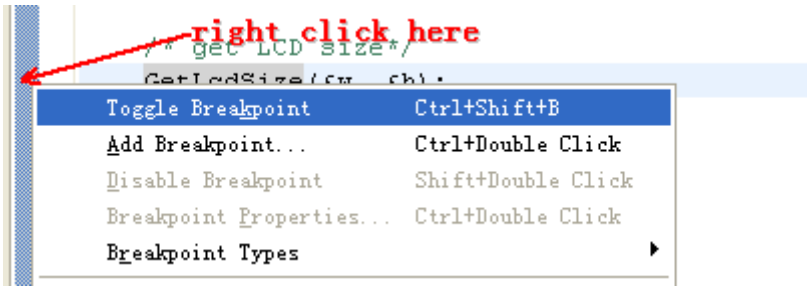

Figure 10.2 Toggle Breakpoint

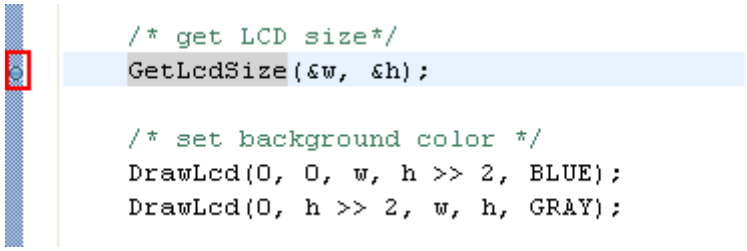

Figure 10.3 Breakpoint

NOTE

You can double click to add/remove breakpoints (Figure 10.3).

10.1.3 Build and Install APP

After successfully generating the .aip file, you can install the APP by choosing menu **PAX→Install Package.** Start and run the app. Now app will halt at the position of the first breakpoint, and at this time you cannot operate the device (Figure 10.4).

Figure 10.4 App halts

10.2 Debug Steps

10.2.1 Open Debug Configurations Page

When the APP halt to wait for debugging, Open **Debug Configurations** page from the toolbar or context menu or system menu **Run→Debug Configurations** (Figure 10.5, Figure 10.6, and Figure 10.7).

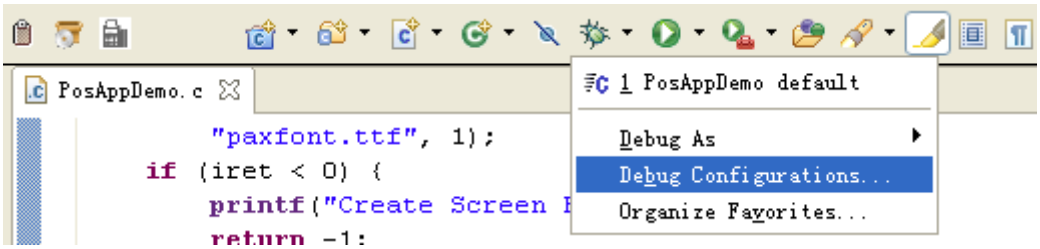

Figure 10.5 Open Debug Configurations Page

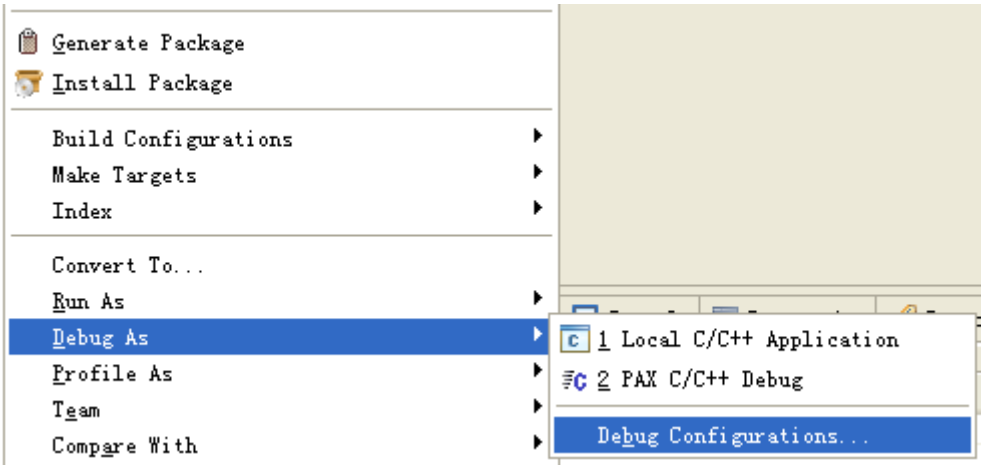

Figure 10.6 Open Debug Configurations Page

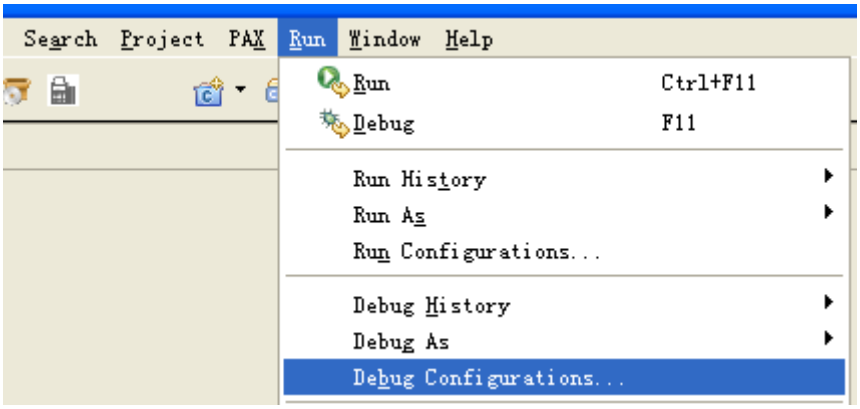

Figure 10.7 Open Debug Configurations Page

10.2.2 Debug Configuration Settings

From **Debug Configuration** page, choose **PAX Debug Launch** and double click on it or right click on it to choose **New** (Figure 10.8).

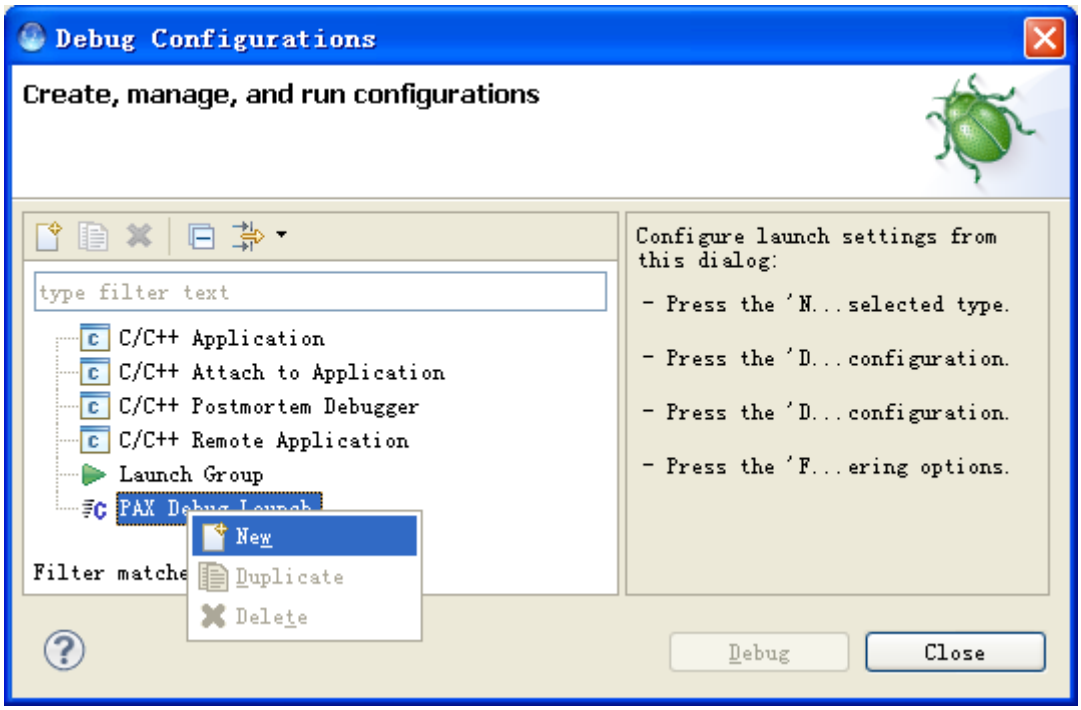

Figure 10.8 Create a new Debug

A new debug configuration instance is created. It will connect with the emulator or the physical machine by network ore serial port (P9-12).

