AMPC™ IDE

USER MANUAL

THE USER MANUAL FOR AMPC™ IDE

PART I: INTRODUCTION

1. **Introduction to Axiomatic Multi-Platform C (AMPC™)**
2. **Components of AMPC™**

1. INTRODUCTION TO AXIOMATIC MULTI PLATFORM C (AMPC™)

1.1 WHAT IS AMPC™?

AMPC™ is an Integrated Development Environment (IDE) for the C programming language, which generates Java byte-code for rapid development of applications. The resulting application software will be able to run on any JVM enabled device.

AMPC™ is based upon American National Standards Institute C (ANSI C), X3.159-1989. This allows users of **AMPC™** to develop software using the standard C programming language and run the executables on JVM enabled devices requiring no knowledge of the Java language.

Examples of JVM enabled devices are PDAs, cell-phones, game consoles and desktop systems.

The Java class file generated by **AMPC™** is in full conformance with Sun's Java Virtual Machine Specification Second Edition (Java 2 Platform)

AMPC™ follows the ANSI C standard, and supports the run-time library for C applications.

AMPC™ is available for the following platforms:

- Windows on x86 PCs
- Linux OS on x86 PCs
- Mac OS X 10.4 on Macintosh Power PC G4.

1.2 SYSTEM REQUIREMENTS

Before installing **AMPC™**, please make sure that your computer meets the following minimum requirements:

1.2.1 For Linux OS Platform

- Intel x86 based processor or compatible.
- 64 megabytes of RAM minimum. It is possible to run the application with less RAM but this is not advisable. It is recommended to have 256MB of RAM.
- CD-ROM drive (for installation from CDs)
- At least 50MB of hard disk space for installation.

1.2.2 For Mac OS X Platform

- PowerPC based processor. Recommended PowerPC G4 and above

- At least 128MB of RAM. Recommended 256MB
- CD-ROM Drive
- At least 150MB of disk space.
- Mac OS X 10.4 Operating System or above.
- JRE/J2SDK 1.4 or above.
- If you want to use AMPCGUI (build using GTK) you are required to have:
    i. X11 for Mac OS X
    ii. Xcode tools for Mac OS X

1.2.3 For Microsoft Windows Platform

- Intel x86 based processor or compatible.
- 128 megabytes of RAM minimum. It is possible to run the application with less RAM but this is not advisable. It is recommended to have 256MB of RAM.
- CD-ROM drive (for installation from CDs)
- At least 150MB of hard disk space for installation.
- The minimum version of JDK needed for **AMPC™** is JDK 1.4.2 and JDK 5.0 is recommended.
- The latest version, JDK 5.0 Update 3 can be downloaded from https://java.sun.com/j2se/1.5.0/download.jsp. The minimum drive space for JDK 5.0 installation is about 132 MB.

1.3 COMPILATION AND INSTALLATION

User compilation is not necessary as **AMPC™** will be distributed with a binary installer.

1.3.1 Linux x86 Platform

ampc-linux-1.0.tar.gz file can be downloaded at http://www.axiomsol.com. Once downloaded, the compressed file will have to be unpacked using the following commands;

- tar or extract ampc-linux-1.0.tar.gz

```
% tar -xvzf ampc-linux-1.0.tar.gz
```

- After extracting all the compressed files, you can start the installation process.
- To install, go to directory ampc-linux-1.0

```
% cd ampc-linux-1.0
```

- Then execute the installation program by issuing the command;

```
% ./install.sh
```

- To uninstall, execute the following command.

```
% ./uninstall.sh
```

- The install script will install the relevant files onto your user's home directory and set the correct permissions on the files so users can access and run **AMPC™**.
- During the installation process, users have to fill in some information that is required by the system.
- Once the disclaimer as below is displayed, users have to respond to the question. If you the installation will proceed to the next step.

```
DISCLAIMER OF WARRANTY

======================

THIS SOFTWARE IS PROVIDED BY AXIOMATIC SOLUTIONS SDN BHD ``AS IS'' AND ANY
EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COMPANY BE LIABLE FOR ANY DIRECT,
INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.


LIMITATION OF LIABILITY

=======================

Licensee acknowledges that the Licensed Software may have defects or
deficiencies which cannot or will not be corrected by Axiomatic. Licensee
will hold Axiomatic harmless from any claims based on Licensee's use of
the Licensed Software for any purposes, and from any claims that later
versions or releases of any Licensed Software furnished to Licensee are
incompatible with the Licensed Software provided to Licensee under this
Agreement.


Licensee shall have the sole responsibility to protect adequately and
backup Licensee's data and/or equipment used in connection with the
Licensed Software. Licensee shall not claim against Axiomatic for lost
data, re-run time, inaccurate output, work delays or lost profits
resulting from Licensee' use of the Licensed Software.

To the extent not prohibited by law, in no event will Axiomatic be liable
for any indirect, punitive, special, incidental or consequential damage in
connection with or arising out of this Agreement (including loss of
business, revenue, profits, use, data or other economic advantage),
however it arises, whether for breach or in tort, even if the other party
has been previously advised of the possibility of such damage.


Do you agree? [Yes/No] yes
```

- After you agree with the disclaimer, then you have to answer all the questions below:

```
Name: <your name>
E-mail: <your_email@mail_domain.com>
Serial No: <our-serial-number>
License: <our-license-code-number>

Information Details:
====================
Product Name: Axiomatic Multi-Platform C
Platform: <purchased-OS-platform>
Version: <purchased-version-number>
License Type: <purchased-license-type>
Licensed To: <your name>
E-mail: <your_email@mail_domain.com>
Serial Number: <our-serial-number>
License Key: <our-license-code-number>
```

- After you have entered all the information, you will get the above information from the system. You have to check the information. If the information is correct, you have to answer yes, if not you have to quit from the installation process.

```
Is the above information correct and complete? [Yes/No/Quit] yes
```

- Upon successful installation, you will get the following message:

```
Congratulations!
Thank you for choosing AMPC.
```

- If you are using KDE, you can access **AMPC™** by navigating through **START APPLICATION > DEVELOPMENT > MORE PROGRAMS > AXIOMATIC MULTI-PLATFORM C**
- For first time installation, the shell may not be automatically loaded unless you re-initialize **AMPC**'s profile script. To initialize the environment, do the following steps;
- For bash shell:

```
% source $HOME/ampc/bin/ampc.sh
```

- For csh shell:

```
% source $HOME/ampc/bin/ampc.csh
```

***Note**: All files and packages have been pre-compiled using glibc version 2.3.2 on Intel x86 machine.*

1.3.2 Mac OS X Platform

- Download ampc-macosx-1.0.pkg.zip file from http://www.axiomsol.com
- To get the ampc-macosx-1.0.pkg file, you have to unpack the file ampc-macosx-1.0.pkg.zip.
- To install, use the Finder program and double click on the ampc-macosx-1.0.pkg installer package. Then, follow the instructions given.
- On the License page, read the license agreement. Once you agree with the license agreement, choose drive/partition destination where you want to install **AMPC™**.
- You need to enter administrator's name and password before you can proceed with the installation.
- The **AMPC™** will be installed in directory **/Applications/AMPC** on the chosen drive/partition.
- To execute the **AMPC™**, double click on **ampcgui (ampcgui.app)**.
- The ampcgui will install relevant files into your user home directory and set the correct permissions on the files.
- **To install license key information**, you have to:
  - Open the **/Applications/AMPC** directory (where ampcgui located).
  - Double click on the install-license file.
  - During installation process, you need to fill in some information. You have to read term described in License.rtf file and answer yes if you agree on it.
  - Do you agree with the term described in License.rtf or LICENSE file ? [Yes/No] yes
  - Once you answered "**yes**", you then have to fill in some information that had been emailed to you earlier.

```
Name: <your name>
E-mail: <your_email@mail_domain.com>
Serial No: <our-serial-number>
License: <our-license-code-number>
```

  - Then the system will prompt you with the details of the information.

```
Information Details:
====================
Product Name: Axiomatic Multi-Platform C
Platform: <purchased-OS-platform>
Version: <purchased-version-number>
```

```
License Type: <purchased-license-type>
Licensed To: <your name>
E-mail: <your_email@mail_domain.com>
Serial Number: <our-serial-number>
License Key: <our-license-code-number>
Is the above information is correct and complete? [Yes/No/Quit] yes
```

- Upon completion of the installation, you will receive the message as below:

```
Congratulations!
Thank you for choosing AMPC.
```

- **To Uninstall AMPC™**:
  - Open a terminal window and change your current directory to **/Applications/AMPC** directory.
  - Execute the uninstall.sh script file. This script will completely remove **AMPC™** from your computer system.

```
% ./uninstall.sh
```

  - You have to login as an administrator's user before you can completely remove **AMPC™** directory from **/Applications** directory.

***Note**: All files and packages have been pre-compiled on MacOS X 10.4 (Tiger), Macintosh Power PC G4 machine.*

1.3.3 Microsoft Windows Platform

- **AMPC™** installer can be downloaded from http://www.axiomsol.com. Before you proceed with the installation, you need to login as an administrator user in Windows XP.
- Click the **AMPC™** installer and it will install all the packages in the specified directory **C:\Program Files\AMPC** directory. During the installation, these are the packages that will be installed:
  - Ampcgui IDE
  - **AMPC™** compiler
  - Jasmin
  - Cygwin
- To install license key information, you have to go to **Start > All Programs > Axiomatic Multi-Platform C > Install License**
- This will execute the install license program and you have to provide few information. You have to read the terms described in License.rtf file and answer yes if you agree to it.

```
Do you agree with the term described in License.rtf or LICENSE file ?
[Yes/No] yes
```

- Once you answered "**yes**", you then have to fill in some information that had been emailed to you earlier.

```
Name: <your name>
E-mail: <your_email@mail_domain.com>
Serial No: <our-serial-number>
License: <our-license-code-number>
```

- Then the system will prompt you with the details of the information.

```
Information Details:
===============
Product Name: Axiomatic Multi-Platform C
Platform: <purchased-OS-platform>
Version: <purchased-version-number>
License Type: <purchased-license-type>
Licensed To: <your name>
E-mail: <your_email@mail_domain.com>
Serial Number: <our-serial-number>
License Key: <our-license-code-number>
Is the above information is correct and complete? [Yes/No/Quit] yes
```

- Upon completion of the installation, you will receive the message as below:

```
Congratulations!
Thank you for choosing AMPC.
```

- When the installation is completed, you need to restart first your computer before you can start using **AMPC™**.
- **To uninstall AMPC™**, open the **Start > Control Panel > Add/Remove Program** and choose **AMPC™** to remove.

1.4 RESTRICTIONS AND NOTES

1. All scalar data types are 1 word long. They are "char", "short", "int", "long", "long long", "float", "double", ...
2. JNI (JVM Native Interface) for AMPC is also supported. An example is given in the directory "ampc_jni".
3. Goto statements across functions or blocks not allowed.
4. fork() followed by exec() functionality is implemented differently. Here's an example of how to use it:

```
#include <stdio.h>
main()
{ char *cmd;
cmd = "ls -l";
INT_java("invokestatic _J_RunIt/fork_and_exec", "S", "V", STR1(cmd));
}
```

5. Memory size models for stack and heap are PICO, NANO, MICRO, TINY, SMALL, MEDIUM, LARGE, and HUGE. They are 0.5 meg, 1 meg, 2 megs, 4 megs, 8 megs, 16 megs, 20 megs, and 32 megs respectively. TINY model (4 megs) is the default.
6. The JVM limits each function/method to occupy at most 64KB of binary code space. Any function/method that is bigger than that will be caught by the JVM and execution is halted.
7. "Bit field" not supported, that is, the colon ":" operator can be used but ignored by the compiler as it is allocated a word-sized space.
8. It is encouraged that the source file names to have only characters that are valid C identifier characters. This is to avoid the possibility of the Jasmin assembler not being able to parse file names used in function calls (method invocations) due to the existence of non-identifier character(s).
9. Please set the "classpath" when running the "RUN" command (that invokes the JVM interpreter) using the "-cp" to include the location of the application being executed followed by the current location, followed by the location LOCAL_CLASSFILES, and followed by any other location you wish to include in the classpath. Alternatively, you may set the environment variable CLASSPATH for this purpose.

   Example:

```
%RUN -cp myapps/hello:/usr/local/lib/acc2jvm:. helloworld
```

1.5 TRADEMARK INFORMATION

1.6 DISCLAIMER OF WARRANTY

This software is provided by Axiomatic Solutions Sdn Bhd "AS IS" and any expressed or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the company be liable for any direct, indirect, incidental, special, exemplary or consequential damages (including but not limited to, procurement of substitute goods or services, loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in anyway out of the use of this software, even if advised of the possibility of such damage.

Licensee acknowledges that Licensed Software may contain errors and is not designed or intended for use in the design, construction, operation or maintenance of any nuclear facility ("High Risk Activities"). Axiomatic disclaims any expressed or implied warranties of fitness for such uses. Licensee represents and warrants to Axiomatic that it will not use, distribute or license the Licensed Software for High Risk Activities.

1.7 LIMITATION OF LIABILITY

Licensee acknowledges that the Licensed Software may have defects or deficiencies which cannot or will not be corrected by Axiomatic. Licensee will hold Axiomatic harmless from any claims based on Licensee's use of the Licensed Software for any purpose and from any claims that later versions or releases of any Licensed Software furnished to Licensee are incompatible with the Licensed Software provide to Licensee under this Agreement.

Licensee shall have the sole responsibility to protect adequately and backup Licensee's data and/or equipment used in connection with the Licensed Software. Licensee shall not claim against Axiomatic for lost data, re-run time, inaccurate output, work delays or lost profits resulting from Licensee's use of the Licensed Software.

To the extent not prohibited by law, in no event will Axiomatic be liable for any indirect, punitive, special, incidental or consequential damage in connection with or arising out of this Agreement (including loss of business, revenues, profits, use, data or other economic advantage), however it arises, whether for breach of in tort, even if the other party has been previously advised of the possibility of such damage.

2. Components of AMPC™

2.1 The User Interface (GUI)

The main GUI of **AMPC™** is the Main Window, which will be launched when **AMPC™** is executed. The Main Window is divided into several panels; **Menu bar** (at the top), **Toolbar**, **Project Panel**, **Editor** and **Output Panel**.

