
Windows Embedded CE 6.0 BSP for MCIMX32 (i.MX32) ADS

User's Guide

WCE600-14
October 18, 2007

07-5508-USRGD-TX30

How to Reach Us:

Home Page:

www.freescale.com

E-mail:

support@freescale.com

USA/Europe or Locations Not Listed:

Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:

Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:

Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:

Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:

Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-521-6274 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2006, 2007. All rights reserved.

Contents

Chapter 1 BSP Installation

1.1	Installing the Windows Embedded Tools and the i.MX32 BSP	1
1.2	Uninstalling the Freescale i.MX32 BSP	2
1.3	Loading the Sample OS Design Solution	2

Chapter 2 BSP Contents and Organization

2.1	System-on-a-Chip Support Package (SOC)	5
2.2	Production Quality OAL (PQOAL)	6
2.3	i.MX32ADS Platform Files	6
2.4	Sample OS Design Solutions	6
2.5	Support Files	6

Chapter 3 Configuring OS Images

3.1	Image Build Type	7
3.2	BSP Environment Variables	7
3.2.1	Configuration Properties Environment Variables	8
3.2.2	Catalog Environment Variables	9

Chapter 4 Building OS Images

4.1	Building the Freescale SOC Libraries	10
4.2	Building Run-Time Images	10
4.2.1	Building the BSP for the First Time	10
4.2.2	Incremental BSP Build	11

Chapter 5 Preparing for Downloading and Debugging

5.1	Serial Debug Messages	12
5.1.1	Desktop Workstation Serial Debug Port	12
5.1.2	Target Serial Debug Port	12
5.1.3	i.MX32ADS Board Configuration	12
5.1.4	i.MX32 CPU Board Settings	13
5.1.5	i.MX32 Base Board Settings	14
5.1.6	MC13783 Board Settings	17
5.2	RealView Toolchain Configuration	18
5.2.1	RVI Configuration	18
5.2.2	RVDEBUG Configuration	18

5.3	EBOOT Installation and Configuration	19
5.3.1	Building an EBOOT Image for NOR Flash	20
5.3.2	Building an EBOOT Image for NAND Flash	20
5.3.3	Loading EBOOT into SDRAM using RVDEBUG	20
5.3.4	Initialize EBOOT Network Configuration	21
5.3.5	Program/Update EBOOT in NOR Flash using Platform Builder	21
5.3.6	Program/Update EBOOT in NAND Flash using Platform Builder	22
5.4	Configuring Ethernet Connection for Downloading and Debugging	23

Chapter 6

Downloading and Debugging Images

6.1	Downloading an Image into SDRAM using EBOOT	25
6.2	Downloading an OS Image into NOR Flash using EBOOT	25
6.3	Running an OS Image from NOR Flash using EBOOT	26
6.4	Downloading an OS Image into NAND Flash using EBOOT	26
6.5	Running an OS Image from NAND Flash using EBOOT	27

Chapter 7

BSP Features

7.1	Supported Features and Device Drivers	28
-----	---	----

About This Book

This document is a user's guide for the Freescale MCIMX32 (i.MX32) Windows Embedded CE 6.0 board support package (BSP). This document will describe how to install the BSP and how to use Microsoft Platform Builder for Windows CE 6.0 to build, download, and debug OS images. Information on the configuration and usage of features included within the Freescale BSP will also be provided.

Audience

This guide is intended for users of Microsoft Platform Builder who want to build and execute Windows CE 6.0 OS images based on the Freescale BSP. The audience also includes Windows CE application developers that want to leverage API interfaces provided by the Freescale BSP.

Organization

This document is organized into the following chapters.

Chapter 1	Describes how to install/uninstall the Freescale BSP.
Chapter 2	Describes the contents and organization of the Freescale BSP.
Chapter 3	Describes how to configure the Freescale BSP for building Windows Embedded CE 6.0 OS images.
Chapter 4	Provides instructions on how to build Windows Embedded CE 6.0 OS images using the Freescale BSP.
Chapter 5	Describes the preparation necessary for downloading and debugging OS images.
Chapter 6	Describes the procedures for downloading and debugging OS images.
Chapter 7	Provides details on the Freescale BSP features included in this release.

Revision History

Revision	Description
WCE600-MX32-E1	Initial revision of this document.
WCE600-MX32-E2	Updated the BSP install and uninstall instructions in Chapter 1 to reflect the use of a Microsoft Installer (.msi) package. Also added images of Platform Builder with the iMX32ADS OS Design Solution loaded.
WCE600-MX32-E3	Updated for the E3 MX32 release. Corrected the SoC directory names in Chapter 2 to include the Vendor and Version fields as required by the current Microsoft naming conventions. Corrected the procedure in Section 3.1 regarding how to select the active build configuration (i.e., Release or Debug). Added documentation for the BSP_WARNLEVEL build environment variable in Table 3-1.
WCE600-14	Minor typographical updates for the latest release.