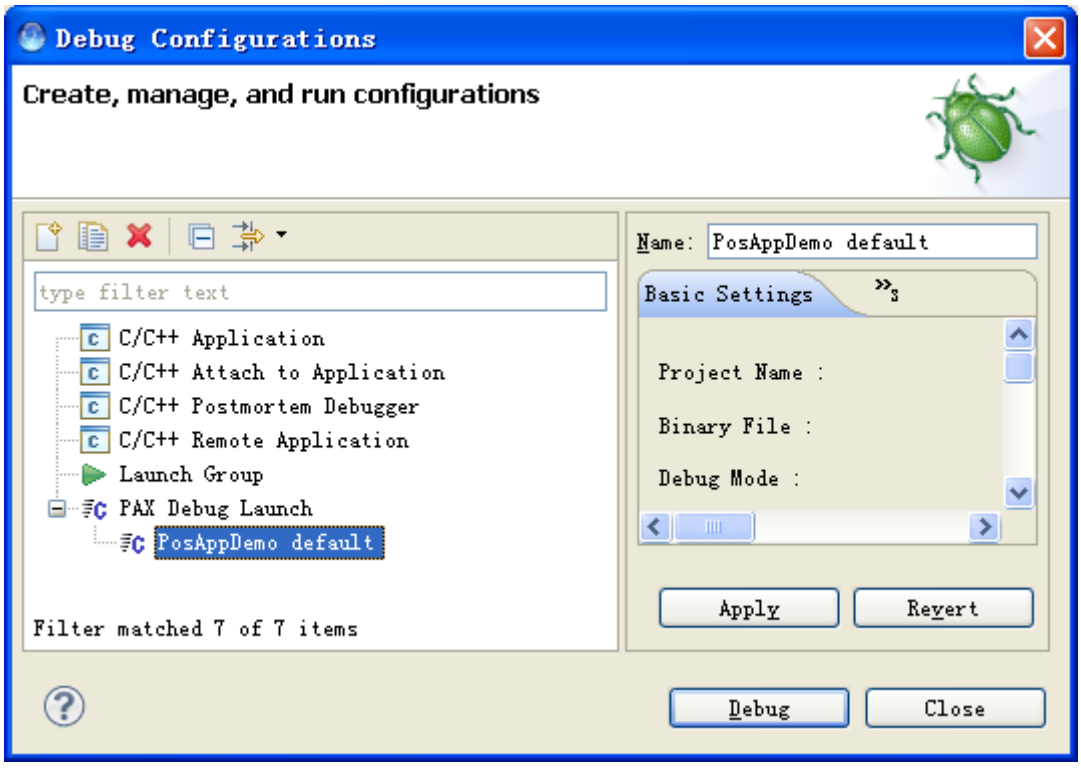

Figure 10.9 Debug Setting

Choose device you want to debug (Figure 10.10). Click on **Generate Commands** button.

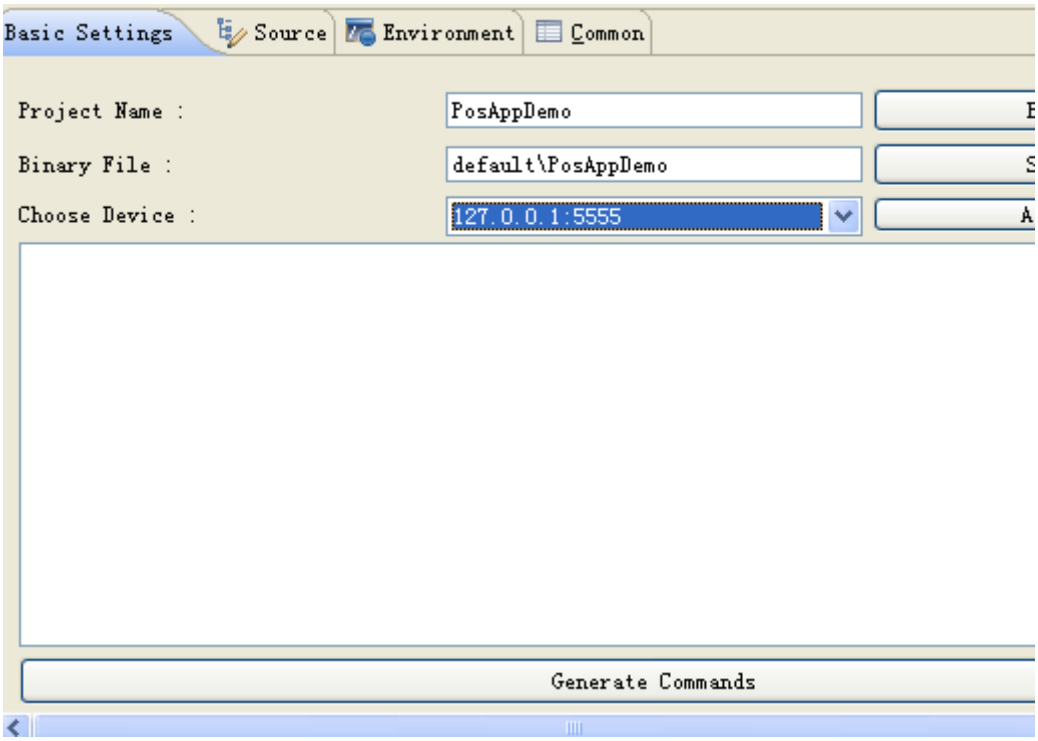

Figure 10.10 Choose Device

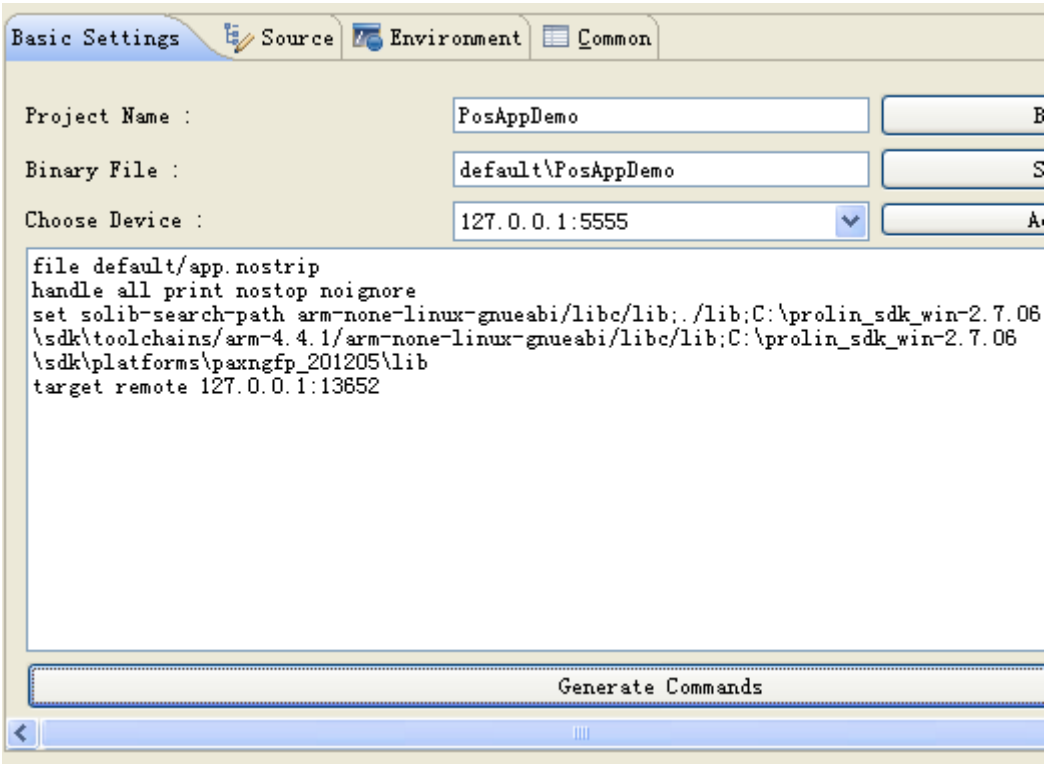

Figure 10.11 Generate Commands

10.2.3 Run debug

Click on **Debug** button to start (Figure 10.12). See the console area below, the GDB is connecting.

