



Plaszma Operating System

Technical Overview

Revision 1.6

© Copyright ZiiLABS Pte Ltd. 2009.

All rights reserved worldwide.

The material in this document is the intellectual property of ZiiLABS and is provided solely for information. You may not reproduce this document in whole or in part by any means.

While every care has been taken in the preparation of this document, ZiiLABS accepts no liability for any consequences of its use. Our products are under continual improvement and we reserve the right to change their specification without notice.

The latest version will be available from the ZiiLABS web site.

ZiiLABS products and technology are protected by a number of worldwide patents. Unlicensed use of any information contained herein may infringe one or more of these patents and may violate the appropriate patent laws and conventions.

ZiiLABS and ZMS are trademarks or registered trademarks of ZiiLABS Pte Ltd.

OpenGL is a registered trademark of Silicon Graphics, Inc.

All other trademarks are acknowledged and recognized.

Contents

1 Introduction

Who Should Read this Document	1-1
About this Document	1-1

2 Plazma OS Overview

The Plazma Platform	2-1
Plazma Operating System	2-2
Applications	2-2
The Cell Framework	2-3
Board Support Package	2-3
Plazma Hardware	2-3
Zii EGG	2-3
Bundled Applications	2-5
SDK	2-5

3 The Cell Framework

ApplicationsCell	3-2
GraphicsCell	3-2
3DCell	3-3
MediaCell	3-3
AudioCell	3-4
ImageCell	3-4
CommsCell	3-5
MathCell	3-5
DataCell	3-5
SensesCell	3-6
ServicesCell	3-6
DNACell	3-6
Sequencer	3-6

4 Plazma Software Development

The Desktop	4-1
-----------------------	-----

Creating an Application	4-1
Development Tools	4-2
Eclipse IDE	4-2
Compiler	4-2
GDB Debugger	4-3
CMake	4-3
Layout Studio	4-3
Resource File Compiler	4-4
Sample Applications	4-4
Update History	4-5

1 Introduction

The Plaszma Platform consists of hardware and software optimized for portable multimedia and networked applications. The Plaszma SDK provides all the tools, libraries and documentation needed to write native applications that take full advantage of the Plaszma Platform's capabilities.

Who Should Read this Document

The *Plaszma OS Technical Overview* is an introductory guide for developers who are considering or starting development for the Plaszma platform. This document should enable you to:

- Familiarize yourself with the Plaszma Platform and its capabilities
- Understand the types of application you can develop for the Plaszma platform
- Learn about the technology framework and how it can be used in your application
- Know where to go to get further information

About this Document

This document includes the following sections:

- **“Plaszma OS Overview” on page 2-1:** Provides an overview of the Plaszma Platform hardware and software, and SDK.
- **“The Cell Framework” on page 3-1:** Explains in greater detail the core technologies and services available for use by applications.
- **“Plaszma Software Development” on page 4-1:** Describes the application development process for the Plaszma Platform and the tools provided in the SDK.