Suggested Reading

Additional information regarding Microsoft Windows CE, the i.MX32, and the ADS platform can be found in these documents:

- <http://msdn.microsoft.com/embedded/windowsce>
- *HW Getting Started - i.MX31ADS*
- *i.MX32 Reference Manual*

Conventions

Use this section to name, describe, and define any conventions used in the book. This document uses the following notational conventions:

- `Courier monospaced type` indicate commands, command parameters, code examples, expressions, datatypes, and directives.
- *Italic type* indicates replaceable command parameters.
- All source code examples are in C.

Definitions, Acronyms, and Abbreviations

The following list defines the abbreviations used in this document.

ADS	application development system
API	application programming interface
BSP	board support package
CSP	chip support package
DHCP	dynamic host configuration protocol
EBOOT	Ethernet bootloader
EVB	platform evaluation board
FAL	flash abstraction layer
GDI	graphics display interface
ICE	in-circuit emulator
IDE	integrated development environment
IST	interrupt service thread
IPU	image processing unit
KITL	kernel independent transport layer
LVDS	low-voltage differential signaling
MAC	media access control
OAL	OEM adaptation layer
OEM	original equipment manufacturer
OS	operating system



PQOAL	production quality OEM adaptation layer
RVDEBUG	RealView debugger
RVI	RealView ICE
SDC	synchronous display controller
SDRAM	synchronous dynamic random access memory
SoC	system on a chip

References

The following documents were referenced to produce this document.

- *Windows Embedded CE 6.0 Online Help*
- *HW Getting Started - i.MX31ADS*
- *Windows CE BSP Reference Guide*



Chapter 1

BSP Installation

The BSP is distributed as a single Microsoft Installer (.msi) file that includes support for the i.MX32ADS platform. Please refer to the Windows Embedded CE 6.0 *i.MX32 BSP Release Notes* for additional instructions and information before installing and using this BSP.

1.1 Installing the Windows Embedded Tools and the i.MX32 BSP

The following steps will install the Microsoft Windows Embedded CE 6.0 development tools and the Freescale i.MX32 BSP to provide a complete development environment.

1. Install Visual Studio 2005 and the Windows Embedded CE 6.0 Platform Builder plugin from the installation discs. Please refer to the *Release Notes* on the Visual Studio and Platform Builder installation discs for installation instructions.

NOTE

During Platform Builder installation, be sure to select which versions of the operating system to include. The **ARMV4I** version must be installed on the local hard drive for this BSP. All other versions are not required.

NOTE

Refer to the Release Notes for information about any additional Microsoft-supplied QFEs or Service Packs that also need to be installed.

2. If you have previously installed an earlier version of the Freescale i.MX32 BSP, remove any existing BSP files by following the instructions in [Section 1.2, Uninstalling the Freescale i.MX32 BSP](#).
3. Download the Freescale i.MX32 BSP and install the contents into the existing WINCE600 top-level folder (the MSI installer will automatically create the necessary subfolders so that all of the files will be installed into the correct location).

NOTE

During the BSP installation, the following file from the existing Platform Builder installation will be overwritten:

WINCE600\PLATFORM\COMMON\SRC\ARM\dirs

Rename or move this file before installing the BSP if you wish to keep a copy so that it can be restored later if you decide to uninstall the BSP.

1.2 Uninstalling the Freescale i.MX32 BSP

This section describes how to remove an installation of the BSP from the Windows Embedded CE 6.0 source code tree and Platform Builder development environment.

NOTE

Uninstalling the BSP will remove all files that were installed with the MSI installer. If you have made any changes to these files, they will be removed. Be sure to save off any modified files you want to keep before uninstalling the BSP.

The following steps will remove the BSP:

1. Close Platform Builder
2. Either rerun the original MSI installer and select the “Remove” option or open up the “Add or Remove Programs” Control Panel applet, select the Freescale BSP item, and then select “Remove”.
3. Manually remove the remaining BSP files and directories (note that some of these files and directories will still remain after completing the previous step because they contain object files or other generated files that were not part of the original BSP installation):

```
\WINCE600\OSDesigns\iMX32ADSMobility
\WINCE600\PLATFORM\COMMON\SRC\ARM\dirs
\WINCE600\PLATFORM\COMMON\SRC\ARM\FREESCALE
\WINCE600\PLATFORM\COMMON\SRC\SOC\FREESCALE
\WINCE600\PLATFORM\iMX32ADS
\WINCE600\SUPPORT
```

4. Manually remove the residual BSP library files that were created after the BSP was installed. To find the libraries, use Windows Explorer with the search name `*mxarm11*; *mx32*; *pmic*; *mc13783*` for the following library path and remove all the LIB, PDB, and DEF files found by the search:

```
\WINCE600\PLATFORM\COMMON\LIB\ARMV4I
```

5. Restore the original version of the `\WINCE600\PLATFORM\COMMON\SRC\ARM\dirs` file.

1.3 Loading the Sample OS Design Solution

After installing the BSP, you can use the sample OS solutions provided in the BSP package to build a Windows Embedded CE 6.0 OS image.