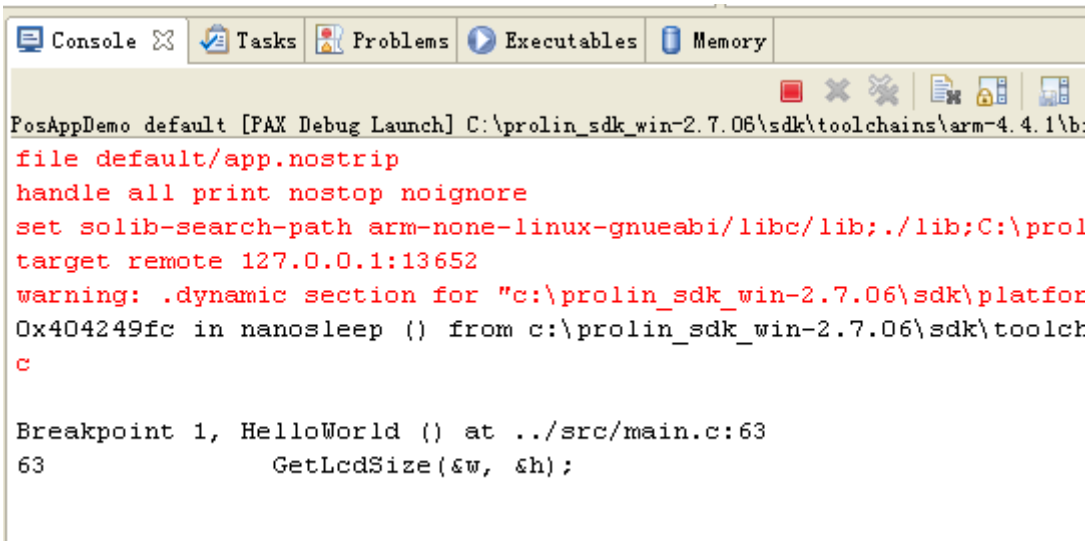

Figure 10.12 Console message

When GDB is connected, you are required to switch to the **Debug perspective** (Figure 10.13).

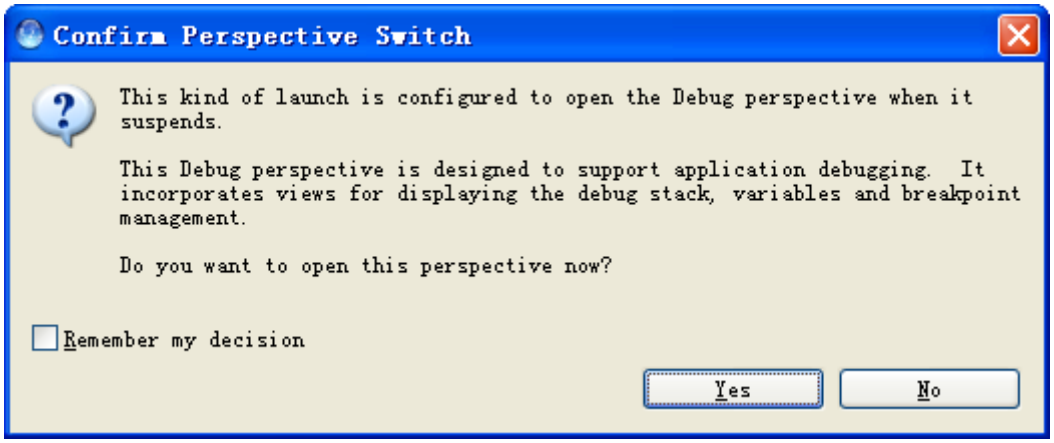

Figure 10.13 Confirm switch

Choose **Yes**, the debug perspective then opens. And the program breaks execution at the first break point location (Figure 10.14).

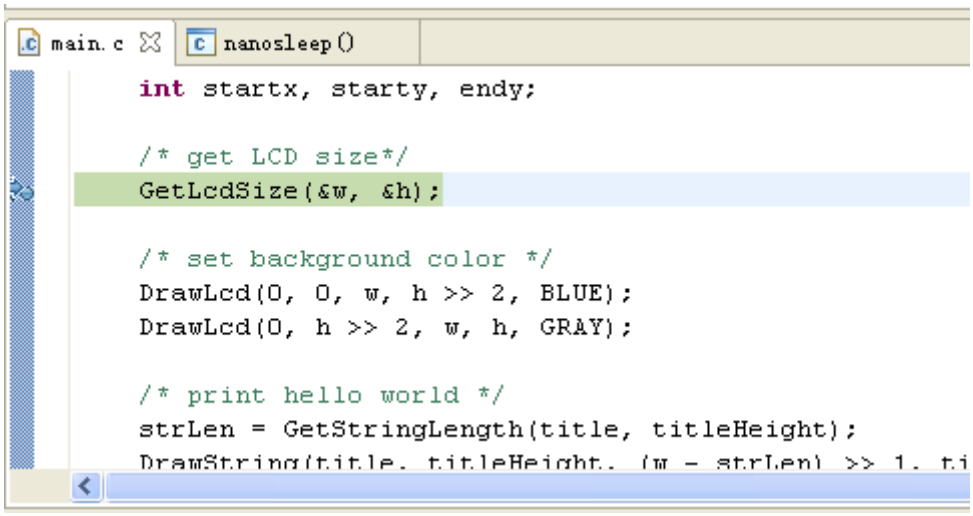

Figure 10.14 Debug perspective

Now, you can step-through, check variable value, etc (Figure 10.15).

Figure 10.15 Debug Perspective

10.2.4 Debug Note

The SDK cannot run multiple GDBs (Figure 10.16). So every time you start a new debug, please switch to the Debug perspective and kill all the last debugging process (Figure 10.17). If it difficult to terminate the process, you can just restart the SDK.

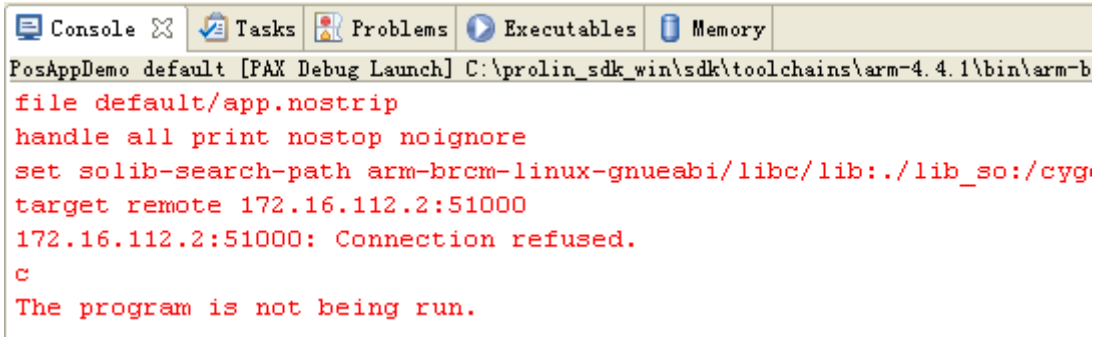

Figure 10.16 Debug error info

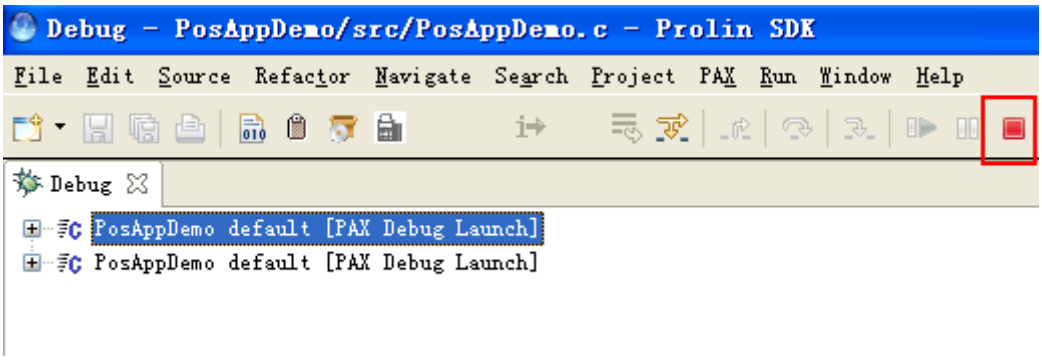

Figure 10.17 Multiple GDBs

11 User Help

Choose menu **PAX→Help** (Figure 11.1). It has 4 sub menus: About SDK, License Info, Feedback and Update SDK.