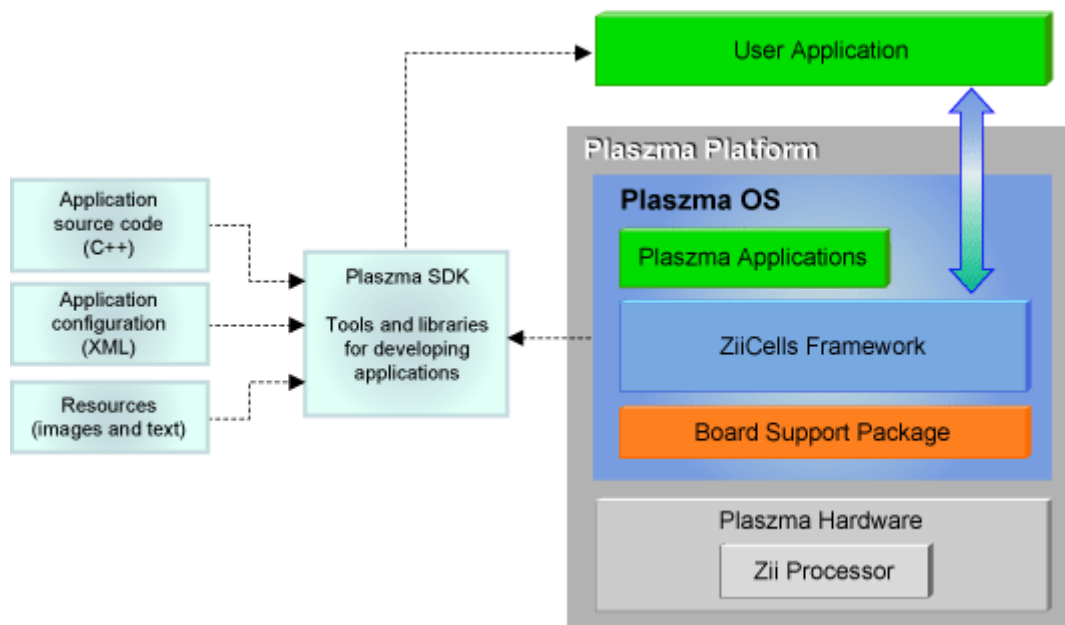
2 Plazma OS Overview

The Plazma Platform consists of Plazma hardware based on the Zii Processor, the Plazma Operating System (OS) and applications.

The Plazma OS includes support for implementing applications with touch-screen based graphical user interface (UI). It also provides a rich range of built-in functions for handling multimedia, displaying high-performance 3D graphics and creating networked applications.

The Plazma SDK provides all the essential developer tools, libraries, and documentation needed to develop applications for devices powered by the Plazma Platform.

Figure 2-1. The Plazma Platform and SDK



The Plazma Platform

The main features of the Plazma Platform are:

- Rich multimedia support: the software supports a large number of standard media formats; the hardware has enough processing performance to handle these at HD resolution.
- Full support for web-based content in applications: HTML, Flash, Javascript etc.

- Advanced 2D and 3D graphics support with hardware acceleration.
- State of the art touch-screen graphical user interface.
- Supports internationalization and localization of applications through UTF-8 encoding and locale-specific resource files.
- Very low power consumption makes it ideal for mobile devices.
- The hardware has support for wide range of interfaces and peripherals devices.
- Based on a high-performance SIMD processor for media, graphics and math processing. This has the performance and flexibility to support current and emerging standards.
- Applications can take full advantage of the functionality and performance of the device through a high-level programming model.
- Support for multiple applications and services running in the background.
- Familiar industry-standard programming tools and APIs. Applications are written in C++ using the gcc compiler.

Plaszma Operating System

The Plaszma OS runs on Plaszma hardware and integrates three layers of software: applications, the Cell Framework and the board support package.

Applications

A number of standard applications are supplied as part of the Plaszma OS. This includes a Media Player, Email client, VoIP application, Maps and Instant Messaging.

There are three types of program which can be run in the Plaszma OS environment:

- **Applications:** Applications are programs which have a graphical user interface that perform specific tasks (for example, play video or send email). They appear on the Desktop and the user can run them by clicking on the icon. An application can be minimized and continue to run in the background.
- **Applets:** Applets appear in the Desktop's Control Panel and are typically used to configure user preferences for an application or the hardware.
- **Services:** Services are programs that do not have a user interface and are run by the Desktop at startup, they continue to run in the background to provide services to other applications. For example, a program running in the background could allow a customer support engineer to access the device remotely in order to solve user problems.

Developers can customize and extend the platform by writing their own applications, applets or services in C++.

The Cell Framework

The most important part of the Plaszma OS is a set of application programming interfaces (APIs) which provide access to the full capabilities of the Plaszma hardware. These APIs are organized in *Cells* of related functions.

Wherever possible, these libraries are based on widely-used, industry-standard APIs to ease the task of porting or writing new applications for the Plaszma Platform. These libraries are optimized for the high-performance multimedia capabilities of the Zii processor. The Plaszma OS integrates and extends these libraries to provide a consistent programming model for portable applications.