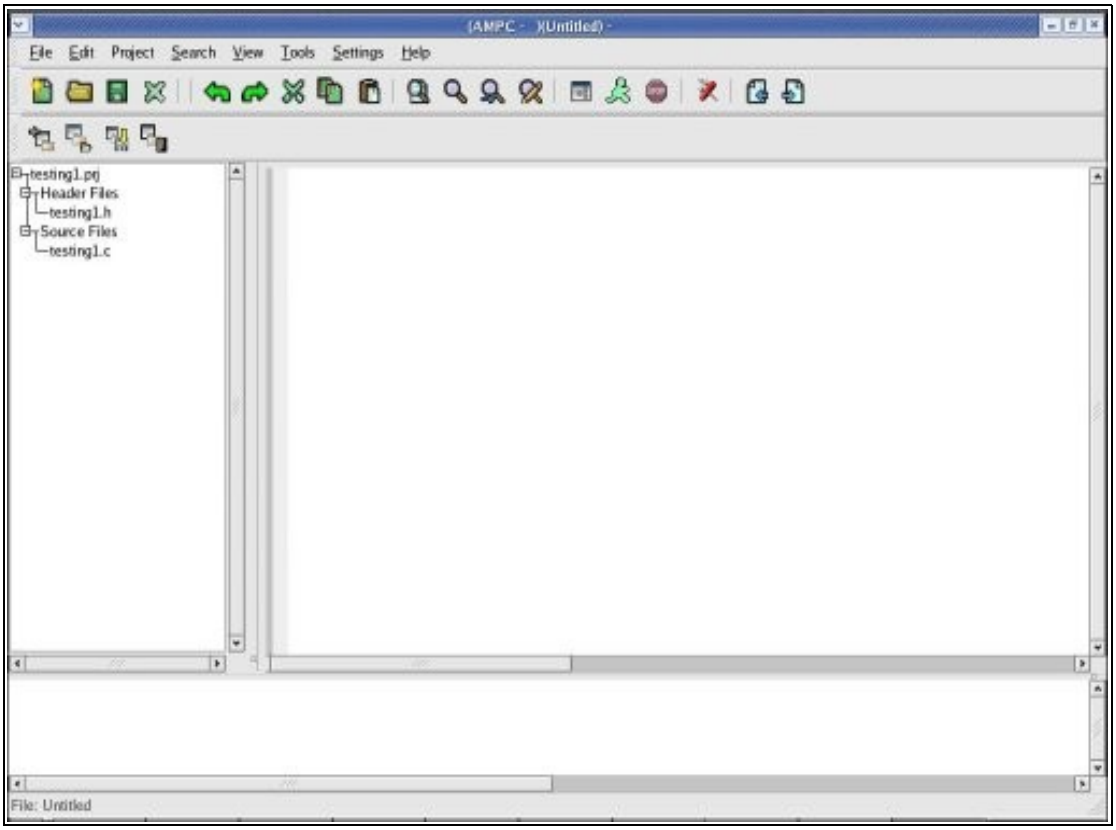

Figure 2.1

The first two panels, Menu bar and Toolbar, are the main interfaces to the backend components. In order to interact with the backend components of the **AMPC™**, users have to use the items in these two bars. Output from all the activities will be displayed in the Output Panel, which is located at the bottom of the Main Window. The Project Navigator Panel, which display files for a project file, is on the left side of the middle panel. The editor panel is on the right side.

2.1.1 Editor Panel

The editor panel is used for editing source code. The editor's main features are syntax styling, error indicators and code completion. The editor uses proportional fonts, bold and italic. Other than that, the editor uses multiple foreground and background colors to indicate the differences in the syntax.

The editor also provides editor margin, such as line numbering, and editor guide such as indentation guide.

2.1.2 Project Navigator Panel

You can manage all the source, header and project files in the Project Navigator panel. Files displayed in this panel will be arranged in tree form as depicted below:

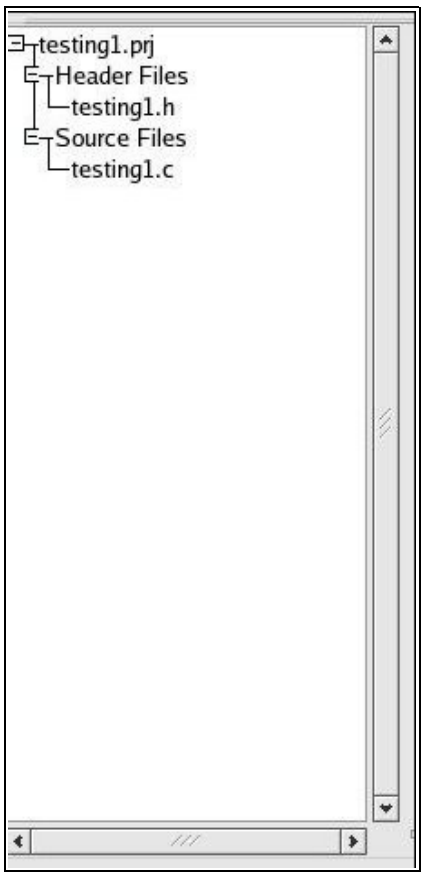

Figure 2.2

From this panel also, you can select and open a file by double clicking on the file.

2.1.3 Output Panel

Output and results of the **AMPC™** are displayed in the Output Window. The output displayed in different foreground colors based on the types of output. The output window can be set visible or invisible.

2.1.4 Menu Bar

You can interact with the other components of the IDE through the menubar. The menubar is categorized into several main categories. The items include **FILE**, **EDIT**, **PROJECT**, **SEARCH**, **VIEW**, **TOOLS**, **SETTINGS**, and **HELP**.

2.1.5 Toolbar

Another way that user can interact is through the items placed in the toolbar. By clicking on the specific item on the toolbar, the system will process the request. The toolbar can be made visible or invisible. This option can be accessed through the View item in the Menubar.

2.2 Compiler

The key feature of **AMPC™** compiler is that it emits Java bytecode instead of machine code. It translates C source codes into java byte codes. There are two ways to use the **AMPC™** compiler.

The first one is from the **AMPC™ GUI** that is available from the Tools menu item, and select "BUILD". This will execute the COMPILER and the compiler will compile the source code files and produce Java bytecode.

The second way is to access it from the console, where you can just type in COMPILE <filename.c> and it will generate the Java bytecode.

2.3 Debugger

(Will be available in future releases)

Although the Compiler produces the final application, the application may still crash during execution or may not even run, due to so-called "bugs" in the code. The bugs inside the codes usually end the execution of a program with the message "Segmentation fault".

AMPC™ IDE provides users with a tool to watch the internal values of an application and the execution step by step with setting "breakpoints" in the code. The debugger stops the execution every time the program comes to a breakpoint during execution. You can easily watch out for values and the setting of breakpoints in the code.

2.4 GUI Builder

(Will be available in future releases)

GUI Builder is a rapid application development tool for users to create a new graphical application. The GUI Builder will create templates and you can add in your source code to the templates based. The C source code will call Java Swing/SWT library for graphical applications.

PART II: STARTING/USING AMPC™

3. **Working on AMPC™ with GUI**

4. **Creating Applications**

5. **Compiler**

6. **Differences**

3. Working on AMPC™ with GUI

With the **AMPC™ IDE**, you will be able to compile and execute C programs in a JVM environment.

In case of more than one source files, a new project file has to be created. **You have to name the project file with the same name of the source file that contains the main function**. You can add in as many source files as you like. To build the application, you invoke the build system from the Tools menu. This will compile all the files and create the application.

3.1 Creating an Application Using the GUI

3.1.1 Creating Project File

To develop a new application with **AMPC™**, users have to create the application skeleton.

To start creating your application, select **NEW PROJECT** from the Project-menu A dialog box as below will appear:

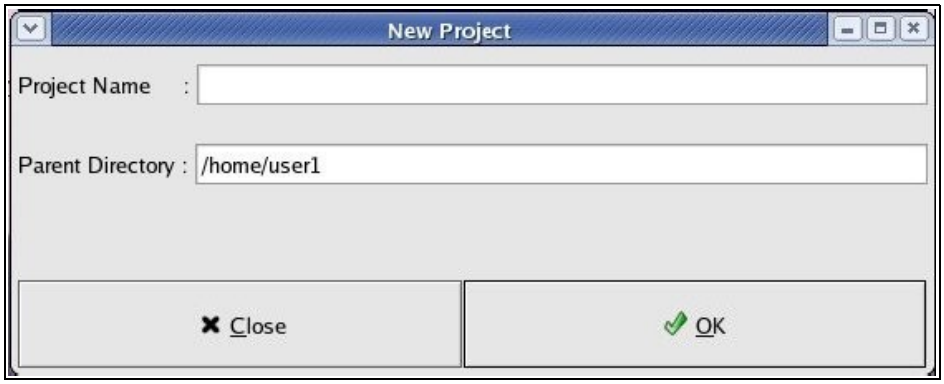

Figure 3.1

Type the project name and click **OK**. The project file will be created, together with a project skeleton. The project skeleton includes the project file (.prj), header file (*.h) and source file (.c).

The project file is the file you have to load to open the project in later sessions.

If you already have header files you want to use, you can choose those files as well. To build the binary, select **BUILD** from the Tools-menu or click on the **BUILD** button from the toolbar.

You can also test the functions already present by selecting **GO** from the Tools-menu.

***Note**: The name of the source file that contains the main function shall be the same as the project name, but with different file extension.*

3.1.2 Save Project File

Before you build/compile the application that you have created, save the project file. To save the project file, select the **SAVE PROJECT** command from the project-menu.

3.1.3 Build(Compile) Application

After you have created the application with a project file and the related source code files, you can produce the java byte code of your application by compiling the application. To compile the application, select **BUILD** from the Tools-menu or click on the **BUILD** button from the toolbar.

3.2 Managing Project Files

3.2.1 Creating A New File

To create a new file, choose **NEW** from the File-menu or click on the **NEW** button from the toolbar. A dialog box will appear, and enter your file name and choose whether it is a header file or a source file. After you have entered the filename the file will be added to your project file.

3.2.2 Adding Header File

To add an existing header file, select **ADD HEADER FILE** from the Project-menu. A dialog box will appear, and select the require header file from the dialog box. The selected file will appear on the Project Navigator Panel.

3.2.3 Opening and Closing of Project

To open your existing project file, select **OPEN PROJECT** from your Project-menu. A dialog box will appear, and select your project file. After you have selected your project file, the project file will be displayed in the Project Navigator Panel. You can choose a file to be displayed in the editor by double clicking on the file in the Project Navigator Panel.

3.3 Navigating AMPC™

3.3.1 Menubar

Users can interact with other components of the IDE through the menu bar. The menu bar is divided into several main categories - **FILE**, **EDIT**, **VIEW**, **PROJECT**, **SETTING**, **TOOLS** and **HELP**.

File-Menu

This section covers the functions which can be accessed via the File-menu in the menu bar.

NEW or **CTRL+N** creates a new file. The file can be created using different templates. The filename will be given during the saving of the file.

OPEN or **CTRL+O** opens a file. The **OPEN FILE** dialog box will be displayed and you will be prompted to select a file to be opened.

CLOSE or **CTRL+W** closes the active file in the editing window.

SAVE or **CTRL+S** saves a file. This saves the active file in the top editing window. If the file has not been saved yet, the **SAVE FILE AS...** dialog will be opened to let you choose a path and filename for the file to be saved.

SAVE AS from the File-menu saves the current file under a new name. Once clicked, the **SAVE FILE AS...** dialog will appear and a new name has to be given for the file.

To exit **AMPC™**, click **EXIT** from the File-menu. If changes are made to any of the files, you will be asked if these files need to be saved.

Edit-Menu

The Edit menu provides editing functions while editing files. The editing functions are available via a context-menu in the editor.

UNDO or **CTRL+Z** reverts the last editing operation

REDO or **CTRL+Y** does the last undo step again

CUT or **CTRL+X** cuts out a selection and copies it to the system clipboard.

COPY or **CTRL+C** copies a selection to the system clipboard.

PASTE or **CTRL+V** inserts the clipboard contents at the current cursor position.

INSERT FILE or **CTRL+INSERT** selects a file and inserts its contents at the current cursor position.

Project-Menu

The Project-menu provides functions to create and maintain projects. User can manage all the source, header and project files from within the Project Navigator Panel. Files displayed in this panel will be arranged in tree form as depicted below:

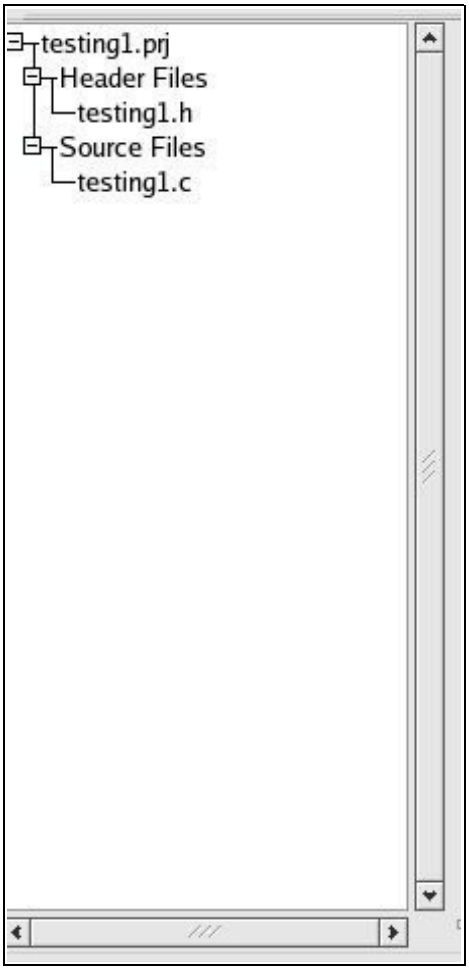

Figure 3.2

From this panel, user can select a file by double clicking on the file and the selected file will be opened in the editor.

The **NEW PROJECT** command from the project-menu will invoke the Application Wizard and allows user to create a new project by choosing application type.

The **OPEN PROJECT** command from the project-menu will call up the Open Project dialog box, where you can choose a project file to be opened. After selection, the project will be loaded.

The **CLOSE PROJECT** command from the project-menu will close the current project.

The **SAVE PROJECT** command from the project-menu will save the current

project file.