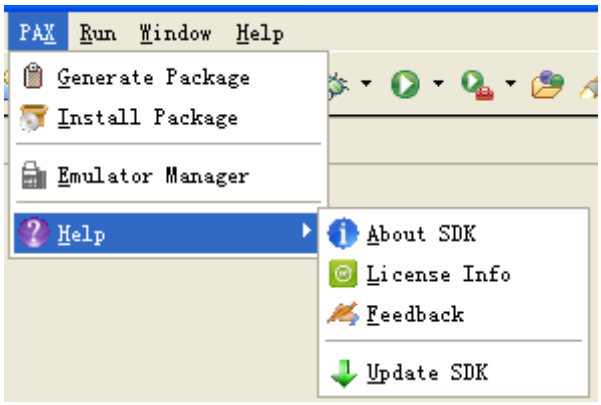

Figure 11.1 PAX Help Menu

11.1 About SDK

Click on sub menu **About SDK**. **About PAX SDK** dialog will be opened (Figure 11.2).

Figure 11.2 About SDK

11.2 License Info

Click on sub menu **License Info**. **Your License Info** dialog will be opened (Figure 11.3).

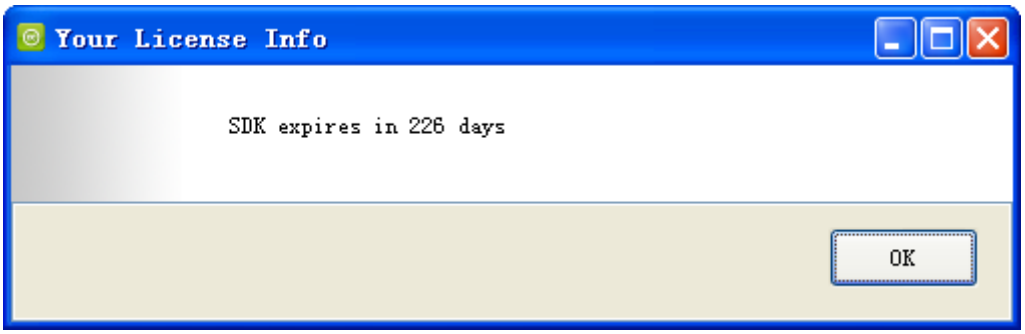

Figure 11.3 License info

11.3 Feedback

Click on sub menu **Feedback**. An e-mail template will be open. You can send your request to stakeholders.

12 Update SDK

You can do SDK online update.

12.1 Update Wizard

Choose menu **PAX→Help→Update SDK**, and then update wizard window will popup (Figure 12.1).

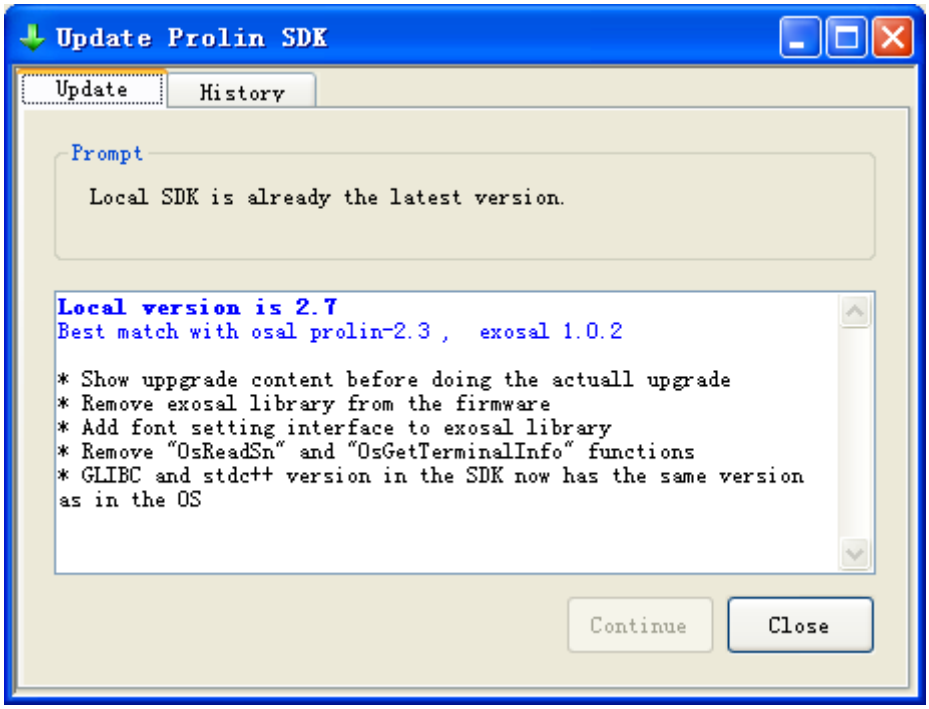

Figure 12.1 Update wizard window

There are 2 sub page of the wizard window. The first page is **update** wizard. It will show current SDK version, best matched libraries. If the current SDK is the latest release, then the button **Continue** is invalid. If a newer version is checked, then the button **Continue** is valid.

Click on the button **History** to switch to the **History** page. Here you can see all the update history record (Figure 12.2).

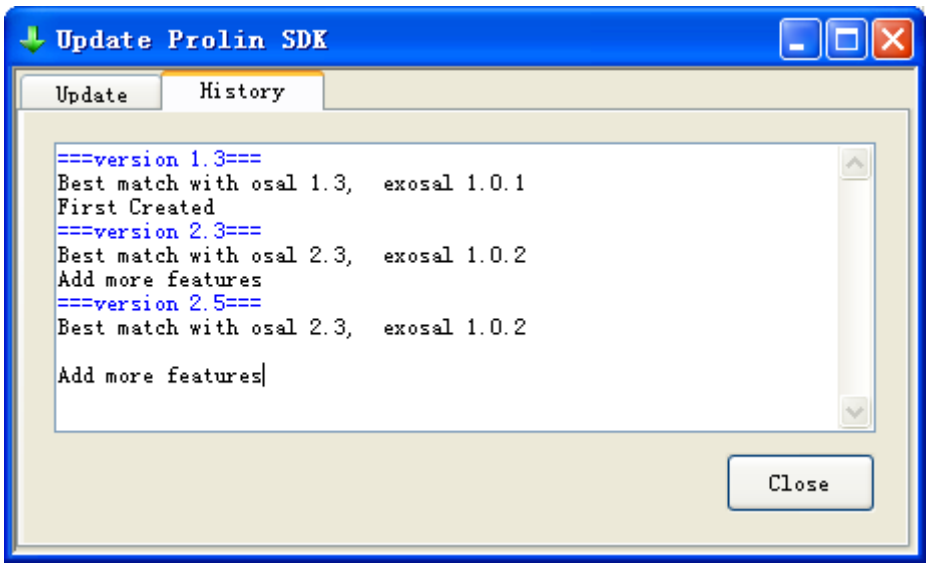

Figure 12.2 Update history

If there is new version checked, then Continue button is valid. Click on the button and goto the next Updater page (Figure 12.3).