Because these libraries have been ported to the powerful array-processing architecture of the Zii Processor, they give “close to the metal” performance for accelerating functions such as media processing and 3D graphics, through high level abstractions.

The developer only needs to understand the APIs exposed by the Cell Framework to get full access to the capabilities of the underlying Zii hardware. These libraries will interface with the appropriate drivers and services at the kernel layer. Such abstraction will ensure that the applications developed will not be affected by changes to the underlying hardware. The standard C++ libraries are also available to the developer.

See “The Cell Framework” on page 3-1 for more information about the range of functions supported by the Plaszma OS.

Board Support Package

The Plaszma OS manages the processing and IO resources on the Plaszma hardware. The board support package (BSP) for each hardware platform provides a Linux kernel and all the necessary device drivers for the hardware devices.

Applications do not interact directly with the board support package but are able to access the full performance of the Plaszma hardware platform using the high-level functions provided in the Cell Framework.

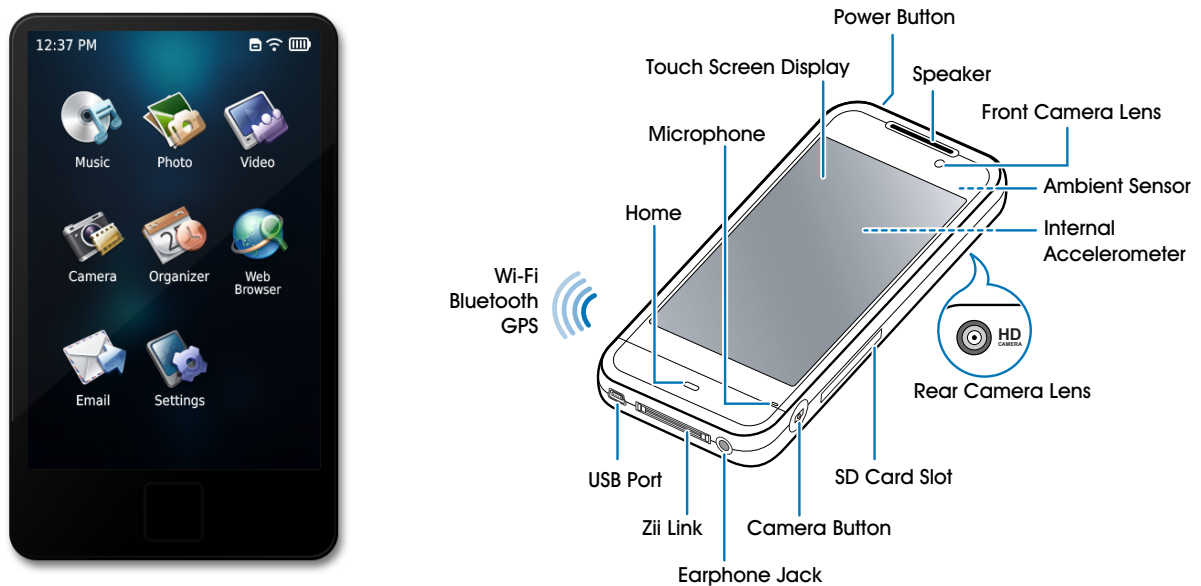
Plaszma Hardware

Plaszma hardware is based on the Zii Processor. Any given Plaszma hardware platform will add extra peripherals and memory and provide external interfaces and user interface devices. ZiiLABS provide the Zii EGG and other evaluation modules. We also expect third parties to develop Plaszma hardware products for specific application areas.

Zii EGG

One example of Plaszma hardware is the Zii EGG. This is a complete, mobile handheld device with touch-screen display, powerful multimedia capability and a wide range of interfaces, including wireless.