1. In Platform Builder, select **File** → **Open** → **Project/Solution....**
2. Select the i.MX32 BSP sample workspace by loading the following file:

```
WINCE600\OSDesigns\iMX32ADSMobility\iMX32ADSMobility.sln
```

The process of loading this solution should also automatically load the associated i.MX32 BSP catalog. Simply select the “Catalog Items View” tab to see all of the available OS solution catalog items.

Figure 1-1 shows how Platform Builder should appear after successfully loading the sample iMX32ADSMobility OS Design Solution. Figure 1-2 shows how to view and, if required, modify the catalog items to change the build configuration.

NOTE

The actual Visual Studio display might differ a little bit from what is shown in the figures due to changes in the list of supported devices and features that are in a specific release.

Figure 1-1. Loading the iMX32ADSMobility OS Design Solution

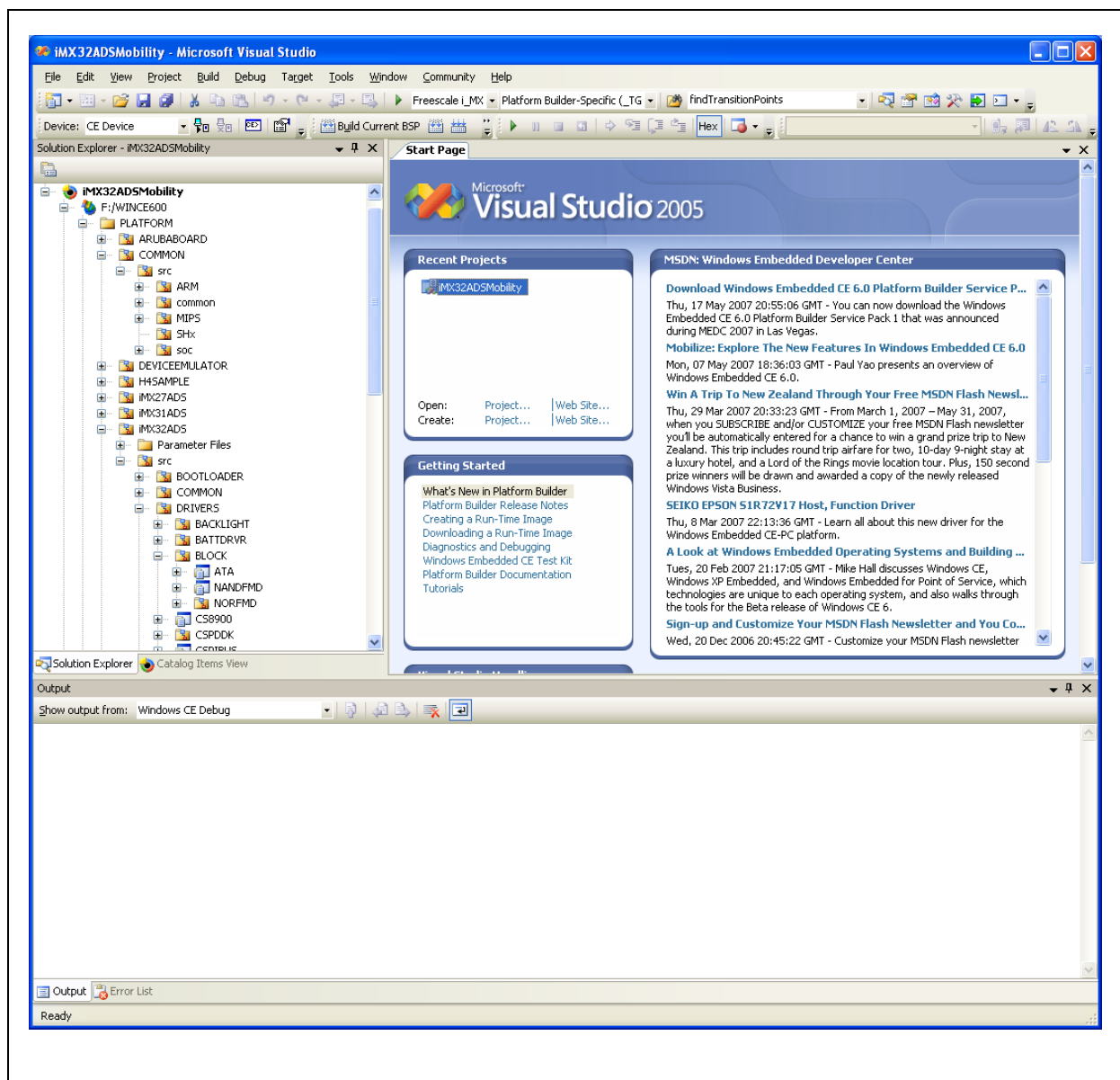
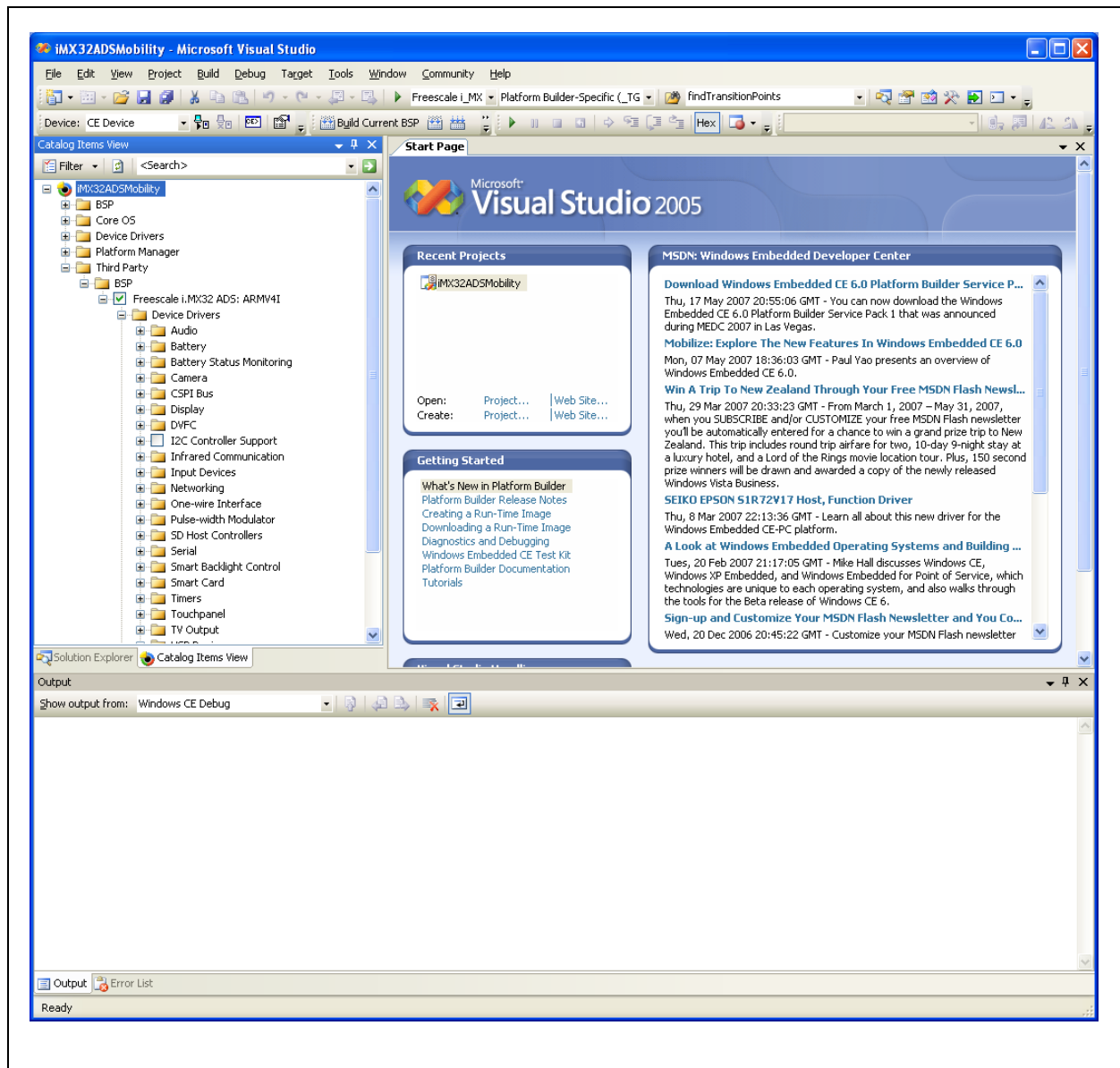


Figure 1-2. Viewing the iMX32ADSMobility Catalog Items



Chapter 2

BSP Contents and Organization

The Freescale BSP is a collection of code and support files that can be integrated into the Microsoft Platform Builder development environment to create Windows Embedded CE 6.0 OS images for the i.MX32-based platforms. A BSP contains the following elements:

- Boot loader for downloading OS images
- OEM Adaptation Layer (OAL) for providing the kernel hardware interface
- Device drivers to support on-chip and on-board peripherals
- Image configuration and build files

The BSP includes a set of directories and files that are installed into an existing Windows Embedded CE 6.0 source tree. The BSP directory structure follows the production-quality OAL (PQOAL) and production-quality drivers (PQD) structure recommended by Microsoft.