The **SAVE AS PROJECT** command from the project-menu will invoke the **PROJECT SAVE AS** dialog box. Enter the name of the project file. The project file will be given *.prj extension.

The **ADD HEADER FILE TO PROJECT** command from the project-menu will invoke the **ADD HEADER FILE TO PROJECT** dialog box. Select the required header file and the file will be added to the project file. You need to save the project file.

The **ADD SOURCE FILE TO PROJECT** command from the project-menu will invoke the **ADD SOURCE FILE TO PROJECT** dialog. Select the required source file and the file will be added to the project file. You need to save the project file.

The **DELETE FILE FROM PROJECT** command will delete the selected file from the project file. It is advisable to save the project file after the deletion is completed.

Search-Menu

The **FIND** from the Search-menu or **CTRL+F** will invoke the **FIND** dialog box that prompts you to input an expression that user want to find in the current file. The cursor will go to the location that has the expression that matches the expression that user has typed.

FIND NEXT or **F3** will ask the system to look for the next location that matches the given expression.

FIND PREVIOUS or **SHIFT+F3** will allow the system to look for the previous location that matches the given expression.

FIND IN FILES… or **SHIFT+CTRL+F** will invoke the **FIND IN FILES** dialog. You will be able to search for an expression from several files. Input the expression and name of files in the dialog box.

REPLACE.. or **CTRL+H** will invoke the **REPLACE** dialog window. Input the expression that you want to replace and the new expression.

GO TO or **CTRL+G** will invoke the **GO TO** dialog window. Type the line number and the cursor will go to the specified line number.

TOGGLE BOOKMARK or **CTRL+F2** will enable or disable the bookmark at the current line number in the editor.

NEXT BOOKMARK or **F2** will move the cursor to the next bookmark.

PREVIOUS BOOKMARK or **SHIFT+F2** will move the cursor to the previous bookmark.

CLEAR ALL BOOKMARK will clear all the bookmarks.

View-Menu

FULL SCREEN or **F11** enables or disables the full screen.

TOOLBAR enables or disables the toolbar.

STATUS BAR enables or disables the status bar.

END OF LINE enables or disables the end of line indicator in the editor.

INDENTATION GUIDES enables or disables the indentation guides in the editor.

LINE NUMBERS enables or disables the display of line numbers in the editor.

MARGIN will enable or disable margin for the editor.

FOLD MARGIN enables or disables fold margin for the editor.

OUTPUT from the View-menu will enable or disable the output window panel.

Tools-Menu

The Tool-menu allow user to compile and build applications.

To compile source files into *.class files: Select **BUILD** from the Tools-menu or press **F7**.

RUN or press **F5** executes an application.

STOP stops the execution of the application.

CLEAR OUTPUT or **SHIFT+F5** clears all the messages in the output pane.

SWITCH PANE or **CTRL+F6** switches the active panel.

Settings-Menu

From Menu, click setting and choose **COMPILER OPTIONS**. The Compiler Options dialog box will appear. There are 4 options available:

- AMPC Memory Model
- AMPC Optimization Model
- Include Path
- Define

3.3.2 Toolbar

Users can interact with the items placed in the toolbar by clicking on the specific item in the toolbar.

The Toolbar can be made visible or invisible. This option can be accessed in the View item in the Menu bar.

The toolbar is as depicted below:

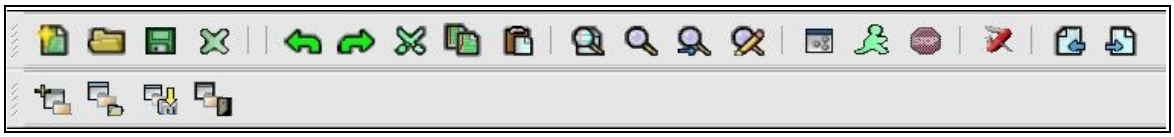

Figure 3.3

Figure 3.4

Click the above icon to create a new file.

A dialog box will appear, and users will be prompted to fill in the filename and select whether the file is a source file or a header file. Then, click **OK**. The file can also be created using different templates.

Figure 3.5

Click the above icon to open a file.

The **OPEN FILE** dialog box will be displayed and you will be prompted to select a file to be opened. The file will be opened in the editor window.

Figure 3.6

Click the above icon to save a file.

This saves the active file in the top editing window. If the file has not been saved yet, the **SAVE FILE AS...** dialog will be opened to let you choose a path and filename for the file to be saved.

Figure 3.7

Click the above icon to close an active file.

To close an active file in the editing window.

Figure 3.8

Click the above icon to undo the last editing operation. Once clicked, the last editing operation will be reverted.

Figure 3.9

Click the above icon to redo the last undo step again.

Figure 3.10

Click the above icon to cut out a selection and copy it to the system clipboard.

Figure 3.11

Click the above icon to copy a selection to the system clipboard.

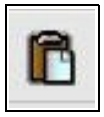

Figure 3.12

Click the above icon to insert the clipboard contents at the current cursor position.

Figure 3.13

Click the above icon to build the project file. All the source files in the project files will be compiled to *.class files.

Figure 3.14

Click the above icon to execute an application.

Figure 3.15

Click the above icon to stop the execution of the application.

Figure 3.16

Click the above icon to display a Compiler Options dialog box.

Figure 3.17

Clicking the above icon will invoke the **APPLICATION WIZARD**. The application wizard allows user to create a new project file. User has to type in a project name. The project file and project directory will be created. A project tree will be displayed on the project window (in the left panel). Besides the project file, a skeleton source file and header file will also be created.

Figure 3.18

Click the above icon, and the system will call up the **OPEN PROJECT** dialog box, where you can choose a project file to be opened. After selection, the project will be loaded into the project window.

Figure 3.19

Click the above icon to save the current project file.

Figure 3.20

Click the above icon to close the current project file.

3.4 Navigating The Editor

3.4.1 Shortcut

To use the editor, you should make yourself comfortable with some keyboard shortcuts that make it easier to position the cursor and edit the file.

Shortcut Keys	Function
Left Arrow	Move one letter to the left
Right Arrow	Move one letter to the right
CTRL + Left Arrow	Move one word to the left
CTRL + Right Arrow	Move one word to the right
Up Arrow	Move one line up
Down Arrow	Move one line down
Home	Move to the beginning of the line
End	Move to the end of the line
PageUp	Move one page up
PageDown	Move one page down
SHIFT+Left Arrow	Move one letter to the left
SHIFT+Right Arrow	Move one letter to the right
CTRL+SHIFT+Left Arrow	Move and select one word to the left
CTRL+SHIFT+Right Arrow	Move and select one word to the right
CTRL+Home	Move to the beginning of the current file
CTRL+End	Move to the end of the current file
SHIFT+PageUp	Move and select one page up
SHIFT+PageDown	Move and select one page down
INS	Enable and disable insert mode
CTRL+C	Copy the selected text to the clipboard
CTRL+V	Insert the text from the clipboard
CTRL+L	Delete current line
CTRL+Z	Undo editing step
CTRL+Y	Redo the undo step

4. Creating Applications

4.1 Console Applications

To create an application, users have to create a project file first by selecting **NEW PROJECT** from Project-Menu. The name of the project file should be the same with the source file that contains a main function.

Example: To print output of "Hello World".

Create a project file, **NEW PROJECT**, then enter the project name. Then go to the source file, and write your source code.

Example:

```
#include <stdio.h>
main ()
{
      printf("Hello World \n");
}
```

Save the source file and the project file. Then compile your application by selecting **BUILD** from Tools-Menu. If there is no compilation error, you can execute the application by selecting **RUN** from the Tools-Menu.

You can pass arguments to the console applications by passing the two arguments, which are argc and argv, on the command line. Both parameters are declared implicitly.

Example:

```
#include <stdio.h>
main ()
{
      int cnt;
      printf("%i parameters entered \.",argc);

      for (cnt = 0; cnt <= argc -1; cnt++)
      {
            printf("Parameter %d is %s\n",cnt, argv[cnt]);
      }
}
```

4.2 Calling Java Method From AMPC™ Code

One of the most interesting features of **AMPC™** is that it allows you to call java methods from your C programs. Therefore, you could interface with your existing or new java methods.

You can get access to java method by using **INT_java**, **FLOAT_java** and **DOUBLE_java**.

Example:

```
INT_java("invokestatic <class>/<method>", "IISI", "V", var1, var2,
STR1(var3), var4);
```

<class> : the java class to you want to refer.

<method> : the java method that you want to access.

"IISI" : is the variable types that you want to pass to the java method.

The types that you can use are :

Integer - I

String - S

Float - F

Char - C

DOUBLE - D

"V" : the return variable. V stands for void. You can replace V with other types of variables except Float. If the returned type of the java method is float, you will have to use Float_java.

```
Float_java("invokestatic <class>/<method>, "IISI", "F", var1, var2,
STR1(var3), var4);
```

Restrictions:
You cannot pass object to the java method. Currently, you can only pass 4 strings at a time. You have to use **STR1** for the first string, and **STR2**, **STR3** and **STR4** respectively.

Example:

```
INT_java("invokestatic <class>/<method>", "IISI", "V", var1, var2,
STR1(var3), var4)
```

var3 is of type string. You have to use **STR1(var3)** in the **INT_java**, **FLOAT_java** and **DOUBLE_java**. If **var4** is also of type string, you have to use **STR2(var4)**.

Example:

```
void Create_Button(int object_ID, int container_ID, char *title)
{
      int done;

      done = INT_java("invokestatic Java_APP/CreateButton", "ISI", "I",
            object_ID, STR1(title), container_ID);
}
```

The above example, called java method of class Java_APP, that is CreateButton by passing 3 parameters, which are of types integer, string and integer with a return type of integer.

In this example, the java method returns an integer value. If the return type of java method is of type float, you have to use **FLOAT_java**.

```
void Create_Button(int object_ID, int container_ID, char *title)
{
      float done;
      done = FLOAT_java("invokestatic Java_APP/CreateButton", "ISI", "F",
            object_ID, STR1(title), container_ID);
}
```

Example:

```
#include <stdio.h>
cont()
{
      DOUBLE Dval;
      Dval = DOUBLE_java("invokestatic java/lang/Math/cos", "D", "D",
            _DBL1(_f2D_DOUBLE(45.0)));
      printf("Dval = %D\n", Dval);
}
```

The above example shows the usage of **DOUBLE_java** where the return value of Java method "**cos**" is of type **DOUBLE**. You have to use **DBL1(DOUBLE)** in the **INT_java**, **FLOAT_java** and **DOUBLE_java**. Please refer 6.1: Using the **DOUBLE** type in **AMPC™**, for the details of **DOUBLE** operations.

4.3 Calling Native C Functions From AMPC™

The JVM Native Interface (JNI) is the native programming interface for **AMPC™**.

This is for the purpose of calling native C functions from **AMPC C** code.

This is handy since huge amount of code has been written over the years and are stable enough not to fiddle with, that it would be convenient to just simply use them by interfacing them from **AMPC™**. Also, the source code for such applications might not be available to be ported to **AMPC™**, and the only way to use them is by means of **JNI**.

4.3.1 Some Background

Writing native functions for **AMPC™** programs requires several steps:

1. You need to start by writing an **AMPC™** program. Create a function that declares the native function; this program contains the declaration or signature for the native function. It also includes a main function that calls the native function.
2. Compile the **AMPC™** program that declares the native function and the main function.
3. Generate a header file for the native function using the utility program called "**javah**". Once the header file has been generated what we have is the formal signature for the native function.
4. Then, write the implementation of the native function in a language such as C or C++.
5. Now you need to compile the header and implementation files into a shared library file.
6. Finally, run the **AMPC™** program.

4.3.2 An Example

1. First, you need to write the **AMPC™** code such as this:

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
FILE *fopen();

void _NATIVE_interface(void)
{
  _NATIVE_prototype("_unix_stat", "S", "I");
}

void _NATIVE_shared_library(void)
{
  _NATIVE_load_library("StatLib");
}

int stat(char *file_name, struct stat2 *buf)
{ int retval;
  FILE *_transit_file;
```

```
  retval = _INT_NATIVE_call("_unix_stat", "S", "I", STR1(file_name));
  if((_transit_file = fopen("._transit_stat_buf.txt", "r")) != NULL)
  { fscanf(_transit_file, "%x %x %x %x %x %x %x %x %x %x %x",
    &(buf->st_dev),
    &(buf->st_ino),
    &(buf->st_mode),
    &(buf->st_nlink),
    &(buf->st_uid),
    &(buf->st_gid),
    &(buf->st_rdev),
    &(buf->st_size),
    &(buf->st_atime),
    &(buf->st_mtime),
    &(buf->st_ctime));
    fclose(_transit_file);
    remove("._transit_stat_buf.txt");
    return(retval);
  }
  else
  { fprintf(stderr, "Error opening file: _transit_stat_buf.txt\n");
    exit(1);
  }
}
```

2. The **_NATIVE_interface** function declaration above provides only the function signature for "**unix_stat**" by passing it in the built-in function **_NATIVE_prototype**. It does not provide the implementation for the function. You need to provide the implementation for "**unix_stat**" in a separate native language source file.
3. Then, you need to compile the above by typing:

```
% COMPILE stat.c
```

   which produces the file "**stat.class**".
4. Then, type:

```
% javah stat
```

   which results in the generation of the "**stat.h**" file. In this example the name "**Java_stat__1unix_1stat**" is automatically created by "**javah**" to be used as the name of the native function in the source file "**unix_stat.c**".
5. The **_NATIVE_shared_library** function is where you load the shared library by calling the function **_NATIVE_load_library**. The **libStatLib.so** library is created by doing the following:

```
gcc -shared -I/usr/local/j2sdk1.4.2/include \
                -I/usr/local/j2sdk1.4.2/include/linux \
                unix_stat.c -o libStatLib.so
```

6. And, this is an example of the "**unix_stat.c**" code.