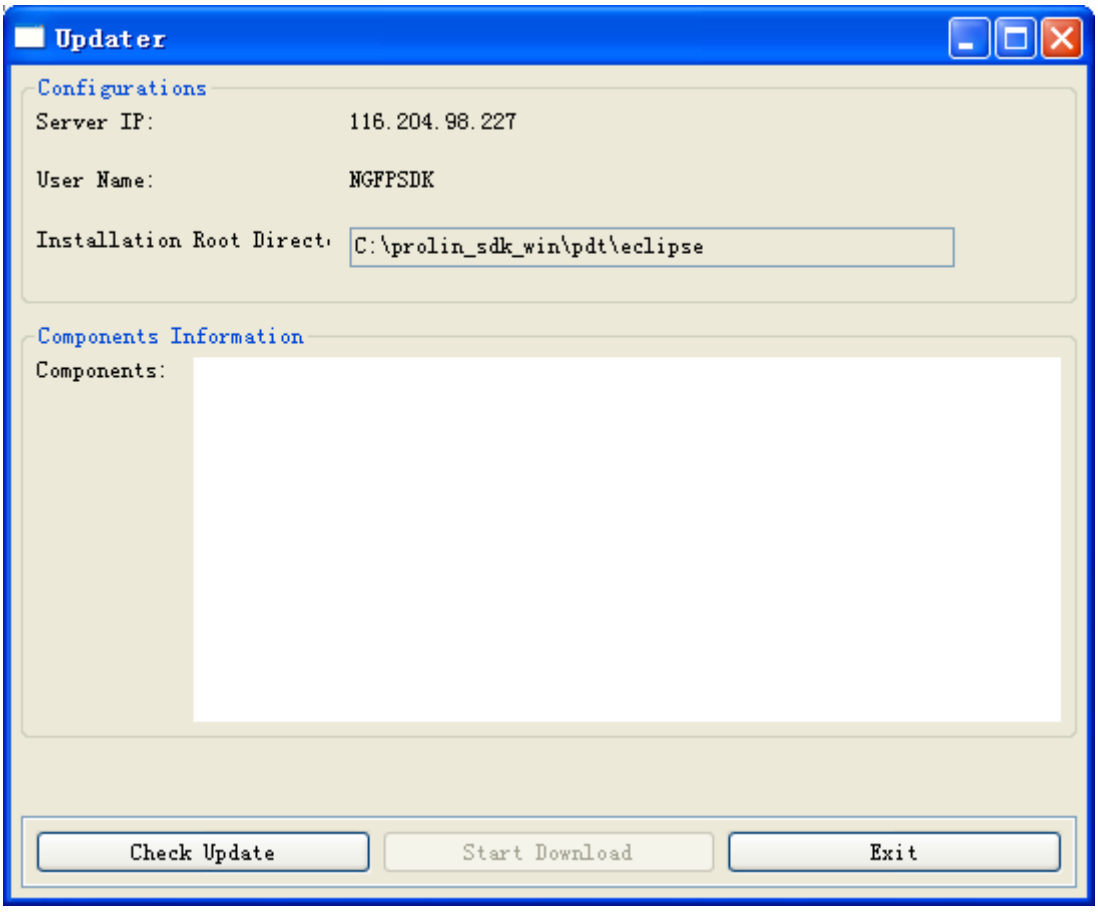

Figure 12.3 Open Updater

12.2 Check Update

Click on button **Check Update** to start. The checking result will show in **Components Information** (Figure 12.4).

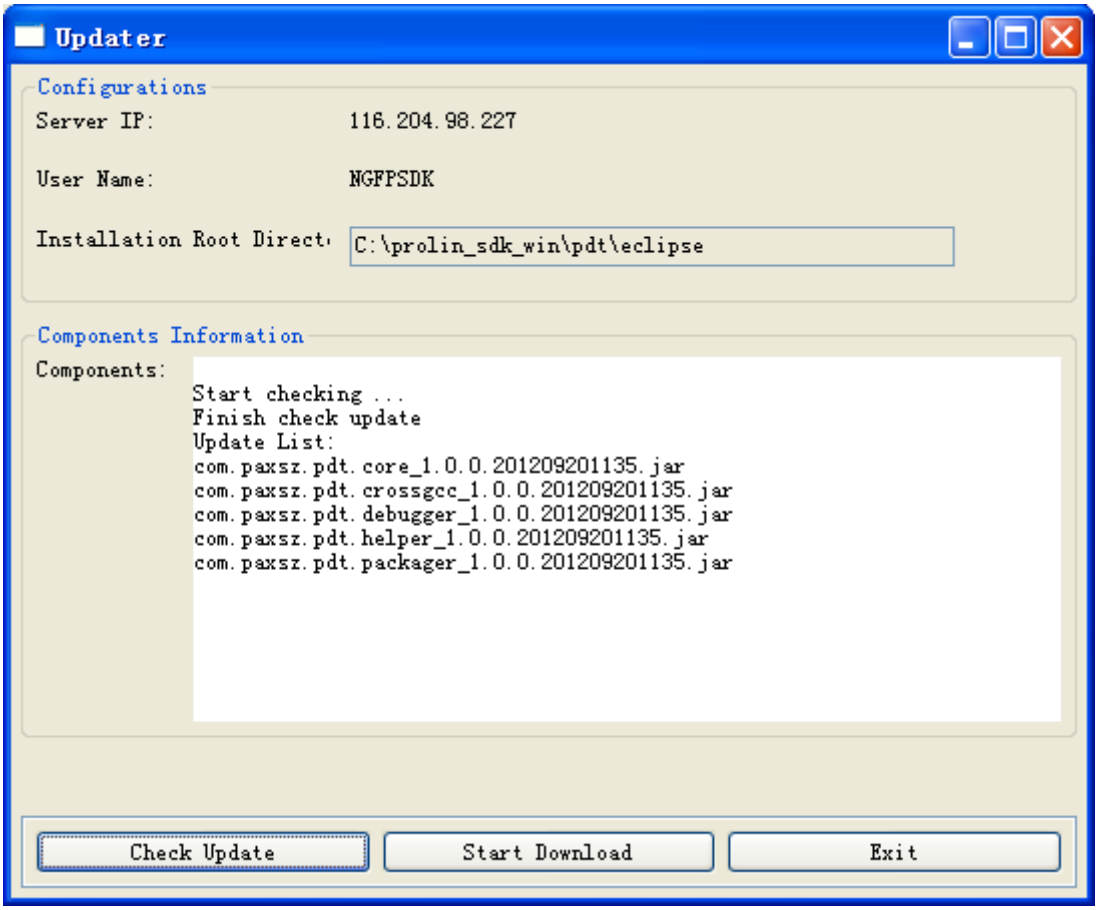

Figure 12.4 Check Update

12.3 Download Update Data

After checking, if there are components to download, then button **Start Download** will be enable. Click on the button to start (Figure 12.5).