Figure 2-2. Zii EGG hardware



The Zii EGG is a typical hardware device based on the Plaszma Platform. This is a mobile, hand-held device with the following features:

- Weight: 104g
- 3.5 inch capacitive multi-touch LCD display, HVGA 480x320 resolution
- 32GB flash memory
- Wireless connectivity: Wi-Fi (802.11b/g) and Bluetooth 2.1 + EDR
- GPS for location-based services
- Two cameras: HD (720p) and VGA for still and video capture
- Multiple interfaces: USB, HDMI, composite video, audio in/out, SD memory card
- Sensors: accelerometer, ambient light

Bundled Applications

The following applications are provided with the Plaszma OS:

- Music Player
- Video Player
- Photo Browser
- Web Browser
- Email
- Calendar
- Address Book
- Notes

SDK

To support third party innovation and allow rapid development of applications, a fully featured SDK is available to interested developers.

The Plaszma OS framework and the Plaszma SDK provide full access to the capabilities of the hardware platform in a device independent way. The Plaszma SDK is based on industry-standard APIs and tools such as the gcc C++ compiler and the gdb debugger. The SDK is currently supported on Linux; we recommend the Ubuntu Long Term Support (LTS) release 8.04.

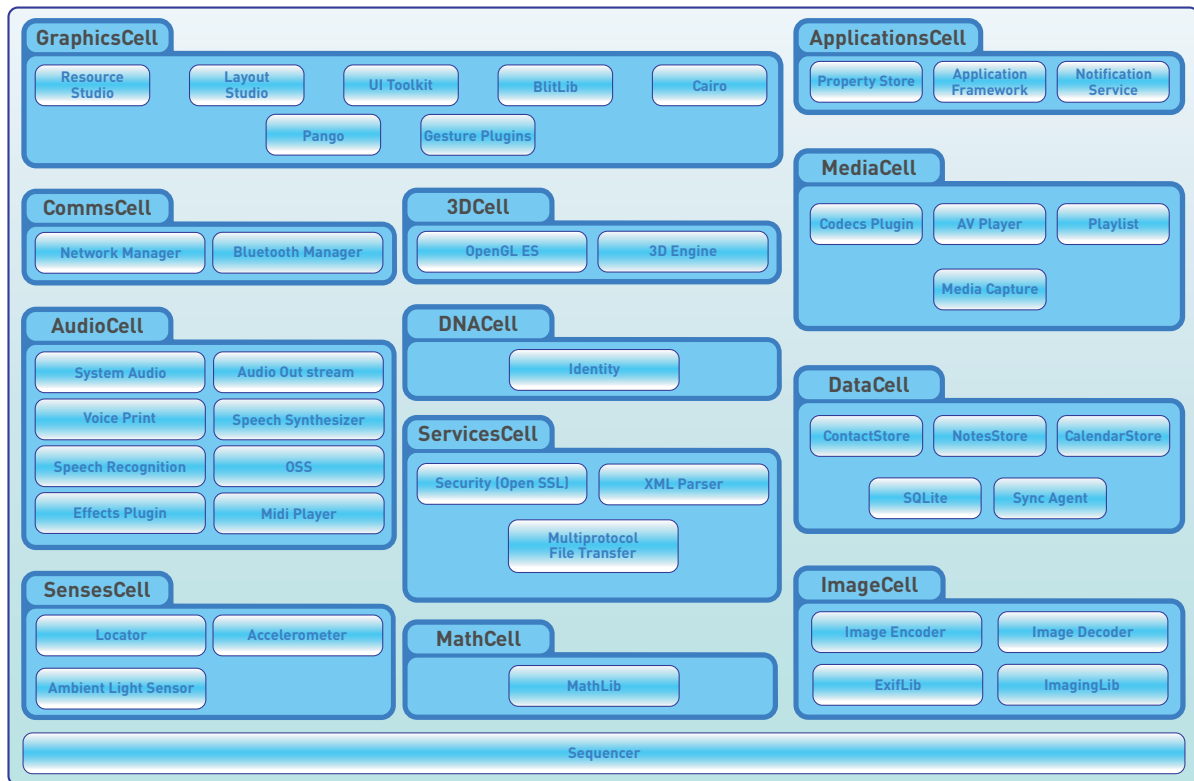
With the comprehensive suite of tools provided in the Plaszma SDK, developers are ensured a smooth and fast development process, enabling them to focus on delivering valuable applications to end users.