```
#include <jni.h>
#include "./stat.h"
#include <stdio.h>
#include <sys/stat.h>
FILE *fopen();

JNIEXPORT jint JNICALL Java_stat__1unix_1stat
  (JNIEnv *env, jobject obj, jstring file_name)
{ char *fname;
  FILE *_transit_file;
  struct stat buf;
  int retval;

  fname = (*env)->GetStringUTFChars(env, file_name, 0);
  retval = stat(fname, &buf);
  if((_transit_file = fopen("._transit_stat_buf.txt", "w"))!= NULL)
  { fprintf(_transit_file, "%x %x %x %x %x %x %x %x %x %x %x",
    (int) buf.st_dev,
    (int) buf.st_ino,
    (int) buf.st_mode,
    (int) buf.st_nlink,
    (int) buf.st_uid,
    (int) buf.st_gid,
    (int) buf.st_rdev,
    (int) buf.st_size,
    (int) buf.st_atime,
    (int) buf.st_mtime,
    (int) buf.st_ctime);
    fclose(_transit_file);
  }
  else
  { fprintf(stderr, "Error opening file: \"_transit_stat_buf.txt\" for
writing\n");
    exit(1);
  }
  (*env)->ReleaseStringUTFChars(env, file_name, fname);
  return(retval);
}
```

7. Finally, write an application that utilizes the above "**stat.class**" function such as the following "**file_exists.c**" program which checks for the existence of the text file "**abc123.txt**" by using the "**stat**" function which in turn calls the native function "**Java_stat__1unix_1stat**" defined in "**unix_stat.c**".
8. Here's the program in the file "**file_exists.c**":

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>

void cont()
{ char *a_file = "abc123.txt";
  struct stat2 statBuf;

  if(stat(a_file, &statBuf))
```

```
    printf("Cannot stat file: \"%s\"\n", a_file);
  else
    printf("File \"%s\" exists\n", a_file);
}

main()
{
  cont();
}
```

9. Test the "**file_exists**" program by typing:

```
% RUN file_exists
```

4.4 GUI-based Applications

AMPC™ also provides you with GUI libraries. Please refer to **AMPC™ Function Libraries** (GUI Libraries - New) for the details of the libraries. Before you create a GUI-based application, first you have to create a project file for the application. Then write in your source code in the source file. An example of a source file is as below.

4.4.1 An Example

This example is to create several button types in a window

```
#include <stdio.h>
#include <stdlib.h>
#include <gui/gui.h>

AmpcWidget *win;
AmpcWidget *shell;
AmpcWidget *button1;
AmpcWidget *button2;
AmpcWidget *btn_up, *btn_left, *btn_right, *btn_down;
AmpcWidget *radio,*toggle,*flat;
AmpcImage *image;

void MainCalc()
{
  win = ampc_display_new();
  shell = ampc_shell_new(win,"Test Button With Image ");

  button1 = ampc_button_new(shell, AMPC_CHECK);
  ampc_button_set_bounds(button1,10,40,100,30);
  ampc_button_set_text(button1, "check box (AMPC_CHECK)");

  button2 = ampc_button_new(shell, AMPC_PUSH);
  ampc_button_set_bounds(button2,10,80,10,10);
  ampc_control_setSize(button2,220,30);
  ampc_button_set_text(button2, "push button(AMPC_PUSH)");

  btn_up = ampc_button_new(shell, AMPC_ARROW);
  ampc_button_set_bounds(btn_up,10,120,30,30);

  btn_left = ampc_button_new(shell, AMPC_ARROW);
  ampc_button_set_bounds(btn_left,50,120,30,30);
```

```
  radio = ampc_button_new(shell, AMPC_RADIO);
  ampc_button_set_bounds(radio,10,160,100,30);
  ampc_button_set_text(radio, "AMPC_RADIO");

  toggle = ampc_button_new(shell, AMPC_TOGGLE);
  ampc_button_set_bounds(toggle,10,200,100,30);
  ampc_button_set_text(toggle, "AMPC_TOGGLE");

  flat = ampc_button_new(shell, AMPC_FLAT);
  ampc_button_set_bounds(flat,10,240,100,30);
  ampc_button_set_text(flat, "AMPC_FLAT");

  image = ampc_graphics_image_new(shell,"one.gif");
  ampc_button_set_image(button1, image);
  ampc_shell_open(shell);
}

void main()
{
  ampc_init();
  MainCalc();
  ampc_main(win,shell);
}
```

4.5 Network-based Applications

AMPC™ also provides you with Network libraries. Currently, there is only a TCP protocol libraries available. Please refer to **AMPC™ Function Libraries** for the details of the libraries. An example of a source file is as below. This example is to create an echo client/server application.

4.5.1 TCP Client Sample Source Code

```
#include <stdio.h>
#include <string.h>
#include <network.h>

main()
{
  int sockID;
  char svrname[100];
  int svrport;

  char msg[] = "A text message";
  char msgrcvd[100];
  char totalmsg[100];
  int msglen;
  int bytercvd;
  int totalbytercvd = 0;

  if ((argc < 1) && (argc > 4))
  {
    fprintf(stderr, "Usage: %s <Query host> <port>\n", argv[0]);
    exit(1);
  }
```

```
  strcpy(svrname, argv[1]);
  svrport = atoi(argv[2]);

  printf("\nServer Name: %s\nServer Port: %d\n", svrname, svrport);

  sockID=socket(svrname, svrport);
  printf("Connected to server... sending echo string.\n");

  msglen = strlen(msg);
  send(sockID, msg, msglen, CLIENT);
  printf("dbg: After send function.\n");

  totalmsg[0]='\0';
  while(totalbytercvd < msglen)
  {
    if((bytercvd = recv(&sockID, msgrcvd,msglen, CLIENT)) == -1)
    {
      printf("Connection close prematurely.\n");
      exit(1);
    }
    totalbytercvd += bytercvd;
    msgrcvd[msglen] = '\0';
    strcat(totalmsg, msgrcvd);
  }

  printf("Received: %s\n", totalmsg);

  socketclose(CLIENT);
}
```

4.5.2 TCP Server Sample Source Code

```
#include <stdio.h>
#include <string.h>
#include <network.h>
#define BUFSIZE 32

int main()
{
  int svrport;
  int recvmsgsize;
  int clntsockID;

  char clnthost[32];
  int clntport;

  char msgrcvd[100];
  int msglen = 0;

  if((argc < 1) && (argc >1))
  {
    fprintf(stderr, "Usage: %s <Server port no.>\n", argv[0]);
    exit(1);
  }

  svrport = atoi(argv[1]);
  printf("Preparing opening port at %d\n", svrport);

  serversocket(svrport);
```

```
  printf("Listening at port %d\n", svrport);

  while(1)
  {
    if((clntsockID=accept(clnthost)) == -1)
      printf("Error accepting client\n");
    else
      printf("Handling client from %s[sockID=%d]\n", clnthost,
clntsockID);

    while((recvmsgsize = recv(&clntsockID, msgrcvd, msglen, SERVER)) !=
-1)
    {
      send(clntsockID, msgrcvd, msglen, SERVER);
    }

    printf("Close connection from client %s.\n", clnthost);
    socketclose(SERVER);
    printf("Closed\n");
  }
  /* NOT REACHED */
}
```

4.6 Database-based Applications

AMPC™ also provides the Open Database Connectivity (ODBC) libraries. The ODBC standard defines the common application programming interface that allows programs and programmers to communicate with any SQL database which has an ODBC driver. Refer to **AMPC™** Function Libraries for the details of the libraries.

4.6.1 An Example

***Note**: This release only supports MySQL database and Linux Platform.*

```
#include <stdio.h>
#include <stdlib.h>
#include <sql.h>
#include <sqlext.h>
#include <sqltypes.h>
#include <odbcinst.h>

int main()
{
  SQLHENV henv;
  SQLHDBC hdbc;
  SQLHSTMT hstmt;
  SQLRETURN rc;

  SQLCHAR dataSource[100] =
    "jdbc:mysql://localhost/employees";

  SQLCHAR errmsg[SQL_MAX_MESSAGE_LENGTH];
  SQLCHAR status[10], colName[255];
  SQLINTEGER error, nRow ;
  SQLSMALLINT mlen, nCol, nIndex, sqlType, scale, nullable, colNameLength;
  SQLUINTEGER colSize;
```

```
  char driver[100];
  char fname[256], lname[256], title[256], email[256], salary[10];
  SQLINTEGER empId;

  printf("\nAllocating Environment Handle .....");

  rc = SQLAllocHandle(SQL_HANDLE_ENV, (void *) SQL_NULL_HANDLE, &henv);

  if ((rc != SQL_SUCCESS) && (rc != SQL_SUCCESS_WITH_INFO)) {
    printf("\nError AllocHandle\n");
    exit (1);
  }
  printf(" Allocated !\nSetting the Environment Version .......");

  rc = SQLSetEnvAttr(henv , SQL_ATTR_ODBC_VERSION , (SQLPOINTER)
SQL_OV_ODBC3, 0);

  if ((rc != SQL_SUCCESS) && (rc != SQL_SUCCESS_WITH_INFO)) {
    printf("\nError SQLSetEnvAttr\n");
    SQLFreeHandle(SQL_HANDLE_ENV , henv);
    exit(1);
  }
  printf(" Set !\nAllocating Connection Handle .....");

  rc = SQLAllocHandle(SQL_HANDLE_DBC , henv , &hdbc);

  if ((rc != SQL_SUCCESS) && (rc != SQL_SUCCESS_WITH_INFO)) {
    printf("\nError Allocating Connection Handle %d\n",rc);
    SQLFreeHandle(SQL_HANDLE_ENV , henv);
    exit(1);
  }
  printf(" Allocated !\nLoading SQL Driver ...................");

  rc = SQLLoadDriver(hdbc,MYSQL , driver);

  if ((rc != SQL_SUCCESS) && (rc != SQL_SUCCESS_WITH_INFO)) {
    printf("\nFailed to load the driver\n");
    SQLFreeHandle(SQL_HANDLE_ENV, henv);
    exit(1);
  }
  printf(" Loaded !");
  printf("\n\n\t*********************************************");
  printf("\n\t Driver Name : %s",driver);
  printf("\n\t*********************************************");
  printf("\n\nEstablishing connection to database ......");

  rc = SQLConnect(hdbc , dataSource, SQL_NTS , (SQLCHAR *)

  "zalfa", SQL_NTS , (SQLCHAR *) "qwerty" , SQL_NTS);
  if ((rc != SQL_SUCCESS ) && (rc != SQL_SUCCESS_WITH_INFO)) {
    printf("\nError SQLConnect\n");
    SQLFreeHandle(SQL_HANDLE_ENV , henv);
    exit(1);
  }
  printf(" Connection Established !\nAllocating Statement Handle ......");

  rc = SQLAllocHandle(SQL_HANDLE_STMT , hdbc , &hstmt);

  if ((rc != SQL_SUCCESS) && (rc != SQL_SUCCESS_WITH_INFO)) {
    printf("\nError AllocStatement %d\n",rc);
    SQLDisconnect(hdbc);
```

```
    SQLFreeHandle (SQL_HANDLE_DBC , hdbc);
    SQLFreeHandle(SQL_HANDLE_ENV , henv);
    exit(1);
  }
  printf(" Allocated !\nExecuting SQL Statement ..............");

  rc = SQLExecDirect(hstmt , "SELECT fname, lname, title FROM
employee_data", SQL_NTS);

  if ((rc != SQL_SUCCESS) && (rc != SQL_SUCCESS_WITH_INFO)) {
    printf("\nError SQLExecDirect %d\n",rc);
    SQLFreeHandle(SQL_HANDLE_STMT , hstmt);
    SQLDisconnect (hdbc);
    SQLFreeHandle(SQL_HANDLE_DBC , hdbc);
    SQLFreeHandle(SQL_HANDLE_ENV , henv);
    exit(1);
  }
  printf(" Success !\n\n");

  while ( (rc = SQLFetch (hstmt)) != SQL_NO_DATA) {
    SQLGetData( hstmt, 1, SQL_C_CHAR, fname, sizeof(fname), &error);
    SQLGetData( hstmt, 2, SQL_C_CHAR, lname, sizeof(lname), &error);
    SQLGetData( hstmt, 3, SQL_C_CHAR, title, sizeof(title), &error);

    printf("Column 1 : %s \t", fname);
    printf("Column 2 : %s \t", lname);
    printf("Column 3 : %s \t", title);
    printf("\n");
  }

  SQLFreeHandle(SQL_HANDLE_STMT , hstmt);
  SQLDisconnect(hdbc);
  SQLFreeHandle(SQL_HANDLE_DBC , hdbc);
  SQLFreeHandle(SQL_HANDLE_ENV , henv);

  return 0;
}
```

4.7 Embedding Assembly Code in Your C Code

If you want to go into lower level, you can embed an assembly code in your C source code. For example, if you want to add in assembly code in the previous example, do the following:

```
#include <stdio.h>

main ()
{
  asm("; Testing inline asm\n"); /* assembly code*/
  printf("Hello World \n");
}
```

The syntax is as below:

```
asm("<assembly code>");
```

You can refer to the jasmin manual for details on assembly code syntax and usage.

5. Compiler

5.1 Using Compiler - GUI

The compiler can be accessed from the Tools-Menu and choose "**BUILD**". **AMPC™** will compile all your source files in your project file. You can change the "**COMPILER OPTIONS**" by selecting the Setting-Menu and choosing the options that you require.

5.2 Using Compiler - Console

Other than through the GUI, users can also use the compiler from console. The syntax to be use with C source file(s) (.c):

5.2.1 Syntax

Usage: COMPILE [-options] [-memory model] [-optimization] [-debuging] sourcefile1.c [sourcefile2.c] ...

e.g: COMPILE test1.c test2.c test3.c

e.g: COMPILE -I../include -DUNIX test1.c test2.c test3.c

Syntax to be use with project file (.prj):

Usage: COMPILE [-options] [-memory model] [-optimization] [-debuging] -f sourceprojfile.prj

e.g: COMPILE -f test.prj

Options:

-S	Compile only, the output will be a .s file (default option).
-R	Compile and resolve link, the output will be a .s file.
-E	Preprocess only; do not compile, assemble or link.
-I/dir	Include the directory /dir to the list of include directories to be searched for header files.
-o file	Write output to file.
-Dname	Predefine name as a macro, with definition 1.
-Dname=definition	Predefine name as a macro, with definition=definition.
-Uname	Cancel any previous definition of name, either built in or provided with a -D option.

Memory model:

-m H huge memory model (32MB).

-m L large memory model (20MB).
-m M medium memory model (16MB).
-m S small memory model (8MB).
-m T tiny memory model (4MB).
-m U micro memory model (2MB).
-m N nano memory model (1MB).
-m P pico memory model (0.5MB).