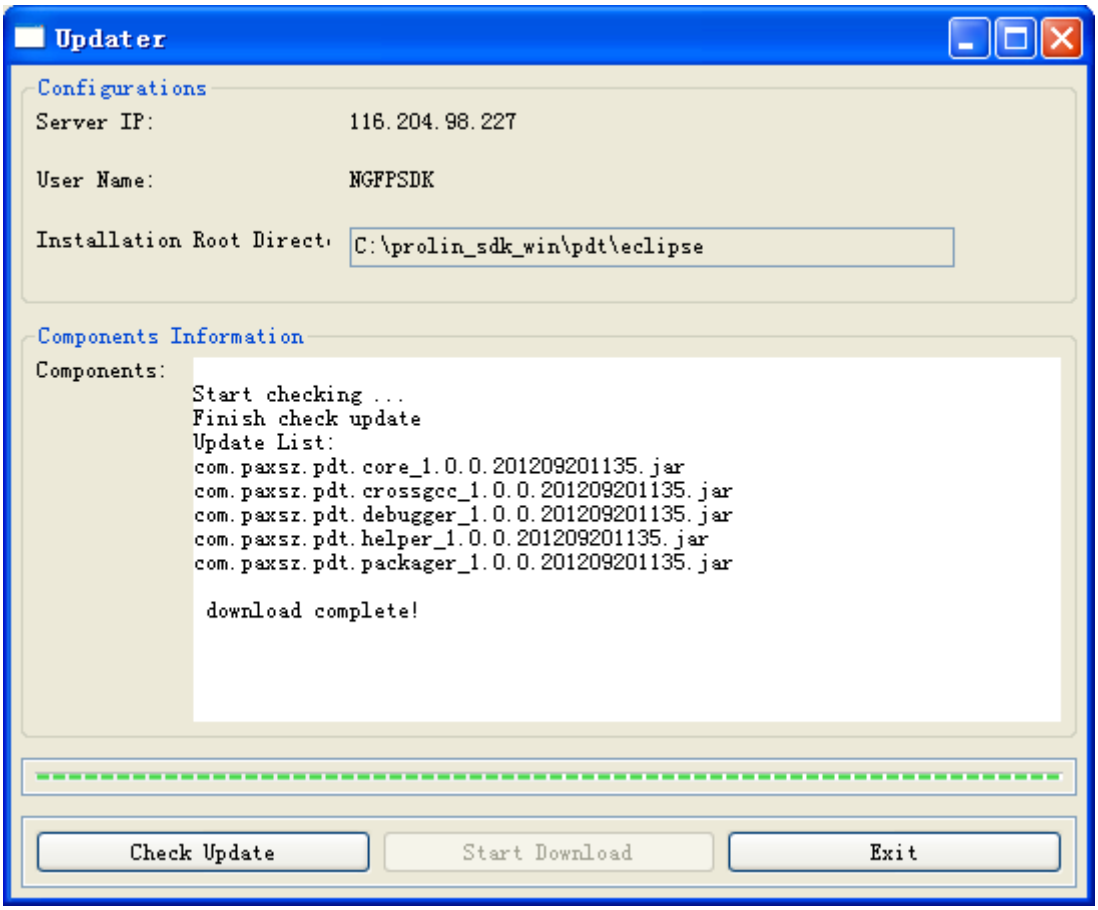

Figure 12.5 Download Update Data

Download result will be shown in **Components Information**.

13 Customize

13.1 Customize Editor

13.1.1 Set Editor Color

Choose menu **Windows→Preferences→C/C++→Editor→Syntax Coloring** (Figure 13.1).

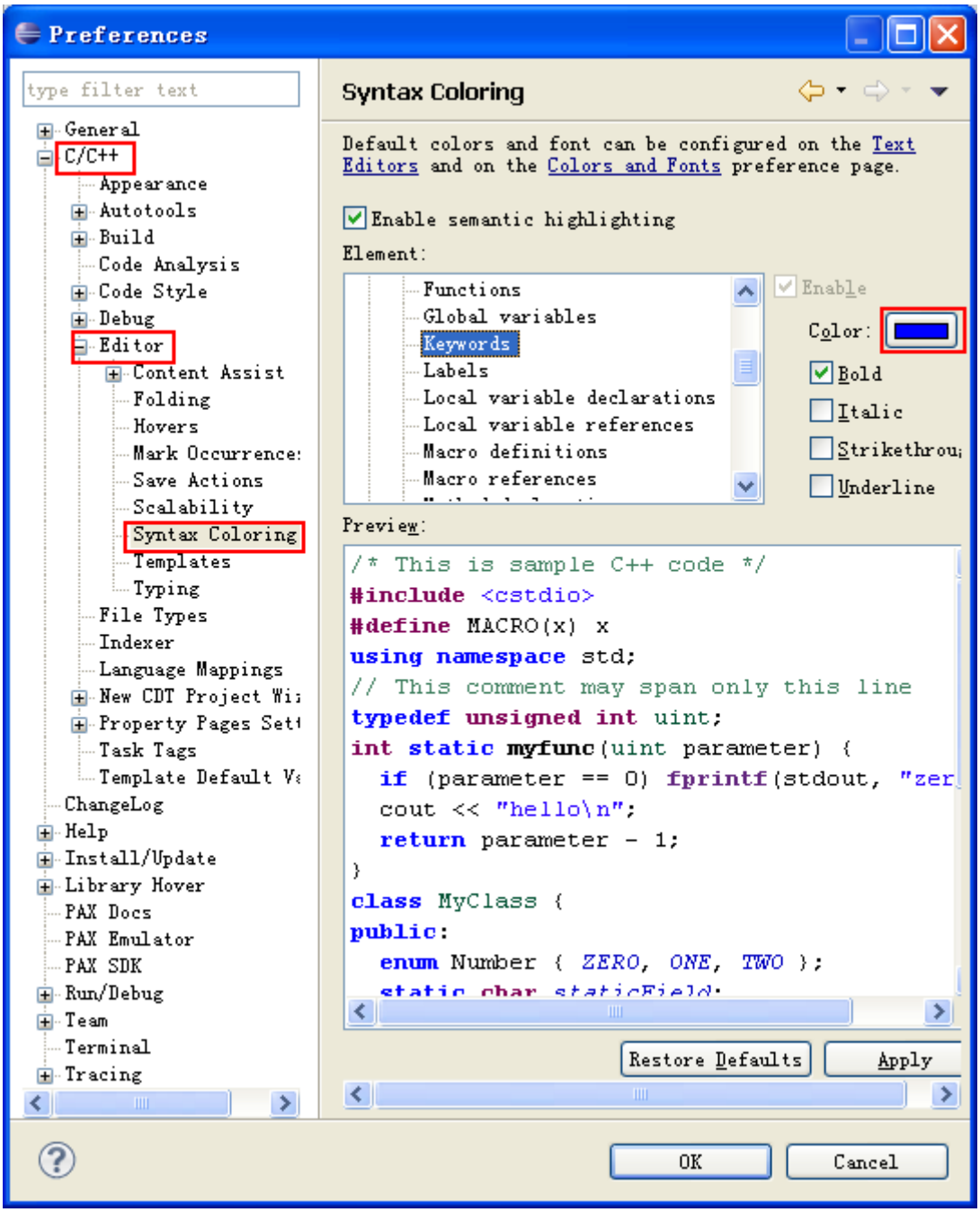

Figure 13.1 Set Code Color

13.1.2 Set Code Font

Choose menu **Windows→Preferences→General→Appearance→Colors and Fonts**. In the **Colors and Fonts** page, choose **C/C++→Editor→C/C++ Editor Text Font….** Click on button **Edit** to set font (Figure 13.2).