The i.MX32 system-on-a-chip (SoC) leverages a common Freescale ARM11-based platform architecture. This platform is found in a series of ARM11-based SoCs available from Freescale. In order to leverage source code that is portable across multiple Freescale ARM11-based SoCs, a common directory called MXARM11 is used to store shared OAL and driver components. This MXARM11 common directory appears in the chip support package and PQOAL directories described below.

2.1 System-on-a-Chip Support Package (SOC)

The Freescale SOC directory contains a collection of chipset-level code that can be leveraged to develop platforms based on the i.MX32 SoC. The driver code and definitions in the SOC directory can be reused in a new platform design without modification. To keep the SOC sources platform agnostic, drivers in the SOC directory utilize hardware abstraction routines that must be ported to a specific platform or board. The SOC source code is compiled into a set of static libraries that are ultimately linked with platform-specific libraries to create drivers for the system.

The following directories contain the SOC source code for the i.MX32:

```
WINCE600\PLATFORM\COMMON\SRC\SOC\FREESCALE\COMMON_FSL_V1
WINCE600\PLATFORM\COMMON\SRC\SOC\FREESCALE\MXARM11_FSL_V1
WINCE600\PLATFORM\COMMON\SRC\SOC\FREESCALE\MX32_FSL_V1
```

SOC driver code in the MXARM11_FSL_V1 directory is reusable across all Freescale ARM11-based SoCs. SOC driver code in the MX32_FSL_V1 directory is reusable across all platforms based on i.MX32.

Finally, there is also an SOC directory for all platform-independent code that is related to the Freescale MC13783 Power Management IC (PMIC). This PMIC is typically used to provide power management, audio recording/playback functionality, and touchpanel and analog-to-digital conversion functions. The platform-independent MC13783 PMIC code can be found in the following directory:

```
WINCE600\PLATFORM\COMMON\SRC\SOC\FREESCALE\PMIC\MC13783_FSL_V1
```

2.2 Production Quality OAL (PQOAL)

Windows Embedded CE 6.0 supports PQOAL components that simplify and shorten the process of developing an OAL. For more information on PQOAL development concepts, refer to the topic “*Production-Quality OAL*” in the *Windows Embedded CE 6.0 Help*.

Where possible, the Freescale BSP leverages the PQOAL architecture and components provided by Microsoft to reduce the OAL code that needs to be modified and maintained by the OEM. In addition, PQOAL components customized for the i.MX32 are available in the following directories:

```
WINCE600\PLATFORM\COMMON\SRC\ARM\FREESCALE\MXARM11
WINCE600\PLATFORM\COMMON\SRC\ARM\FREESCALE\MX32
```

PQOAL code in the MXARM11 directory is reusable across all Freescale ARM11-based SoCs. PQOAL code in the MX32 directory is reusable across all platforms based on i.MX32.

2.3 i.MX32ADS Platform Files

The i.MX32 BSP provides direct support for the interfaces and peripherals found on the i.MX32 Application Development System (ADS) board. All of the driver and OAL content that is specific to the underlying hardware platform is located in the following directory:

```
WINCE600\PLATFORM\iMX32ADS
```

The i.MX32ADS platform directory implements the hardware abstraction routines invoked by driver code in the Freescale SOC directory. In addition, this directory implements certain aspects of the PQOAL that may need to be modified by the OEM for their specific platform.

2.4 Sample OS Design Solutions

Design solutions are used by Platform Builder to encapsulate the OS components and build options necessary for the Windows Embedded CE 6.0 tools to generate an OS image. Default solutions have been included in the BSP within the following directory:

```
WINCE600\OSDesigns\iMX32ADSMobility
```

The solution was created using the **New** → **Project** feature within Platform Builder and utilizing the Custom Device Design Template. For more information about creating an OS solution, refer to the topic “*Creating an OS Design with the Windows Embedded CE OS Design Wizard*” in the *Windows Embedded CE 6.0 Help*.

2.5 Support Files

Support files that complement the BSP source tree are located within the following directory:

```
WINCE600\SUPPORT
```

This directory will be used to store applications and tests targeted for the supported platforms. Support files necessary to configure development tools will also be placed here.