Optimization:
-O0 optimization level 0 (safe mode option).
-O1 optimization level 1 (default option).
-O2 optimization level 2.
-O3 optimization level 3.

Debuging:
-s Do not delete assembly file (.s file).

Note: *The filename must be a .c file.*

5.2.2 Descriptions

COMPILE is a program to resolve "**invokestatic**" and "**invokespecial**" reference call to functions/methods from other file at assembler code level. It is just like a linker where it resolves external functions in a different file. Binary package, support for Linux, Mac OS X and MS Windows.

5.2.3 Dependencies

- libxml2 (/usr/lib/libxml2.so.2)
- libz (/lib/libz.so.1)
- libpthread (/lib/i686/libpthread.so.0)
- libm (/lib/i686/libm.so.6)

5.3 Executing Application - GUI

To execute the application that has been compiled through **AMPCGUI**, it can be accessed from the Tools-Menu and choose "**RUN**". **AMPCGUI** will execute and display the output in new terminal/shell window.

5.4 Executing Application - Console

To execute the application that has been compiled, use the following syntax.

5.4.1 Syntax

Usage: RUN javabytecodefile [arg1] [arg2] ...

e.g: RUN helloworld

5.4.2 Descriptions

RUN is a script to execute program compiled by **AMPC™**.

The program source file must contain **main()** function to make it runable. The script also accepts multiple argument variables which passed from command line.

Note: *Shell script file only support for Linux and Mac OS X. For Ms Windows platform, use* ***runjava.bat*** *instead.*

5.4.3 Dependencies

- /bin/sh shell program.
- java program.

6. Differences From Standard C

AMPC™ supports a very large subset of **ANSI C**. One notable difference is that "**double**" in **AMPC™** is 32 bits long. In order to utilize 64-bit floating point you can use "**DOUBLE**". Please refer to **section 6.1** for the details on how to use "**DOUBLE**".

The other difference is thay you do not need to declare variables "**argc**" and "**argv**". They are already pre-declared. The details are in **section 6.2**.

6.1 Using the DOUBLE type in AMPC™

6.1.1 DOUBLE Functions

"**DOUBLE**" is our implementation of the 64-bit long floating point type. It is declared as a struct of two float fields. The operations associated with "**DOUBLE**" are as follows:

```
DOUBLE _f2D_DOUBLE(float);
DOUBLE _i2D_DOUBLE(int);
DOUBLE _c2D_DOUBLE(char);
float _D2f_DOUBLE(DOUBLE);
int _D2i_DOUBLE(DOUBLE);
char _D2c_DOUBLE(DOUBLE);
DOUBLE _add_DOUBLE(DOUBLE, DOUBLE);
DOUBLE _sub_DOUBLE(DOUBLE, DOUBLE);
DOUBLE _mul_DOUBLE(DOUBLE, DOUBLE);
DOUBLE _div_DOUBLE(DOUBLE, DOUBLE);
int _equal_DOUBLE(DOUBLE, DOUBLE);
int _less(DOUBLE, DOUBLE);
int _less_equal(DOUBLE, DOUBLE);
int _not_equal(DOUBLE, DOUBLE);
int _greater(DOUBLE, DOUBLE);
int _greater_equal(DOUBLE, DOUBLE);
```

Mathematical operations associated with "**DOUBLE**" are as follows:

```
DOUBLE _sin_DOUBLE(DOUBLE);
DOUBLE _sinh_DOUBLE(DOUBLE);
DOUBLE _sqrt_DOUBLE(DOUBLE);
DOUBLE _tan_DOUBLE(DOUBLE);
DOUBLE _tanh_DOUBLE(DOUBLE);
DOUBLE _acos_DOUBLE(DOUBLE);
DOUBLE _asin_DOUBLE(DOUBLE);
DOUBLE _atan2_DOUBLE(DOUBLE,DOUBLE);
DOUBLE _atan_DOUBLE(DOUBLE);
DOUBLE _ceil_DOUBLE(DOUBLE);
DOUBLE _cos_DOUBLE(DOUBLE);
```

```
DOUBLE _cosh_DOUBLE(DOUBLE);
DOUBLE _exp_DOUBLE(DOUBLE);
DOUBLE _fabs_DOUBLE(DOUBLE);
DOUBLE _floor_DOUBLE(DOUBLE);
DOUBLE _fmod_DOUBLE(DOUBLE,DOUBLE);
DOUBLE _fmod_DOUBLE(DOUBLE,int);
DOUBLE __huge_val_DOUBLE(void);
DOUBLE __IsNan_DOUBLE(DOUBLE);
DOUBLE _ldexp_DOUBLE(DOUBLE,int);
DOUBLE _log10_DOUBLE(DOUBLE);
DOUBLE _log_DOUBLE(DOUBLE);
DOUBLE _modf_DOUBLE(DOUBLE,DOUBLE);
DOUBLE _pow_DOUBLE(DOUBLE,DOUBLE);
DOUBLE _roundf_DOUBLE(DOUBLE);
```

6.1.2 An Example

```
#include <stdio.h>

main()
{ DOUBLE x;
  int i;

  x = _f2D_DOUBLE(12.34);
  printf("%D\n", x);
  i = 2;
  x = _add_DOUBLE(x,_i2D_DOUBLE(i));
  printf("%D\n",x);
}
```

Note: *To print the output properly for* ***DOUBLE*** *variable, JDK 1.5.0 (also known as JDK 5.0) are required.*

6.1.3 Passing DOUBLE value to JAVA method

You can call Java method and pass Double value as a parameter to a Java method. You have to use **_DBL1(DOUBLE)** function during the passing of Double value to a Java method. The following example illustrates the passing of a DOUBLE value as a parameter to a Java method:

```
#include <stdio.h>

cont()
{ DOUBLE Dval;

  Dval = DOUBLE_java("invokestatic java/lang/Math/cos", "D", "D",
  _DBL1(_f2D_DOUBLE(45.0)));
  printf("Dval = %D\n", Dval);
}

main()
{
  cont();
}
```

Notice that **_DBL1()** needs to be used when passing the DOUBLE value returned by **_f2D_DOUBLE(45.0)**.

PART III: AMPC™ MOBILE

7. **AMPC™ Mobile**

7. AMPC™ Mobile

This document describes the creation of a self-contained Microsoft® Windows Installer file for the Microsoft® Windows Mobile application. This file automates the deployment of the application instead of directly copying the appropriate CAB file to your device.

*** This release is currently available only for the Microsoft® Windows® Platform**

7.1 System Requirements

Please make sure that your host computer meets the following minimum system requirements before installing **AMPC™ Mobile**:

Devices	System Requirements
Processor	Intel x86 based processor or compatible
Operating System	**AMPC™ Mobile** can only be installed on the following systems: • Microsoft® Windows® XP Professional • Microsoft® Windows® XP Home Edition
Memory (RAM)	Minimum : • 128 Megabytes. It is not advisable to run the application with less RAM It is best recommended to have 256 of RAM
Hard disk drive	At least 700 MB of hard disk space for installation • **AMPC™ Mobile** : Approximately 40 MB • J2SE 1.4.2_x : Approximately 430 MB • Microsoft .NET Framework : Approximately 60 MB • Microsoft ActiveSync : Approximately 20 MB • IBM J9 JVM : Approximately 140 MB
CD-ROM drive	Required

Initially, before you start developing your mobile application with **AMPC™ Mobile**, please ensure the following software packages are installed:

Software	Software Requirements
J2SE 1.4.2_x	Download J2SE 1.4.2_x at http://java.sun.com
Microsoft® ActiveSync	Install Microsoft ActiveSync 4.0 or newer to synchronize Windows Mobile/Pocket PCs with your computer. Download the latest version at: http://www.microsoft.com/windowsmobile/downloads/activesync41.mspx
IBM J9 JVM	Install J9 JVM from IBM on your target device in order to provide a platform for your applications deployment J9 JVM can be downloaded at : http://www-128.ibm.com/developerworks/websphere/zones/wireless/weme_eval_runtimes.html
Microsoft® .NET Framework Version 2.0	Download the .NET Framework Version 2.0 at http://www.microsoft.com/downloads/details.aspx?FamilyID=0856EACB- 4362-4B0D-8EDD-AAB15C5E04F5&displaylang=en

7.2 Installation

AMPC™ Mobile installer can be downloaded at http://www.axiomsol.com. Please log in as an administrator to allow the installation mode and then proceed with the installation.

Once the process of downloading has been completed, double-click the **AMPC™ Mobile** setup file for the installation of all the packages into the specified directory '**C:\Program Files\AMPC**'.

To install the license-key information, go to **Start > Program Files > Axiomatic Multi-Platform C Mobile > Install AMPC License Key**. Please read the license agreement as stated in the License.rtf carefully. Type 'yes' if you agree to the terms prescribed. Upon completion, restart your computer before you start using **AMPC™ Mobile**.

Complete uninstallation of **AMPC™ Mobile** could only be done manually through the **Add/Remove Programs**.

7.3 Using AMPC™ Mobile

The following are the steps to develop and deploy your Microsoft® Windows Mobile/Pocket PCs applications.

- Create the mobile application using **AMPC™ Mobile**
- Create a shortcut (*.lnk) file and a JAR file to enable users to launch the

application from the target device.

- Create an information file (*.inf) to describe all your resources and all the CE setups required.
- Package the application into a CAB (Cabinet) file.
- Create a setup (*.ini) file which will then be used by CE App Manager to install application.
- Finally, create a self-extracting application executable (*.exe) which will install your device application from the desktop.

7.3.1 Step 1 - Getting Started

Getting started, to create a new project, go to **Project > New Project** and name the project name. Click **OK** to proceed. The name of the project file should be the same as the source file name that contains the main function.

Write your mobile application. Save your project before proceeding. Set the compiler options by selecting **Settings > Compiler Options** from the project-menu and a dialog box will appear. Tab on **AMPC Memory Model** and choose **Pico – 0.5 MB** or **Nano – 1 MB**. Click **OK** and **build/compile** your application. If there is no error, run your application to ensure that it is working correctly.

The next step is to install **AMPC™ Mobile** on your PocketPC/Windows Mobile device. Select **Mobile Device > Install AMPC Mobile**. A setup wizard will appear. Click **Next** and click **Finish** if you accept the terms prescribed. Once clicked, **AMPC™ Mobile** will be installed on your device. Choose location to install **AMPC™ Mobile** on your device and tab **Install** to proceed.

Figure 7.1

7.3.2 Step 2 - Create a shortcut and a JAR file

Create a shortcut (.lnk) for your application. This is done by going to **Mobile Device >> Create JAR Files**. Fill in the text boxes with appropriate values.

For example:

1. Application Name will be your project name
2. Set path to where **AMPC™ Mobile JAR** files on your target device reside
   Eg: \Program Files\AMPC Mobile\ampc_mobile.jar
3. Set path to where your application JAR files on your target device reside
   Eg: \Program Files\yourappdirectory\yourapp.jar

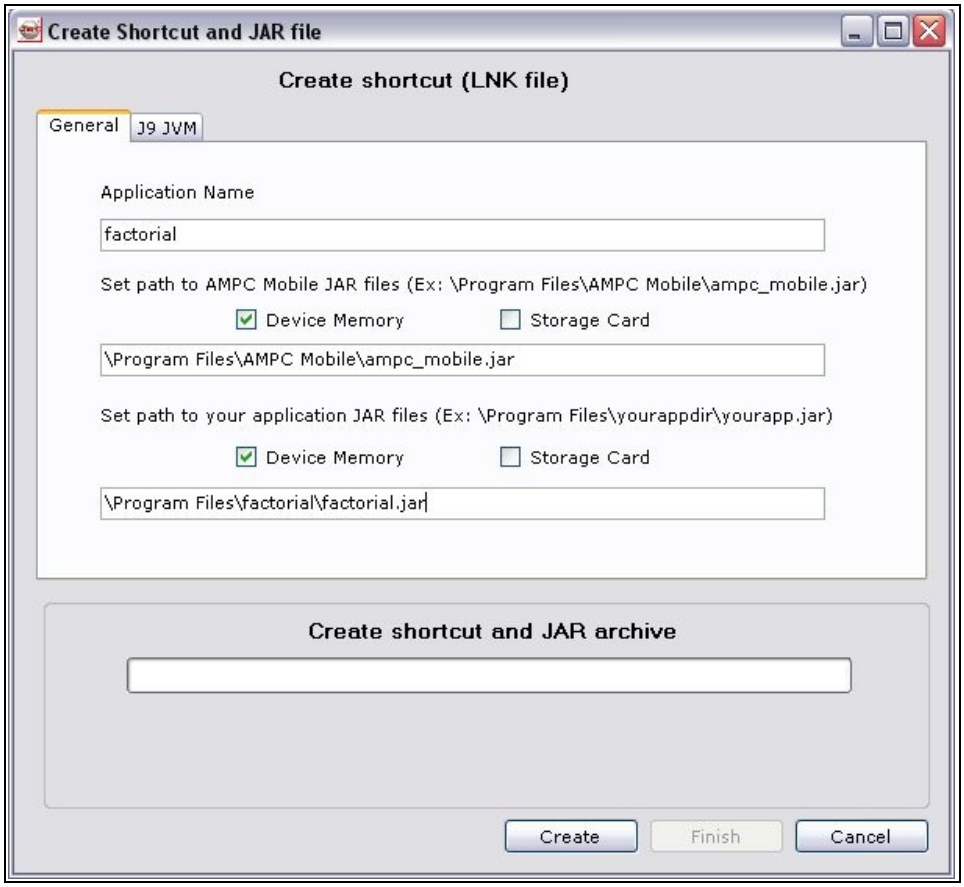

Figure 7.2

4. Tab on **J9 JVM** and set path to where J9 JVM on your target device reside. Check J9 version.

   Eg: \Storage Card\Program Files\J9\PPRO11\j9w.exe or \Program Files\ J9\PPRO11\j9w.exe

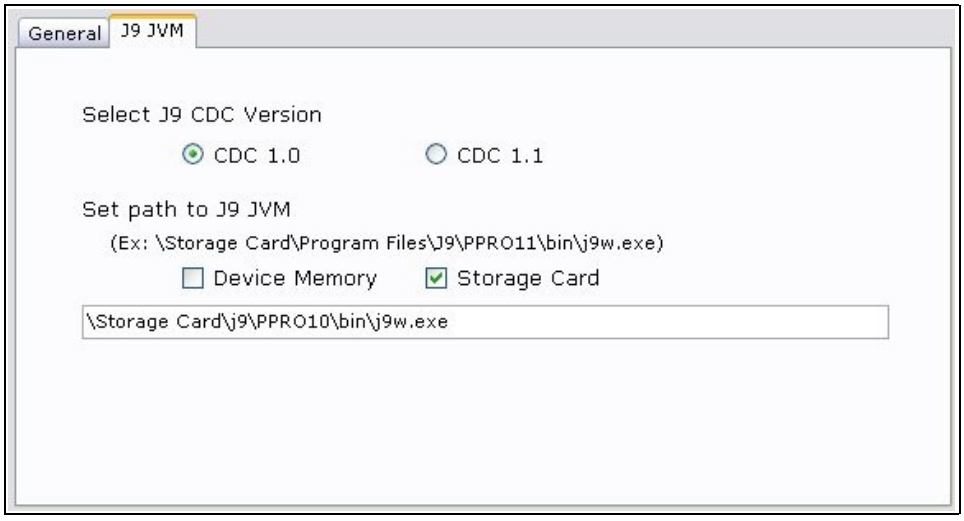

Figure 7.3

5. Once completed, click **Create** to generate a shortcut file and a JAR file. Click **Finish** when all the files have been created.