See "Plaszma Software Development" on page 4-1 for more detail on the development process and the SDK.

3 The Cell Framework

To simplify the task of locating the functions needed in an application, the libraries are organized in *Cells*. Each Cell contains a group of related libraries and APIs. This makes it easy to identify the components needed to build an application. Note that the Cells just provide a logical way to organize and document the libraries and classes in the Plazma OS, there is no corresponding “cell” hierarchy in the software structure.

Figure 3-1. Contents of Cell Framework



Although there is no hierarchical structure within the Cell Framework, all applications will use the ApplicationsCell and the GraphicsCell. These implement the user interface, handle basic interactions with the OS and support commonly used application functions. The rest of the Cells provide access to the core features of the platform such as high definition media playback and 3D acceleration.

ApplicationsCell

The ApplicationsCell is a key component of the Framework. It provides the basic infrastructure and C++ objects required to implement an application in the Plaszma OS.

Application Framework	The Application Framework provides the base classes used to implement an application. A user application is created as an object derived from one of these classes.
Property Store	The Property Store provides a standard interface for an application to manage settings such as user preferences.
Notification Service	The Notification Service allows an application to interact with the desktop and react to other system events.

GraphicsCell

One of the major features of the Plaszma Platform is the user interface based on a touch-screen display. The GraphicsCell makes it easy to build applications using this interface. It also provides access to 2D drawing functions and resource management. Graphics and font rendering are accelerated by the array processing capabilities of the Zii Processor. Full support for HTML, Javascript and Flash is provided by the Opera rendering engine which can be used by any application to provide web-based, interactive content.

UI Toolkit	The UI Toolkit provides a rich set of graphical user interface widgets for creating touch-screen based applications. The user interface can also include multimedia, 3D graphics and web components.
Layout Studio	The Layout Studio manages the layout of user interface components and allows the application to be easily configured for displays of different resolutions.
Resource Studio	The Resource Studio provides a uniform method to access image and text assets associated with an application. It also supports internationalization by providing a standard interface to load the string resources for a locale from a resource file.
Cairo	Cairo is a widely used vector graphics library. It is used by the UI Toolkit and can also be used directly to generate 2D graphics in a window. Drawing operations in Cairo are accelerated on the Zii Processor array when appropriate.
Pango	Pango adds support for rendering internationalized text using Cairo. This includes support for complex scripts and right-to-left languages.
Gesture Plugins	The Plaszma UI recognizes a number of gestures for user interaction. This can be extended with new gestures using this API.
BlitLib	The Bit Block Transfer Library provides a number of basic "bit blt" primitives such as copy, stretch and fill.

3DCell

Many important applications need to display 3D images. The Plaszma Platform supports high performance 3D rendering, accelerated by the array processing core of the Zii Processor.

OpenGL ES and EGL	The OpenGL ES v1.1 standard is the most commonly used API for embedded 3D graphics. Rendering is accelerated on the Zii Processor array.
3D Engine	The 3D Engine is a higher-level cross-platform 3D graphics library. It allows models generated in 3ds Max to be easily imported and rendered with minimal programming.

MediaCell

The ability to create applications with rich multimedia content is a key feature of the Plaszma Platform. This makes the MediaCell an important part of the Cell Framework, presenting high level APIs to access the platform's media processing capabilities. It comes with a variety of audio and video codecs for encoding and decoding of popular media file formats.

AV Player	This component allows applications to play audio and video media files.
Playlist	This module supports an application by managing a list of items to be played. This is used by the Media Player and could also be used, for example, in a Photo Album application.
Media Capture	Images and video can be captured in a variety of formats from any of the cameras in the Zii hardware. Audio can be captured from the built-in microphone or from a headset.
Codecs Plugin	The Codecs Plugin interface allows further codecs to be incorporated into the system to extend the range of formats supported.