Chapter 3

Configuring OS Images

You can use Platform Builder to select one of the two default build configurations provided with the BSP sample solution. These configurations control the type of OS image (debug or release) that will be generated by the Platform Builder tools. In addition, the build configuration encapsulates all of the platform environment variables and custom build instructions that will be used during OS image creation. This section will describe the configuration options available for the i.MX32 BSP.

NOTE

The sample solution provided within the i.MX32 BSP is properly configured to generate a default image targeted for the i.MX32 ADS hardware. It is not necessary to adjust any of the image configuration settings prior to building the BSP.

3.1 Image Build Type

The type of OS image generated by Platform Builder can be controlled using the **Build → Configuration Manager...** menu item and choosing the appropriate **Active Solution Configuration** item from the drop-down list. The sample workspace provided with the i.MX32 BSP provides the following two image build types:

- **Freescall i_MX32_ADS_ARMV4I_Release**—Retail build that includes KITL and kernel debugger support. This image type provides a smaller image with faster execution at the expense of limited debug capability.
- **Freescall i_MX32_ADS_ARMV4I_Debug**—Debug build that includes KITL and kernel debugger support. This image type provides full debug capability at the expense of a larger image size and slower execution.

NOTE

The Standard toolbar can be enabled/disabled by selecting the **Tools → Customize...** menu item followed by the **Toolbars** tab and then selecting/deselecting the **Standard** toolbar item.

For more information about the build types available for Windows CE, refer to the topic “*Build Configurations*” in the *Windows Embedded CE 6.0 Help*.

3.2 BSP Environment Variables

BSP environment variables control optional BSP support for OS images. The i.MX32 BSP environment variables are controlled using the **Project → Properties → Configuration Properties → Environment** configuration dialog and the i.MX32 BSP Catalog. The following sections will describe the methods of configuring BSP environment variables and provide a summary of the variables supported by the i.MX32 BSP.

3.2.1 Configuration Properties Environment Variables

BSP environment variables can be set for each build configuration within the Platform Builder solution. The variables can be viewed and configured within **Environment** dialog as follows:

1. Open the sample solution for the i.MX32 BSP.
2. From the **Project** menu, choose **Properties**.
3. Expand **Configuration Properties** if necessary and select the **Environment** item.

The following table provides a summary of BSP environment variables supported with the **Environment** settings dialog.

Table 3-1. Supported Platform Settings BSP Environment Variables

Variable Name	Description	Settings
BSP_NOPCIBUS	Used to exclude support for PCI bus.	BSP_NOPCIBUS = 1 Excludes PCI bus support from the OS image. DO NOT remove from sample workspace.
BSP_NOCSPDDK	Used to exclude i.MX32 CSP driver development kit (CSPDDK) support. The CSPDDK is required for most BSP device drivers.	BSP_NOCSPDDK = 1 Excludes CSPDDK from the OS image. Only use this configuration when building an OS design that does not include BSP device drivers. DO NOT define for sample workspace.
BSP_NOAUDIO	Used to exclude support for audio driver.	BSP_NOAUDIO = 1 Excludes audio driver support from the OS image.
BSP_NODISPLAY	Used to exclude support for display driver.	BSP_NODISPLAY = 1 Excludes display driver support from the OS image.
BSP_NOTOUCH	Used to exclude support for touch driver.	BSP_NOTOUCH = 1 Excludes touch driver support from the OS image.
BSP_NOKEYBD	Used to exclude support for keypad driver.	BSP_NOKEYBD = 1 Excludes keypad driver support from the OS image.
BSP_NONLED	Used to exclude support for notification LED driver.	BSP_NONLED = 1 Excludes notification NLED driver support from the OS image.
BSP_NO_TRUST	Excludes trusted environment support to the image.	BSP_NO_TRUST = 1 Excludes trusted environment support from the OS image. DO NOT remove from sample workspace.
BSP_OAL_TIMER32K	Enable/Disable support for the onboard 32 KHz timer.	BSP_OAL_TIMER32K=1 Enables support for the onboard 32 KHz timer which helps to reduce power consumption in the low power modes by allowing the PLLs to be disabled.