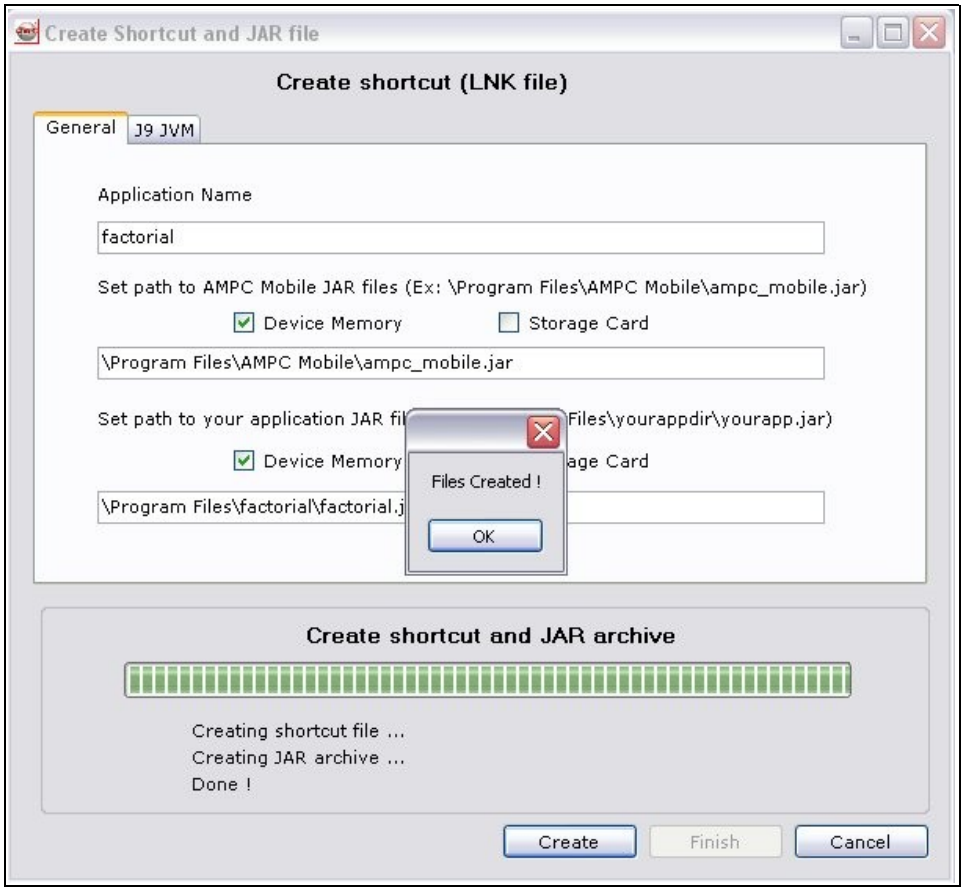

Figure 7.4

6. The next step is to generate a CAB file. A CAB (Cabinet) file is a self-extracting archive file that contains installation instructions and all of the files required by your application (this includes your dependencies such as DLLs, resources, help files, etc)

7.3.3 Step 3 - Create an INF file and package the application into a CAB file

Go to **Mobile Device** and select **Install Application** in the project-menu to start creating information file (.inf). A '**Create Setup Information File (INF)**' box will appear. Fill in each text box in each tab with appropriate values. This INF file is editable.

For example:

1. Check either Windows NT or Windows 95 for the Signature Name.
2. Type the provider's name, for example your company name.

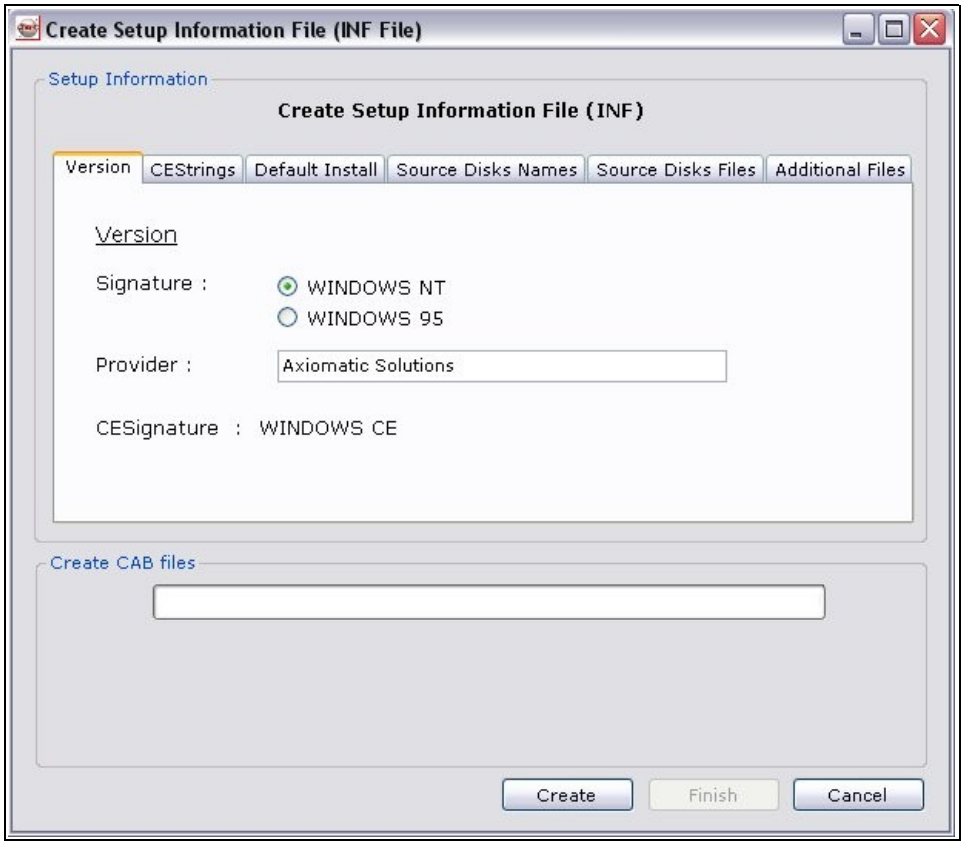

Figure 7.5

3. Choose installation directory on your target device from the combo box provided.

   Eg: %CE1%/%AppName% will be \Program Files\factorial

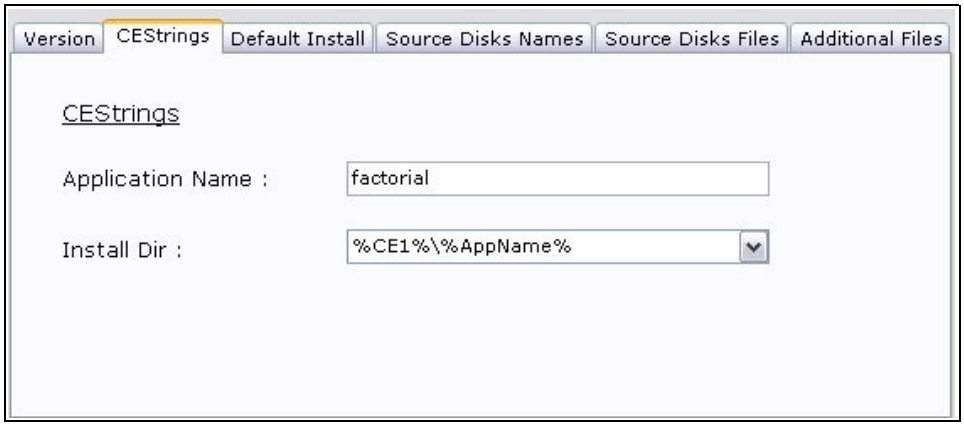

Figure 7.6

Substitutions	Directories
%CE1%	\Program Files
%CE2%	\Windows
%CE3%	\Windows\Desktop
%CE4%	\Windows\Startup
%CE5%	\My Documents
%CE6%	\Program Files\Accessories
%CE7%	\Program Files\Communications
%CE8%	\Program Files\Games
%CE9%	\Program Files\Pocket Outlook
%CE10%	\Program Files\Office
%CE11%	\Windows\Programs
%CE12%	\Windows\Programs\Accessories
%CE13%	\Windows\Programs\Communications
%CE14%	\Windows\Programs\Games
%CE15%	\Windows\Fonts
%CE16%	\Windows\Recent
%CE17%	\Windows\Favorites

The InstallDir should be the same as the path you set for your application JAR files in the previous form.

4. Default Install: Check the checkbox available in this tab.

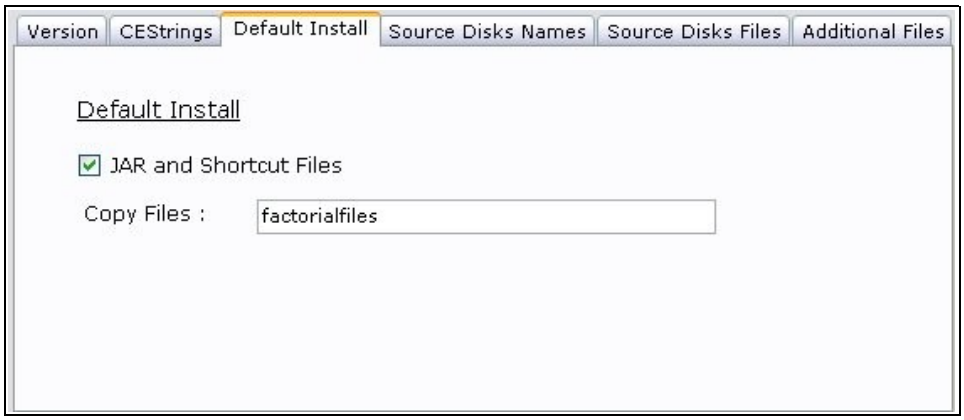

Figure 7.7

5. Select location to where your project source resides on your desktop.

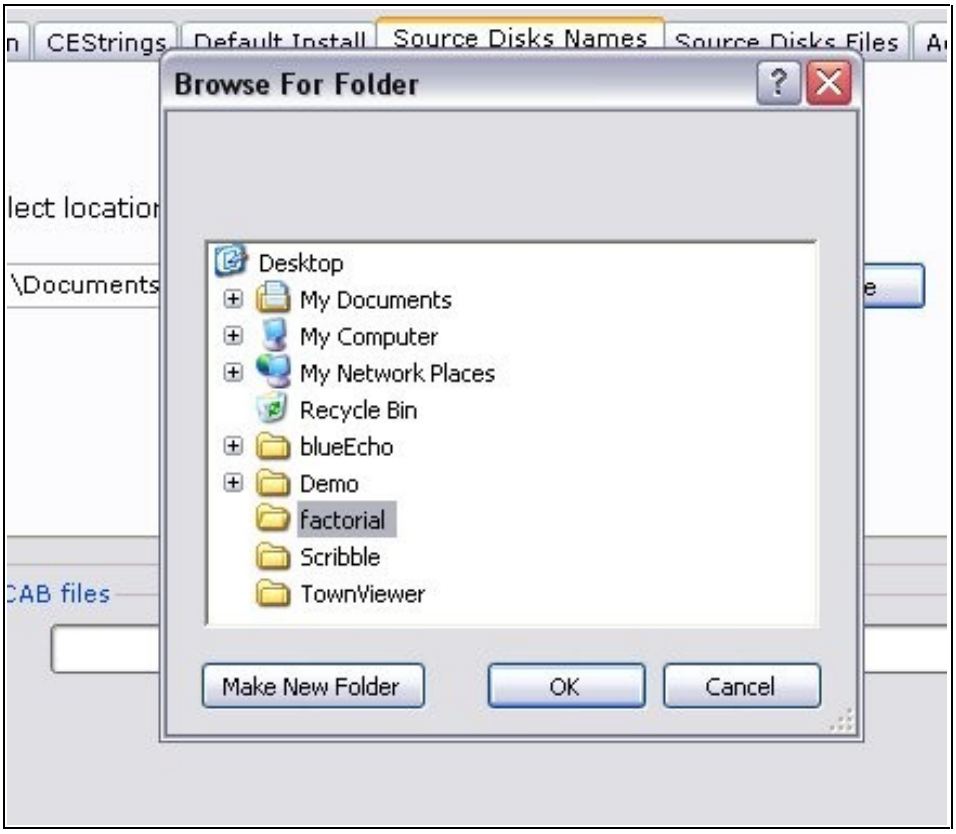

Figure 7.8

6. Select your application shortcut's name and your JAR file's name.

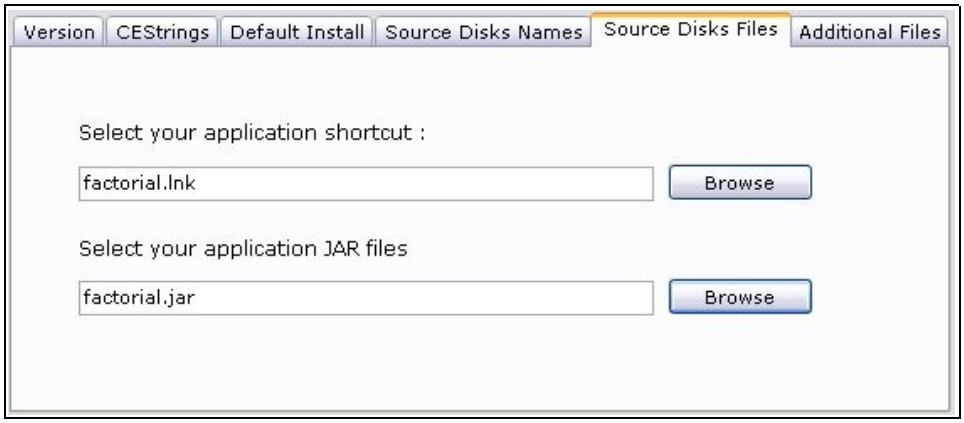

Figure 7.9

7. If you have any additional files to be added to your mobile application (such as image files etc), tab on **Additional Files** and select them. Use Shift-Click or Control-Click to select multiple files.