The audio and video container file formats supported are:

- MP4 (and M4A for audio)
- MOV
- AVI
- ASF (WMV).

The audio and video codecs supported are shown in [Table 3-1](#).

Table 3-1. Supported media codecs

Video	Support	Audio	Support
H.264 (BP)	Encode + Decode	MP3	Decode
H.264 (MP)	Decode	WMA9	Decode
WMV9/VC1 (MP/AP)	Decode	AAC LC	Encode + Decode
MPEG-4 (SP/ASP)	Decode	AAC HE	Decode

Table 3-1. Supported media codecs (Continued)

Video	Support	Audio	Support
Motion JPEG	Decode	FLAC	Decode
		WAV (LPCM)	Decode
		WAV (ADPCM/PCM)	Encode

AudioCell

The AudioCell libraries provide support to the AV Player and extend the audio processing functionality beyond the basics provided by the MediaCell.

Audio Out Stream	The Audio Out Stream API handles the playback of audio data coming in from a network connection.
System Audio	This API is used to play system sounds and user defined sounds.
Voice Recognition	The Voice Recognition module allows interaction with the device via voice recognition. The device can be trained to recognize voice commands specified by users and trigger events tied to respective commands.
Speech Synthesizer	The Speech Synthesizer module generates an audio stream based on input text. For example, the Speech Synthesizer function could be used to convert the contents of an email message to speech.
Voice Print	The Voice Print module can generate a user's voice signature through a simple training process. This can be used to verify users by their voice signature.
Open Sound System	The Open Sound System (OSS) is a standard Linux interface for sound devices. This simplifies the task of porting applications using OSS to the Plazma OS.
Midi Player	The Midi Player is used to playback MIDI music files.
Effects Plugin	The Effects Plugin interface allows new audio effects to be added to the standard effects included in the AV Player.

ImageCell

The ImageCell is used by applications to handle the storage and display of image files.

Image Decoder	This module provides functions to load, decode and display images in a variety of formats.
Image Encoder	The Image Encoder allows image data to be encoded in a variety of standard file formats.

ExifLib	EXIF is a standard image file format specification used by the digital camera industry to encode data about images in JPEG files. This library allows EXIF data to be extracted from and added to JPEG image files.
ImagingLib	The Imaging Library provides a set of common image processing primitives.

CommsCell

The CommsCell manages access to network and wireless connectivity.

Network Manager	The Network Manager provides automatic network detection and configurations for a device. It can be used to manage, for example, the Wi-Fi interface.
Bluetooth Manager	The Bluetooth Manager provides applications with the ability to detect connected Bluetooth devices and query their capabilities.

MathCell

The MathCell includes a number of math functions which are accelerated on the Zii Processor.

MathLib	The Math library provides primitives to accelerate compute intensive math functions, such as FFT, DFT and matrix multiplication.
----------------	--

DataCell

The DataCell provides an application with the ability to maintain persistent data using a relational database manager.

Contact Store	The Contact Store provides an application with access to the user's address book information on the device.
Calendar Store	The Calendar Store maintains information about scheduled events in the user's calendar.
Notes Store	The Notes Store provides an application with the ability to record user notes.
Sync Agent	The Sync Agent allows an application to control the synchronization of user data such as Contacts, Calendar and Notes with another device.
SQLite	SQLite is a widely used library that provides an embedded application with access to a relational database using SQL. This also provides the underlying implementation for the other functionality in the DataCell.

SensesCell

The SensesCell provides APIs to communicate with the position sensors in the Plaszma hardware. This uses GPS and accelerometers to determine location, orientation and movement of the device.

Accelerometer	The Accelerometer module provides acceleration data in the x, y and z directions. This could be used, for example, to determine the orientation of the device and hence display output in landscape or portrait mode.
Locator	The Locator interface allows an application to access the device's current latitude and longitude to provide location-based services.
Ambient Light Sensor	The ambient light sensor can be used to determine the optimum brightness of the display, for example.