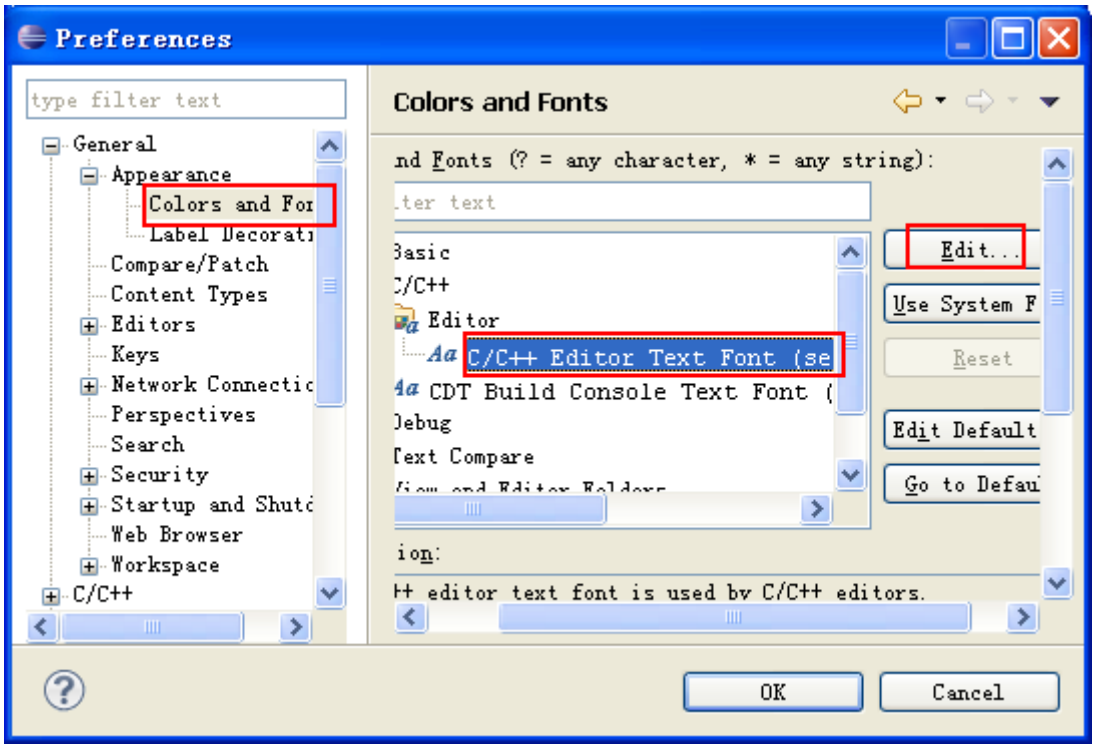

Figure 13.2 Set Code Font

13.1.3 Open/Close Code Auto-prompts

Choose menu **Windows→Preferences→C/C++→Editor→Hovers**. In the **Hovers** page, make the **Combined Hover** list be checked or unchecked to open or close auto-prompts (Figure 13.3).

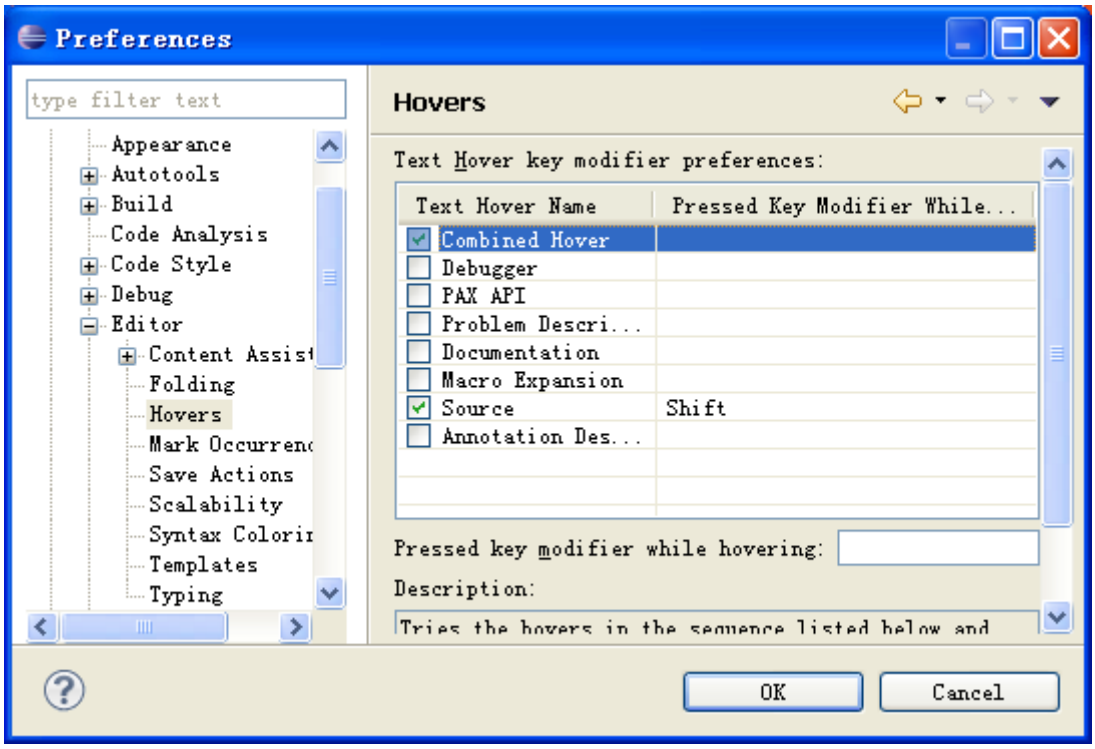

Figure 13.3 Set Code Auto-prompts

13.1.4 Show Line Number

Choose menu **Windows→Preferences→General→Editors→Text Editors**, and then tick **Show line numbers** check-box (Figure 13.4).

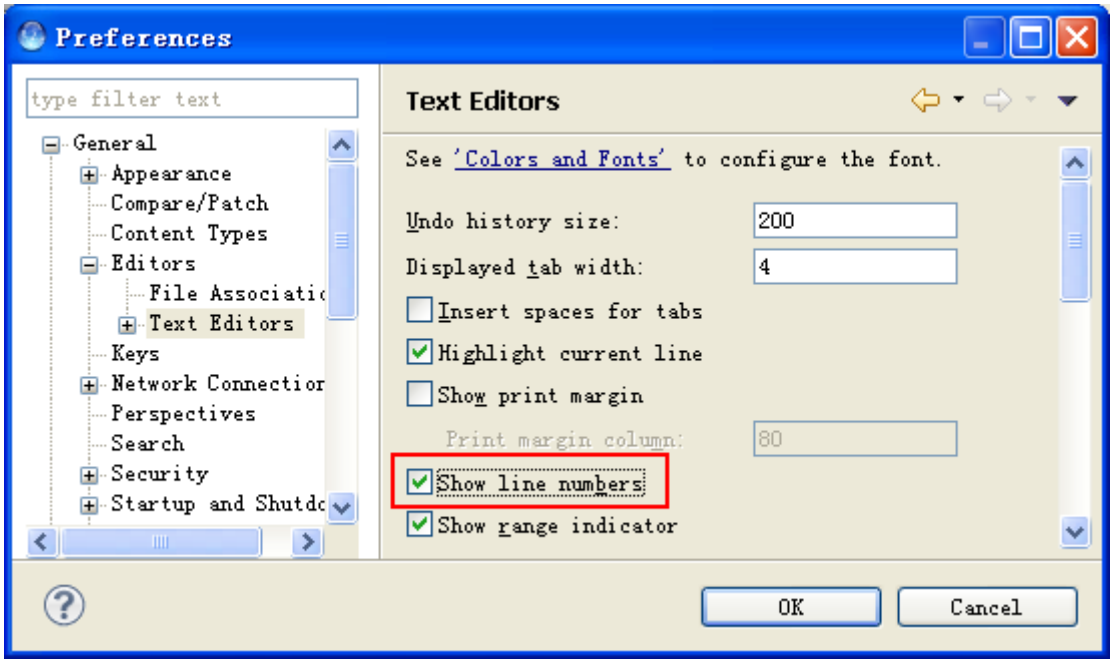

Figure 13.4 Show Line Number

13.2 Customize Perspectives

You can customize perspectives such as **Tool Bar Visibility**, **Menu Visibility**, **Shortcuts**, etc. The following are detailed steps.

13.2.1 Open Customize Page

Choose Menu **Windows**, you will see several perspectives operations that you can easily understand (Figure 13.5).

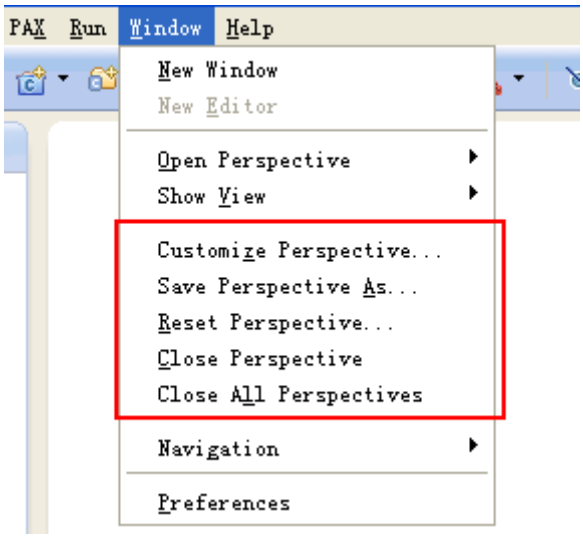

Figure 13.5 Customize Menu

Click on **Customize Perspective** sub menu, and then customize page will open (Figure 13.6).