Figure 7.10

8. Once completed, click **Create** to generate information file (.inf) and a CAB file. Click **Finish** to complete the creation of a CAB file.

7.3.4 Step 4 - Create a setup (.ini) file and install your application on the device

Start installing your application by going to **Mobile Device > Install Application**. Fill in the text boxes available in '**Create INI File**' form. This INI file is editable.

For example:

1. Component will be your project name. So does for uninstall field.
2. Describe your application in the Description field.
3. Select your cabfile's name.

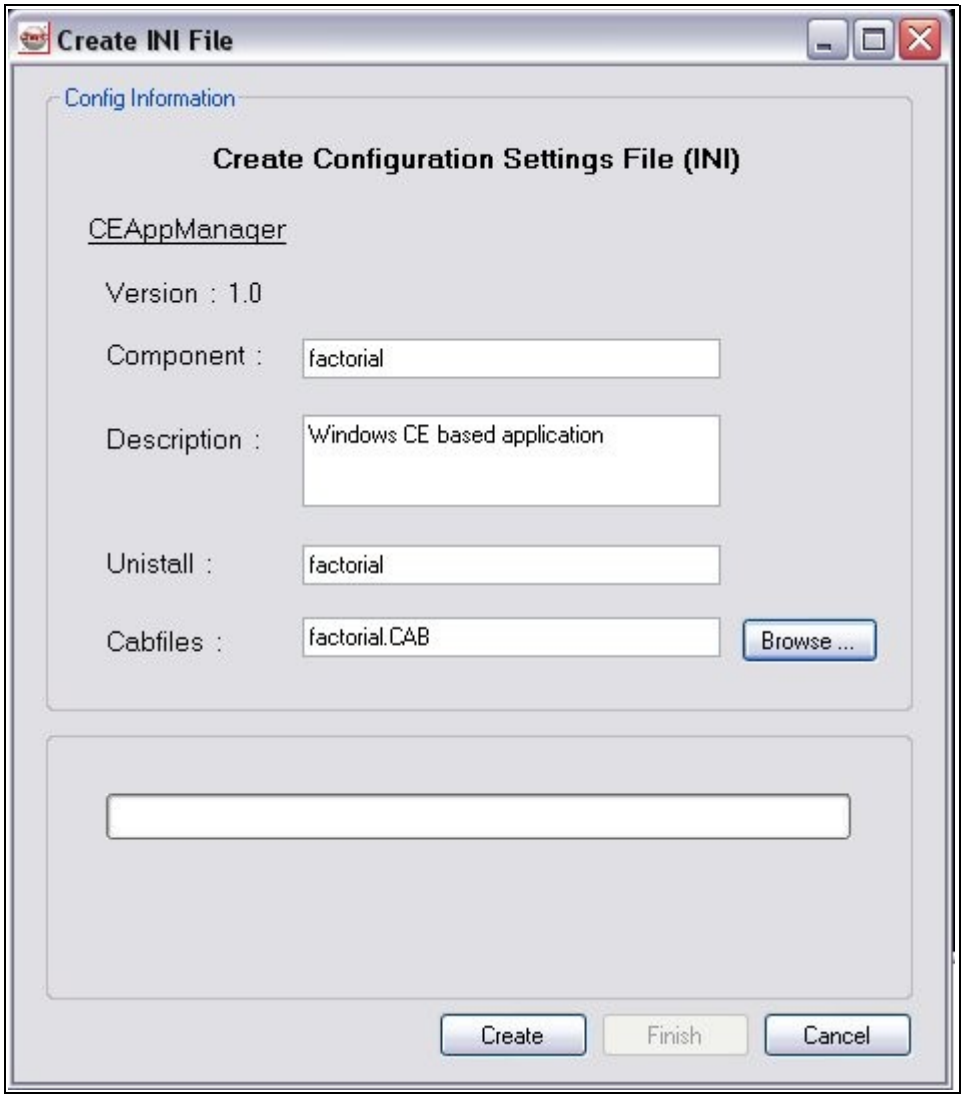

Figure 7.11

4. This form will generate a setup (.ini) file. Once the file has been created, click **Finish** to install your application on your device.

7.3.5 Step 5 - Un-install

The application and the .CAB file you created can be uninstalled. You can uninstall them by using PocketPC/Windows Mobile directly or from your desktop. To uninstall from PocketPC/Windows Mobile device, select **Remove Programs** from the **Settings** application in the **Start Menu**.

Alternatively, you can uninstall the application from the **Control Panel > Add or Remove Programs** and click the **Remove** button.

PART IV: AMPC™ API

8. **AMPC™ Header Files**

9. **AMPC™ Standard**

10. **AMPC™ Graphics**

11. **AMPC™ Network**

12. **AMPC™ Database**

8. AMPC™ Header Files

8.1 List of AMPC Header Files

- acc2jvm.h
- ansidecl.h
- ansi.h
- assert.h
- ctype.h
- errno.h
- ext_fmt.h
- fcntl.h
- features.h
- float.h
- getopt.h
- GLOBALS.h
- io.h
- jni.h
- limits.h
- locale.h
- loc_incl.h
- loc_time.h
- math.h
- network.h
- rusagestub.h
- setjmp.h
- signal.h
- softfloat.h
- stdarg.h
- stddef.h
- stdio.h
- stdlib.h
- string.h
- time.h
- unistd.h
- xalloc.h
- xstate.h
- xstrxfrm.h
- yfuns.h
- yvals.h
- AMPC
    - em-wm.h
    - header.h
    - optimize.h
    - size.h

- bits
    - confname.h
    - environments.h
    - posix_opt.h
    - types.h
    - typesizes.h
    - wordsize.h
- gnu
    - stubs.h
- gui
    - gui.h
- sql
    - driver.h
    - driverextras.h
    - ini.h
    - log.h
    - lst.h
    - odbcinst.h
    - odbcinstext.h
    - sql.h
    - sqlext.h
    - sqltypes.h
    - sqlucode.h
    - uodbc_stats.h
- sys
    - errno.h
    - mman.h
    - stat.h
    - times.h
    - types.h
    - wait.h

9. AMPC™ Standard

9.1 I/O Functions (stdio.h)

void clearerr(FILE *stream)
int fclose(FILE *stream)
int feof(FILE *stream)
int ferror(FILE *stream)
int fflush(FILE *stream)
int fgetc(FILE *stream)
int fgetpos(FILE *stream, fpos_t *pos)
char *fgets(char *str, int n, FILE *stream)
FILE *fopen(const char *filename, const char *mode)
int fprintf(FILE *stream, const char *format, ...)
int fputc(int char, FILE *stream)
int fputs(const char *str, FILE *stream)
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream)
FILE *freopen(const char *filename, const char *mode, FILE *stream)
int fscanf(FILE *stream, const char *format, ...)
int fseek(FILE *stream, long int offset, int whence)
int fsetpos(FILE *stream, const fpos_t *pos)
long int ftell(FILE *stream)
size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream)
int getc(FILE *stream)
int getchar(void)
char *gets(char *str)
void perror(const char *str)
int printf(const char *format, ...)
int putc(int char, FILE *stream)
int putchar(int char)
int puts(const char *str)
int remove(const char *filename)
int rename(const char *old_filename, const char *new_filename)
void rewind(FILE *stream)
int scanf(const char *format, ...)

void setbuf(FILE *stream, char *buffer)
int setvbuf(FILE *stream, char *buffer, int mode, size_t size)
int sprintf(char *str, const char *format, ...)
int sscanf(const char *str, const char *format, ...)
FILE *tmpfile(void)
char *tmpnam(char *str)
int ungetc(int char, FILE *stream)
int vprintf(const char *format, va_list arg)
int vfprintf(FILE *stream, const char *format, va_list arg)
int vsprintf(char *str, const char *format, va_list arg)

9.2 Character Class Tests Functions (ctype.h)

int isalnum(int character)
int isalpha(int character)
int iscntrl(int character)
int isdigit(int character)
int isgraph(int character)
int islower(int character)
int isprint(int character)
int ispunct(int character)
int isspace(int character)
int isupper(int character)
int isxdigit(int character)
int tolower(int character)
int toupper(int character)

9.3 String Manipulation Functions (string.h)

void *memchr(const void *str, int c, size_t n)
int memcmp(const void *str1, const void *str2, size_t n)
void *memcpy(void *str1, const void *str2, size_t n)
void *memmove(void *str1, const void *str2, size_t n)
void *memset(void *str, int c, size_t n)
char *strcat(char *str1, const char *str2)

char *strchr(const char *str, int c)
int strcmp(const char *str1, const char *str2)
char *strcpy(char *str1, const char *str2)
size_t strcspn(const char *str1, const char *str2)
char *strerror(int errnum)
size_t strlen(const char *str)
char *strncat(char *str1, const char *str2, size_t n)
int strncmp(const char *str1, const char *str2, size_t n)
char *strncpy(char *str1, const char *str2, size_t n)
char *strpbrk(const char *str1, const char *str2)
char *strrchr(const char *str, int c)
size_t strspn(const char *str1, const char *str2)
char *strstr(const char *str1, const char *str2)
char *strtok(char *str1, const char *str2)

9.4 Mathematical Functions (math.h)

double acos(double x)
double asin(double x)
double atan(double x)
double atan2(doubly y, double x)
double ceil(double x)
double cos(double x)
double cosh(double x)
double exp(double x)
double fabs(double x)
double floor(double x)
double fmod(double x, double y)
double frexp(double x, int *exponent)
double ldexp(double x, int exponent)
double log(double x)
double log10(double x)
double modf(double x, double *integer)
double pow(double x, double y)
double sin(double x)

double sinh(double x)
double sqrt(double x)
double tan(double x)
double tanh(double x)

9.5 Utility Functions (stdlib.h)

void abort(void)
int abs(int x)
int atexit(void (*func)(void))
double atof(const char *str)
int atoi(const char *str)
long int atol(const char *str)
void *bsearch(const void *key, const void *base, size_t nitems, size_t size,
int (*compar)(const void *, const void *))
void *calloc(size_t nitems, size_t size)
div_t div(int numer, int denom)
void exit(int status)
void free(void *ptr)
char *getenv(const char *name)
long int labs(long int x)
ldiv_t ldiv(long int numer, long int denom)
void *malloc(size_t size)
void qsort(void *base, size_t nitems, size_t size, int (*compar)(const void *,
const void*))
int rand(void)
void *realloc(void *ptr, size_t size)
void srand(unsigned int seed)
double strtod(const char *str, char **endptr)
long int strtol(const char *str, char **endptr, int base)
unsigned long int strtoul(const char *str, char **endptr, int base)
int system(const char *string)

9.6 Program Diagnostic Function (assert.h)

void assert(int expression)

9.7 Variable Argument List Functions (stdarg.h)

void va_start(va_list ap, last_arg)
type va_arg(va_list ap, type)
void va_end(va_list ap)

9.8 Non-Local Jump Functions (setjmp.h)

int setjmp(jmp_buf environment) (not implemented)
void longjmp(jmp_buf environment, int value) (not implemented)

9.9 Signal Functions (signal.h)

void (*signal(int sig, void (*func)(int)))(int)
int raise(int sig)

9.10 Time, Date & Other System Related Functions (time.h)

char *asctime(const struct tm *timeptr)
clock_t clock(void)
char *ctime(const time_t *timer)
double difftime(time_t time1, time_t time2)
struct tm *gmtime(const time_t *timer)
struct tm *localtime(const time_t *timer)
time_t mktime(struct tm *timeptr)
size_t strftime(char *str, size_t maxsize, const char *format,
const struct tm *timeptr)
time_t time(time_t *timer)

9.11 Setting Location Specific Functions (locale.h)

struct lconv *localeconv(void)

char *setlocale(int category, const char *locale)

9.12 Non ANSI-C Functions

void itoa (int value, char c[]) – (stdlib.h)

void ltoa (long int value, char buffer[]) – (stdlib.h)

ldiv_t ldiv2(long int numer, long int denom) – (stdlib.h)

long int lseek (int fildes, long int offset, int whence) – (stdio.h)

int strcoll(const char *str1, const char *str2) – (string.h)

size_t strxfrm(char *str1, const char *str2, size_t n) – (string.h)