Table 3-1. Supported Platform Settings BSP Environment Variables (continued)

Variable Name	Description	Settings
BSP_WARNLEVEL	Selects the compiler warning/verbosity level. See the online help for the /w compiler option for additional information. Note that WARNISERROR=1 is also defined by default in the PUBLIC\COMMON\sources.cmn, PLATFORM\COMMON\sources.cmn, and the PLATFORM\iMX32ADS\sources.cmn files so that any compiler warnings will be treated as build errors.	BSP_WARNLEVEL=4 This means that compiler level 4 warnings will be displayed and treated as build errors if WARNISERROR=1 is also defined.
BUILD_OPTIONS	Specifies an optional set of BSP directories for the build tools	BUILD_OPTIONS = FREESCALE MXARM11_FSL_V1 MX32_FSL_V1 PMIC MXARM11 MX32 Required for proper build of BSP. DO NOT modify or remove from sample workspace.
IMG NAND	Used by EBOOT and OS build files to determine if images are targeted for NAND flash.	IMG NAND = 1 Link EBOOT and OS images for NAND flash. Note: This environment variable is ignored for OS images if you have selected Platform → Settings → Build Options → Write Run-time Image to Flash Memory (IMGFLASH=1) to link the OS image for NOR flash.

3.2.2 Catalog Environment Variables

The i.MX32 BSP utilizes the Platform Builder Windows CE Catalog to allow users to configure BSP components in the OS design. The *Windows CE BSP Reference Guide* describes the environment variables associated with each of the BSP features exposed in the i.MX32 BSP Catalog.

Chapter 4 Building OS Images

This chapter provides instructions for building Windows Embedded CE 6.0 OS images using the BSP.

4.1 Building the Freescale SOC Libraries

The Freescale SOC libraries that support the i.MX32 are generated during the **Build → Advanced Build Commands → Sysgen** or **Build → Advanced Build Commands → Build Current BSP and Subprojects** build procedures. Windows CE ships with pre-built SOC libraries for various ARM processors, but the libraries for the i.MX32 must be built from the sources since they are not included with the standard Microsoft distribution.

Note that the sample OS design solution that is provided is already preconfigured to build the Freescale SOC that is required. That is, the sample i.MX32 OS design solution will automatically build the MX32 SOC sources.

Follow these steps to build the Freescale SOC libraries:

1. Open the sample solution.
2. Select the desired build type discussed in [Section 3.1, Image Build Type](#).
3. Select a **Sysgen** build if you have not yet performed a Sysgen operation before. Otherwise, you may select to just **Build Current BSP and Subprojects** if you have already performed a Sysgen before and just need to rebuild the Freescale SOC libraries.

For a release build type, the SOC libraries will be placed in:

```
WINCE600\PLATFORM\COMMON\LIB\ARMV4I\RETAIL
```

For a debug build type, the SOC libraries will be placed in:

```
WINCE600\PLATFORM\COMMON\LIB\ARMV4I\DEBUG
```

4.2 Building Run-Time Images

The remaining steps to build an OS image follow the standard procedures described in the Platform Builder documentation. For more information, refer to the topic “*Building a Run-Time Image*” in the *Windows Embedded CE 6.0 Help*.

4.2.1 Building the BSP for the First Time

1. Open the sample solution.
2. Select the desired build type discussed in [Section 3.1, Image Build Type](#).
3. Using the **Build → Global Build Settings** menu, configure the build options as follows:
 - **Copy Files to Release Directory After Build** selected
 - **Make Run-Time Image After Build** selected
4. Select **Build → Advanced Build Commands → Sysgen** to start the build process.

For a release build type, the resulting OS image files will be placed in the following directory depending upon the OS design that is being used:

```
WINCE600\OSDesigns\iMX32ADSMobility\RelDir\Freescale_i_MX32_ADS_ARMV4I_Release
```

For a debug build type, the resulting OS image files will be placed in the following directory:

```
WINCE600\OSDesigns\iMX32ADSMobility\RelDir\Freescale_i_MX32_ADS_ARMV4I_Debug
```

4.2.2 Clean Build for the BSP

By default, Platform Builder performs incremental builds of the BSP components, even during the **Sysgen** build procedure. It may be desirable under certain circumstances to force a clean build for the BSP.

A clean build of the all the Freescale BSP components can be accomplished as follows:

1. **Copy Files to Release Directory After Build** selected.
2. **Make Run-Time Image After Build** unselected.
3. Select **Build** → **Advanced Build Commands** → **Rebuild Current BSP and Subprojects** to perform a clean build of the BSP platform directory (including the SOC libraries) and complete the creation of a new OS image.

4.2.2 Incremental BSP Build

The **Sysgen** build phase results in pre-built OS component binaries being copied to the release directory for the current build configuration. It is not necessary to perform a **Sysgen** again unless components are being added or removed from your **OS design**. Instead, the user can perform an incremental build of the Freescale BSP components to quickly build an updated OS image as follows:

1. Open a solution.
2. Select the desired build type discussed in [Section 3.1, Image Build Type](#).
3. Using the **Build** → **Global Build Settings** menu, configure the build options as follows:
 - **Copy Files to Release Directory After Build** selected
 - **Make Run-Time Image After Build** selected
4. Select **Build** → **Advanced Build Commands** → **Build Current BSP and Subprojects** to perform an incremental build of the BSP platform directory (including the SOC libraries) and complete the creation of an OS image.