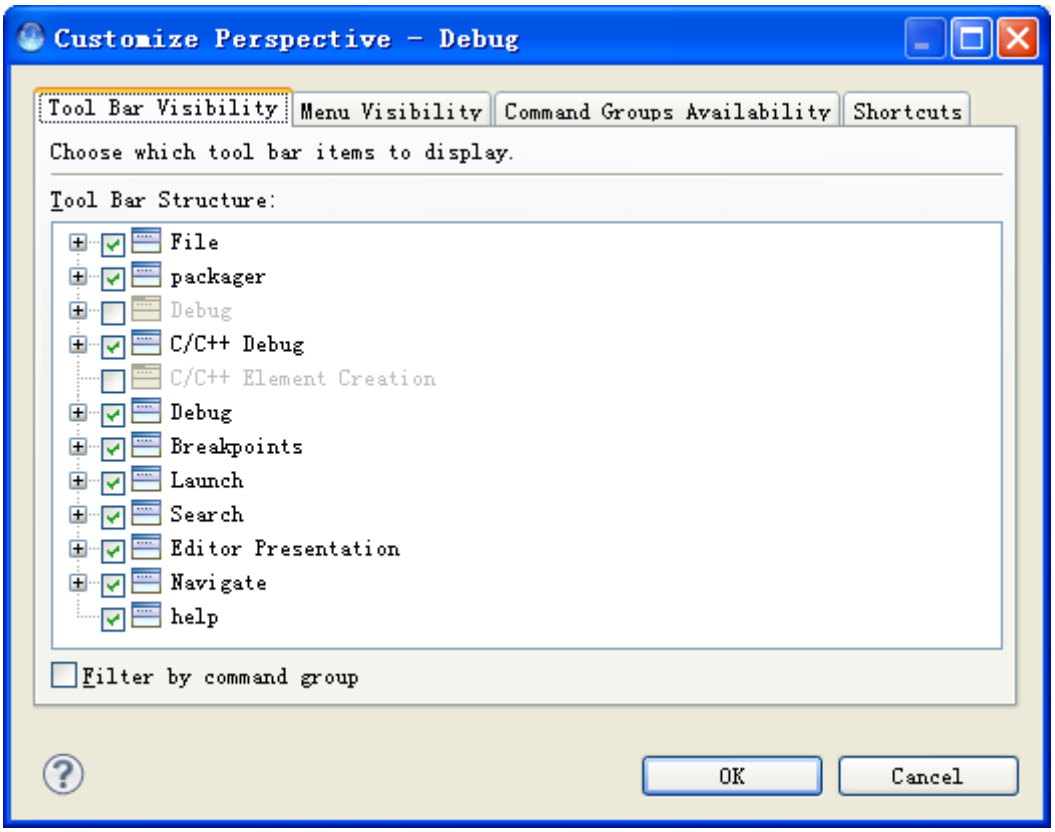

Figure 13.6 Customize Page

13.2.2 Set Toolbar

Choose Toolbar **Bar Visibility** sub page, then set it as you wish (Figure 13.7).

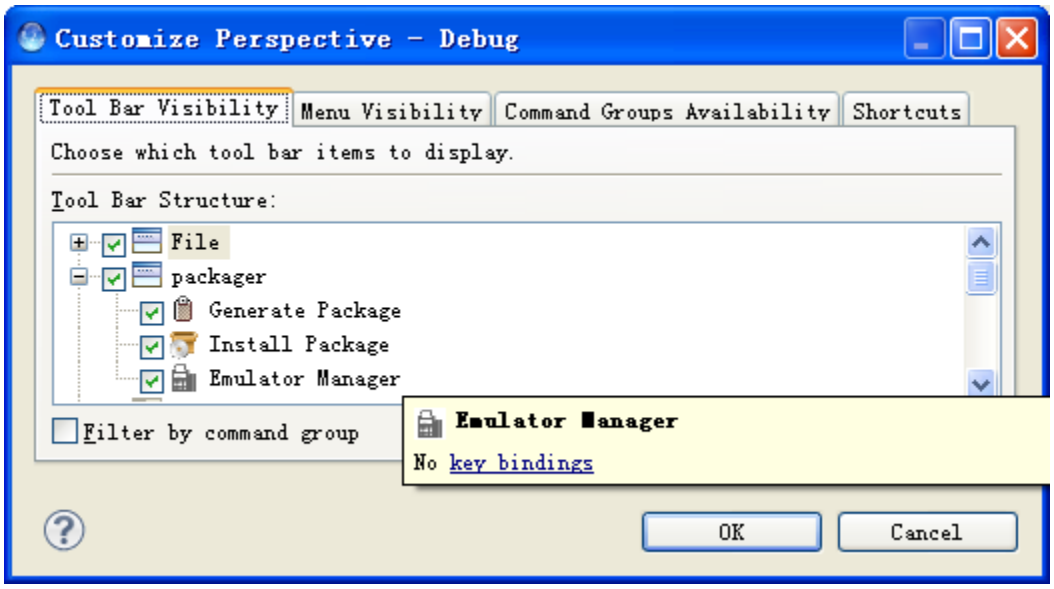

Figure 13.7 Customize Toolbar

13.2.3 Set Menu

Choose **Menu Visibility** sub page, then set menu as you wish (Figure 13.8).

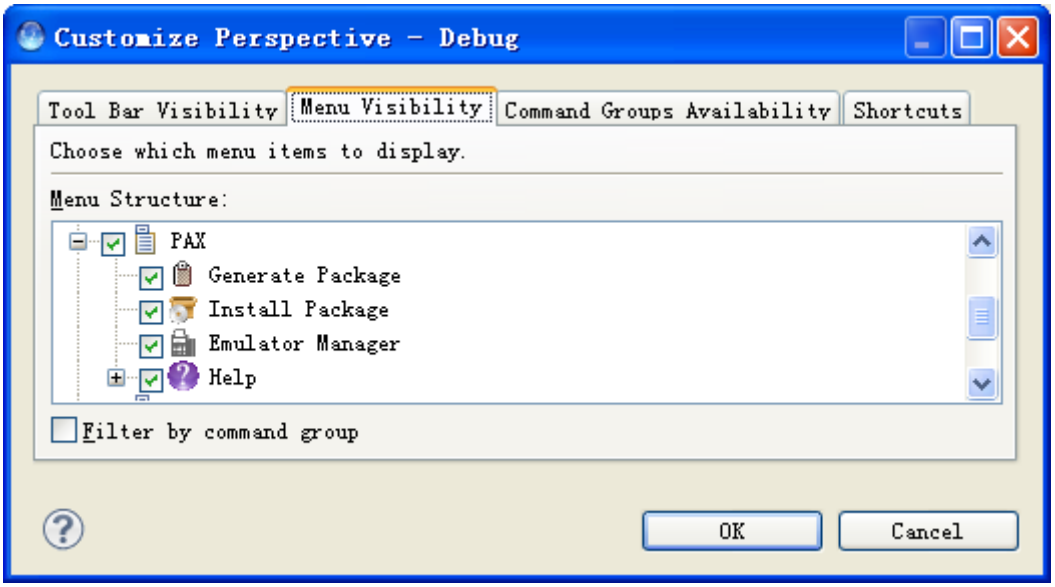

Figure 13.8 Customize Menu

14 Keyboard Shortcuts

Ctrl+D: Delete current line.

Ctrl+Alt+↑(↓): Copy current line to the previous (next) line

Alt+↑(↓): Move the current line to the previous(next) line.

Shift+Enter: Add a new line to the next line.

Ctrl+Shift+Enter: Add a new line to the previous line.

Ctrl+L: Jump to the specific line number.

Ctrl+Shift+F: Format the code of current file.

Prolin SDK Tutorial

PAX Technology Limited
www.pax.com.hk

Hong Kong
Room 2416, 24/F, Sun Hung Kai Centre, 30 Harbour Road,
Wanchai, Hong Kong
Tel: +852-25888800
Fax: +852-28023300

Shenzhen
4/F, No.3 Building, Software Park, Second Central Science-Tech Road,
High-Tech Industrial Park, Shenzhen, Guangdong 518057, P.R. China
Tel: +86-755-86169630
Fax: +86-755-86169634