9.13 Additional Features

Shortcircuit

Stderr

Stdout

Stdin

10. AMPC™ Graphics

```
void _add_radiobutton_to_buttongroup(gui_Object *an_object,
        gui_Object *container);
void _clear_kb_EVENT();
void _clear_mouse_CLICKED_status();
void _clear_mouse_EVENT();
void _clear_mouse_PRESSED_status();
void _clear_mouse_RELEASED_status();
void _clearRect(int x, int y, int width, int height);
void colorchooser(gui_Object *container, int *red, int *green, int *blue);
void _create_button(gui_Object *an_object, gui_Object *container);
void _create_buttongroup(gui_Object *an_object, gui_Object *container);
void _create_checkbox(gui_Object *an_object, gui_Object *container);
void _create_combobox(gui_Object *an_object, gui_Object *container);
void _create_list(gui_Object *an_object , gui_Object *container, char str[],
        int visible, int selmode);
void _create_passwordfield(gui_Object *an_object, gui_Object *container,
        int col);
void _create_popupMenu(gui_Object *an_object, gui_Object *container);
void _create_radiobutton(gui_Object *an_object, int status,
        gui_Object *container);
void _create_textfield(gui_Object *an_object, gui_Object *container, int col,
        int editable);
void _create_window(gui_Object *an_object)void drawLine(int x1, int y1,
        int x2, int y2, int RGB_Red, int RGB_Green, int RGB_Blue);
void drawOval(int x, int y, int width, int height, int RGB_Red, int RGB_Green,
        int RGB_Blue);
void drawRect(int x, int y, int width, int height, int RGB_Red, int RGB_Green,
        int RGB_Blue);
void drawRoundRect(int x, int y, int width, int height, int arcWidth,
        int arcHeight, int RGB_Red, int RGB_Green, int RGB_Blue);
void _get_effective_event(int *effective_event_count, int *effective_ID,
        char **effective_menu_item_str);
```

```
char _get_kb_MODE();
char _get_kb_READ();
int _get_kb_READ_status();
void _get_list_selectedindex(gui_Object *an_object, int indx);
void _get_mouse_CLICKED_status();
void _get_mouse_pos(int *cur_X, int *cur_Y, int *which_button);
void _get_mouse_pos_DRAGGED(int *cur_X, int *cur_Y, int *which-button);
void _get_mouse_pos_MOVED(int *cur_X, int *cur_Y);
int _get_mouse_PRESSED_status();
int _get_mouse_READ_status();
int _get_mouse_RELEASED_status();
void _get_passwordfield_value(gui_Object *an_object, char **passwdval);
int _getPixel(int x, int y);
void separateRGB(int *RGB_Red, int *RGB_Green, *RGB_Blue, unsigned color);
void _set_backgroundcolor(gui_Object *an_object, int color);
void _set_list_listdata(gui_Object *an_object, gui_Object *cp_object);
void _set_textfield_editable(gui_Object *an_object , int editable);
void _set_textfield_text(gui_Object *an_object, char *str);
char *show_input_dialog(char *a_str);
void show_message_dialog(char *str1);
void _window_set_backgroundcolor(gui_Object *an_object, int color);
void _window_set_backgroundcolor_rgb(gui_Object *an_object, int red,
        int green, int blue);
```

11. AMPC™ Network

```
#define SERVER 0
#define THREAD 1
#define CLIENT 2
#define CLIENTTHREAD 3

int accept(char *clnthost);
int checkmessage();
int checksocket();
int gethostaddr(char *buf);
char *gethostbyaddr(char *abuf, int alen, int atype);
char *gethostbyname(char *buf);
int gethostcname(char *buf);
int gethostname(char *buf);
int getsocket(int index);
void listen_and_accept();
void sendrecvclient();
extern int recv(int * connID, char * buffer, int buflen, int flags);
extern int send(int connID, char * msg, int msglen, int flags);
int serversocket(int port);
int socket(char * ipaddr, int port);
int socketclose(int flags);
```

12. AMPC™ Database

SQLAllocConnect (SQLHENV envHandle , SQLHDBC *connectHandle);

- Allocates a connection handle that deals you with the actual database connection.

SQLAllocEnv (SQLHENV *envHandle);

- Allocates an environment handle.

SQLAllocHandle (SQLSMALLINT handleType, SQLHANDLE inputHandle,
SQLHANDLE *outputHandlePtr);

- Obtains a handle.

SQLAllocStmt (SQLHDBC connectHandle , SQLHSTMT *stmtHandle);

- Obtains a statement handle.

SQLBindCol (SQLHSTMT stmtHandle, SQLUSMALLINT iCol,
SQLSMALLINT dataCType, SQLPOINTER targetValue,
SQLINTEGER *targetValueBuffLength, SQLINTEGER *lengthOrIndicator);

- Binds application variables to columns in the result set.
- Supported C data types (for column number iCol)
  - SQL_C_BINARY
  - SQL_C_BIT
  - SQL_C_CHAR
  - SQL_C_DATALINK
  - SQL_C_DOUBLE
  - SQL_C_FLOAT
  - SQL_C_LONG
  - SQL_C_NUMERIC
  - SQL_C_SHORT
  - SQL_C_TINYINT
  - SQL_C_TYPE_DATE
  - SQL_C_TYPE_TIME
  - SQL_C_TYPE_TIMESTAMP

SQLCancel (SQLHSTMT stmtHandle);

- Cancels the processing on a statement.

SQLCloseCursor (SQLHSTMT stmtHandle);

- Closes a cursor that has been opened on a statement and discard pending results.

SQLConnect (SQLHDBC connectHandle , SQLCHAR *dataSourceName,
SQLSMALLINT dataSourceLength, SQLCHAR *userID,
SQLSMALLINT userIDLength, SQLCHAR *password,
SQLSMALLINT passwordLength);

- Establishes a connection to specific driver by data source name (DSN), user ID and password.
- Data source name (DSN) is the name or alias name of the database to which you are connected. For instance : ”jdbc:mysql://localhost/ employees”.The actual content of DSN is loosely specified as jdbc:<subprotocol>:<subname>. The subprotocol identifies which driver to use and the subname provides the driver with any required connection information – usually the local host name and the database name.

SQLDescribeCol (SQLHSTMT stmtHandle, SQLUSMALLINT iCol,
SQLCHAR *columnName, SQLSMALLINT columnNameMaxLength,
SQLSMALLINT *columnNameStringLength, SQLSMALLINT *sqlType,
SQLUINTEGER *columnPrecision, SQLSMALLINT *columnScale,
SQLSMALLINT *nullable);

- Describes column attributes (column name, type, precision, scale, nullability).

SQLDisconnect (SQLHDBC connectHandle);

- Closes the connection that is associated with the database connection handle.

SQLEndTran (SQLSMALLINT handleType, SQLHANDLE handle,
SQLSMALLINT completionType);

- Commits or rolls back a transaction.
- For handleType , specify one of the following values :

    - SQL_HANDLE_DBC for connection handle.
    - SQL_HANDLE_ENV for environment handle.
- For completionType argument , use one of the following values:
    - SQL_COMMIT to commit a transaction.
    - SQL_ROLLBACK to roll back a transaction.

SQLError (SQLHENV envHandle, SQLHDBC connectHandle,
SQLHSTMT stmtHandle, SQLCHAR *sqlState,
SQLINTEGER *nativeError, SQLCHAR *errorMsg,
SQLSMALLINT errorMsgMax, SQLSMALLINT *errorMsgLength);

- Retrieve error information.

SQLExecDirect (SQLHSTMT stmtHandle, SQLCHAR *sqlString, SQLINTEGER sqlStringLength);

- Prepares and executes an SQL Statement directly.

SQLExecute (SQLHSTMT stmtHandle);

- Executes an SQL statement.

SQLFetch (SQLHSTMT stmtHandle);

- Fetch the next row.

SQLFreeConnect (SQLHDBC connectHandle);

- Releases the connection handle.

SQLFreeEnv (SQLHENV envHandle);

- Releases the environment handle.

SQLFreeHandle (SQLSMALLINT handleType, SQLHANDLE handle);

- Releases environment, connection or statement handle.
- For handleType , specify one of the following values :
    - SQL_HANDLE_STMT to free the statement handle.
    - SQL_HANDLE_DBC to free the connection handle.
    - SQL_HANDLE_ENV to free the environment handle.

SQLFreeStmt (SQLHSTMT stmtHandle, SQLUSMALLINT option);

- Ends statement processing, closes the associated cursor, discards pending results, and frees all resources that are associated with the statement handle.
- Supported options:
  - SQL_CLOSE
  - SQL_ DROP

SQLGetConnectAttr (SQLHDBC connectHandle, SQLINTEGER attribute,
SQLPOINTER valuePtr, SQLINTEGER buffLength,
SQLINTEGER *stringLengthPtr);

- Returns the value of a connection attribute.
- Currently supported attributes (for attribute):
  - SQL_ATTR_ACCESS_MODE
  - SQL_ATTR_AUTOCOMMIT
  - SQL_ATTR_CURRENT_CATALOG
  - SQL_ATTR_LOGIN_TIMEOUT
  - SQL_ATTR_TXN_ISOLATION

SQLGetConnectOption (SQLHDBC connectHandle, SQLUSMALLINT attribute,
SQLPOINTER valuePtr);

- Returns the value of a connection attribute.
- Refer to SQLGetConnectAttr for a complete list of currently supported connection attributes.

SQLGetCursorName (SQLHSTMT stmtHandle, SQLCHAR *cursorName,
SQLSMALLINT cursorMax, SQLSMALLINT *cursorLength);

- Returns the cursor name that is associated with a statement.

SQLGetData (SQLHSTMT stmtHandle, SQLUSMALLINT iCol,
SQLSMALLINT dataCType, SQLPOINTER targetValue,
SQLINTEGER targetValueMaxLength, SQLINTEGER *lengthOrIndicator);

- Returns part or all of one column of one row of a result set.
- Refer to SQLBindCol for currently supported C data types.

SQLGetDiagRec (SQLSMALLINT handleType, SQLHANDLE handle,
SQLSMALLINT recordNum, SQLCHAR *sqlState,
SQLINTEGER *nativeError, SQLCHAR *errorMsg,
SQLSMALLINT buffLength, SQLSMALLINT errorMsgLength);

- Returns additional diagnostic information.
- For handleType , specify one of the following values:
  - SQL_HANDLE_DBC for connection handle.
  - SQL_HANDLE_STMT for statement handle.
  - SQL_HANDLE_ENV for environment handle.

SQLGetEnvAttr (SQLHENV envHandle, SQLINTEGER attribute,
SQLPOINTER valuePtr, SQLINTEGER bufferLength,
SQLINTEGER *stringLengthPtr);

- Returns the value of an environment attribute.
- Currently supported environment attributes:
  - SQL_ATTR_ODBC_VERSION
  - SQL_ATTR_OUTPUT_NTS

SQLGetFunctions (SQLHDBC connectHandle, SQLUSMALLINT function,
SQLUSMALLINT *exists);

- Returns supported driver functions.

SQLGetStmtAttr (SQLHSTMT stmtHandle, SQLINTEGER attribute,
SQLPOINTER valuePtr, SQLINTEGER bufferLength,
SQLINTEGER *stringLengthPtr);

- Returns the value of a statement attribute.
- Currently supported statement attributes (for attribute):
  - SQL_ATTR_MAX_LENGTH
  - SQL_ATTR_MAX_ROWS
  - SQL_ATTR_QUERY_TIMEOUT
  - SQL_ATTR_TXN_ISOLATION
  - SQL_ATTR_STMTTXN_ISOLATION

SQLGetStmtOption (SQLHSTMT stmtHandle, SQLINTEGER attribute,
SQLPOINTER valuePtr);

- Returns the value of a statement attribute:
  - Refer to SQLGetStmtAttr for a complete list of currently supported statement attributes.

SQLNumResultCols (SQLHSTMT stmtHandle, SQLSMALLINT *nCol);

- Returns the number of columns in the result set.

SQLPrepare (SQLHSTMT stmtHandle, SQLCHAR *sqlString,
SQLINTEGER sqlStringLength);

- Prepares an SQL statement for subsequent execution

SQLRowCount (SQLHSTMT stmtHandle, SQLINTEGER *nRows);

- Returns the number of rows that are affected by insert, update, or delete request.

SQLSetColAttributes (SQLHSTMT stmtHandle, SQLUSMALLINT iCol,
SQLCHAR *colStr, SQLSMALLINT colStrLength, SQLUSMALLINT dataType,
SQLUINTEGER colDef, SQLSMALLINT colScale, SQLSMALLINT nullable);

- Sets attributes of a column in the result set.

SQLSetConnectAttr (SQLHDBC connectHandle, SQLINTEGER attribute,
SQLPOINTER valuePtr, SQLINTEGER stringLength);

- Sets a connection attribute.
- Currently supported connection attributes (for attribute):
  - SQL_ATTR_ACCESS_MODE; valuePtr can be either:
    - SQL_MODE_READ_ONLY
    - SQL_MODE_READ_WRITE
    - SQL_MODE_DEFAULT
  - SQL_ATTR_AUTOCOMMIT; valuePtr can be specified whether to be in auto commit or manual commit mode
    - SQL_AUTOCOMMIT_ON
    - SQL_AUTOCOMMIT_OFF
    - SQL_AUTOCOMMIT_DEFAULT

- SQL_ATTR_CURRENT_CATALOG
- SQL_ATTR_LOGIN_TIMEOUT
- SQL_ATTR_TXN_ISOLATION; valid values for valuePtr:
  - SQL_TXN_READ_UNCOMMITTED
  - SQL_TXN_READ_COMMITTED
  - SQL_TXN_REPEATABLE_READ
  - SQL_TXN_SERIALIZABLE

SQLSetConnectOption (SQLHDBC connectHandle, SQLUSMALLINT attribute,
SQLUINtEGER valuePtr);

- Sets a connection attribute.
- Refer to SQLSetConnectAttr for a complete list of currently supported connection attributes.

SQLSetCursorName (SQLHSTMT stmtHandle, SQLCHAR *cursorName,
SQLSMALLINT cursorNameLength);

- Specifies a cursor name.

SQLSetEnvAttr (SQLHENV envHandle, SQLINTEGER attribute,
SQLPOINTER valuePtr, SQLINTEGER stringLength);

- Sets an environment attribute.

SQLSetStmtAttr (SQLHSTMT stmtHandle, SQLINTEGER attribute,
SQLPOINTER valuePtr, SQLINTEGER stringLength);

- Sets a statement attribute.
- Currently supported statement attributes (for attribute):
  - SQL_ATTR_MAX_LENGTH
  - SQL_ATTR_MAX_ROWS
  - SQL_ATTR_QUERY_TIMEOUT
  - SQL_ATTR_TXN_ISOLATION
  - SQL_ATTR_STMTTXN_ISOLATION

SQLSetStmtOption (SQLHSTMT stmtHandle, SQLUSMALLINT attribute,
SQLUINTEGER valuePtr);

- Sets a statement attribute.

SQLTransacf (SQLHENV envHandle, SQLHDBC connectHandle,
SQLUSMALLINT transactionType);

- Commits or rolls back a transaction.
- For transactionType argument , use one of the following values:
  - SQL_COMMIT to commit a transaction
  - SQL_ROLLBACK to roll back a transaction

SQLLoadDriver (SQLHDBC connectHandle, SQLSMALLINT database,
SQLCHAR *driverName);

- Loads and registers a driver in order to connect to database.