ServicesCell

The ServicesCell implements the basic network and security protocols used by applications to exchange information with other devices.

File Transfer	This module provides access to the <code>libcurl</code> library so an application can perform data transfer between systems using a variety of standard protocols including http, https and ftp.
Security	The Security module uses the OpenSSL library to allow an application to implement secure communications using SSL and TLS.
XML Parser	The XML Parser uses the standard <code>libxml2</code> library to allow an application to parse and create XML documents, and transform XML to other formats such as HTML.

DNACell

The DNACell manages information specific to a device.

Identity	The Identity API provides access to device and hardware specific information such as USB ID, OS version, manufacturer and model of the device.
-----------------	--

Sequencer

The array processors in the Plaszma hardware provide the compute power behind the whole Plaszma OS. Many CPU intensive tasks can be executed on the array, leaving the ARM cores free to do other tasks and so increasing the processing power of the whole Plaszma platform.

In any application, there may be many functions that need to use the array: media decoding, 3D graphics, user interface rendering, and so on. The Sequencer manages the scheduling of all these different tasks on the array.

4 Plazma Software Development

An application, as seen by the user, consists of one or more user interface views. Each of these is a window containing one or more UI components or *widgets*. Widgets can be as simple as a button or checkbox, or more complex user interfaces such as dialogs. There are also *container widgets* which are used to manage the layout of other widgets in the window.

From the programmer's point of view, windows and widgets are represented by C++ objects of the corresponding class. A separate XML file is used to define the layout of the widgets within a window.

A Plazma application's interaction with the user interface and the operating system is event driven. This means that in order to handle input from the user, or notifications of state changes from the operating system, the programmer writes event handler functions for each of the relevant events.

The Desktop

The desktop environment is the parent of all other programs running on the Plazma OS. It is responsible for application detection, displaying application icons to the user, starting and terminating programs and system status display.

The main components of Desktop are:

- **Launcher:** The Desktop Launcher displays desktop icons and allows user to launch applications.
- **Control Panel:** The Desktop Control Panel displays applets, in various categories, so the user can configure the behavior of the hardware and software. There are a number of pre-defined applets as part of the operating system. In addition, applications can provide their own applets.
- **System Tray:** The System Tray, or notification area, is part of the Desktop Launcher view. It displays system status information such as time, battery level and so on. Applications can communicate with Desktop to display or hide the notification area.

Creating an Application

The Plazma application framework provides base classes which are used to create an application, applet or service. An application is created by deriving a class from the appropriate base class, as shown in the following example.

```
class CMyApp: public PlazmaApp
```

```
{  
    ...  
};
```

The program can then create an object of this class and call the `Run()` method to start the application and its user interface, for example:

```
CMyApp *myApp = new CMyApp();  
myApp->Run();
```

The application will define one or more windows for the various views of the application. Each window will contain one or more widgets for interacting with the user. When the user touches a widget, an event is generated and sent to the window's event handler. The event handler can then take the appropriate action. Similarly, event handlers in the main application handle any events from the Desktop or the operating system.

Development Tools

The Plazma SDK is supported on the Ubuntu Linux distribution. This is the preferred desktop development environment as it is an easy to use Linux distribution with a long term support (LTS) policy. The current LTS version of Ubuntu is 8.04.

The developer tool package in the Plazma SDK comprises of both industry-standard tools and proprietary components.

Eclipse IDE

The Eclipse IDE integrates the entire software development process: editing, compiling and debugging of source code. Recommended because of its ease of use and cross-platform support.

Eclipse includes support for:

- code editing
- source-level debugging
- project creation
- managed build for various tool chains
- source code refactoring and code generation

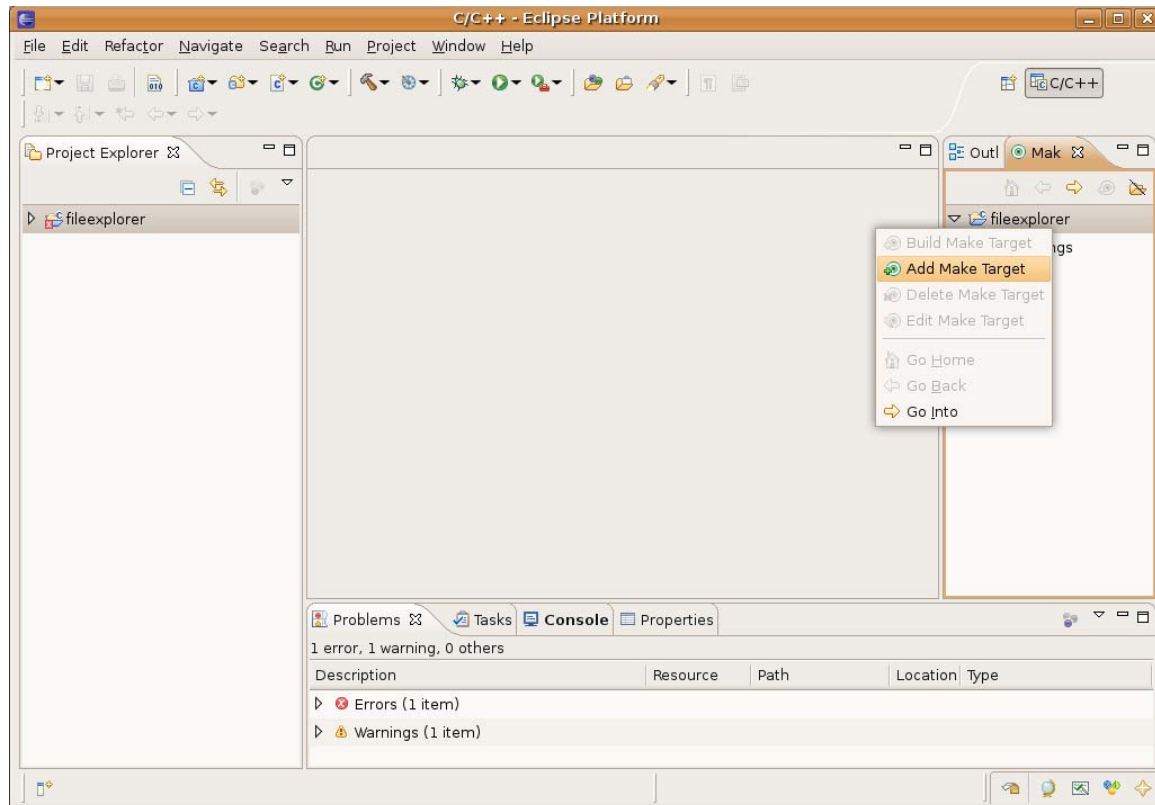
Compiler

The industry-standard gcc toolchain is used to compile the C++ application code and generate executables for the ARM processor cores in the Zii Processor.

GDB Debugger

GDB is the most popular debugger for Linux systems and is available as a plugin in Eclipse. It provides advanced source code and symbolic debugging features to assist in identifying problems. Using a network connection between the Plaszma hardware and the PC running the Eclipse IDE, remote debugging can also be conducted on an application that is running on the device.

Figure 4-1. The Eclipse IDE



CMake

CMake is a cross-platform, open-source build system for generating native makefiles and managing multiple project file types. This tool is used to manage the build process in the Eclipse IDE.

Layout Studio

The Layout Studio simplifies the task of creating graphical user interfaces by using a configuration file to describe the type and position of UI components in a window. Multiple configuration can be defined for an application making it easy to manage changes in display size or orientation.

Resource File Compiler

The resource file compiler converts image and text resource files into the form used by applications. This creates a binary resource file which can be accessed using the Resource Manager.

Sample Applications

A number of sample applications are provided with the SDK. These can be used to learn the basics of developing for the Plaszma OS and as a starting point for custom applications:

- File Explorer
- UI Toolkit Tutorials
 - Widgets and Controls
 - Windows (QWERTY keyboard)

Update History

Date	Revision	Comments
13/07/2009	V1.6	First version released