Chapter 5

Preparing for Downloading and Debugging

The target and development workstation must be properly configured and initialized before OS images can be downloaded and executed. This section will discuss the steps required to prepare the target and development workstation so that Platform Builder can be used to download and debug images on the target.

5.1 Serial Debug Messages

Serial debug messages are used by the boot loader and OS images to report status and error information. In addition, the boot loader uses serial input to allow for user interaction. This section will describe the configuration of the desktop workstation and Freescale BSP to support serial debug messages.

5.1.1 Desktop Workstation Serial Debug Port

Any terminal emulation application can be used to display messages sent from the serial port of the target. Configure your terminal application with the following communications parameters:

- Bits per second: 115200
- Data bits: 8
- Parity: None
- Stop bits: 1
- Flow control: none

5.1.2 Target Serial Debug Port

The i.MX32ADS BSP has been configured to use internal UART1 routed to the UARTC ADS board connector for serial debug message. UARTC on the ADS is the upper DB-9 connector next to the Ethernet port. Connect a standard serial cable between UARTC of the i.MX32ADS and your development workstation.

5.1.3 i.MX32ADS Board Configuration

This section will describe the switch and jumper configuration required for proper operation of the BSP on the i.MX32ADS board. For specific BSP features, the ADS may need to be reconfigured to support the necessary interface. Refer to the *Windows CE BSP Reference Guide* for board configuration required by some BSP features.

NOTE

Refer to the i.MX32ADS documentation for more information on the full set of jumper and switch configurations available.

5.1.4 i.MX32 CPU Board Settings

This section will describe the configuration for the i.MX32 CPU board.

Table 5-1. CPU Board Settings

Jumper/Switch Setting	Configuration
NVCC Power Selection Jumpers	
JP3 = REMOVED JP13 = REMOVED JP4 = REMOVED JP5 = REMOVED JP16 = REMOVED JP23 = REMOVED JP28 = REMOVED JP36 = REMOVED	These jumpers should be removed to support the MC13783 daughter card.
Reset Jumper	
JP2 = 2-3	This jumper setting is only available and required on older CPU boards.
CPU Power Jumpers	
JP12 = 2-3 JP17 = 2-3 JP20 = 2-3 JP32 = 2-3 JP35 = 2-3	Power CPU from MC13783.
DRAM Power Selection Jumper	
JP38 = 2-3	Power DRAM from MC13783.
Reset/Boot Switches SW2	
SW2-8 = OFF	GPIO1_6 not used for tamper detect.
SW2-7 = ON SW2-6 = ON	Connect CPU board reset chip to i.MX32.
SW2-5 = ON SW2-4 = ON SW2-3 = ON SW2-2 = ON SW2-1 = ON	Internal ROM boot. Used initially for connecting with RVDEBUG and downloading EBOOT.
SW2-5 = OFF SW2-4 = ON SW2-3 = OFF SW2-2 = ON SW2-1 = ON	Direct external boot from NOR. Used to execute EBOOT or OS images previously programmed into NOR flash memory.

Table 5-1. CPU Board Settings (continued)

Jumper/Switch Setting	Configuration
SW2-5 = OFF SW2-4 = ON SW2-3 = ON SW2-2 = ON SW2-1 = OFF	Direct external boot from NAND. Used to execute EBOOT or OS images previously programmed into NAND flash memory.
PLL Clock Reference	
JP22 = 1-2	CKIH is PLL clock reference. This configuration <u>cannot</u> be used when booting from NAND.
JP22 = 2-3	FPM is PLL clock reference. This configuration <u>must be</u> used when booting from NAND.
Miscellaneous CPU Board Jumper Selections	
JP33 = INSTALLED JP29 = INSTALLED JP24 = INSTALLED JP19 = INSTALLED JP14 = INSTALLED JP7 = INSTALLED JP8 = INSTALLED JP9 = INSTALLED JP10 = INSTALLED JP11 = INSTALLED JP15 = INSTALLED JP18 = INSTALLED JP21 = INSTALLED JP25 = INSTALLED JP27 = INSTALLED JP31 = INSTALLED JP34 = INSTALLED JP1 = 1-2 JP6 = 1-2 JP26 = 1-2 JP30 = INSTALLED	Factory configuration.

5.1.5 i.MX32 Base Board Settings

This section will describe the configuration for the i.MX32 base board.

Table 5-2. Base Board Settings

Jumper/Switch Setting	Configuration
USB HS Jumper Selections	
JP1 = 2-3 JP2 = 2-3	Select the PHY for VUSB source.
UART Configuration Switches SW1	

Table 5-2. Base Board Settings (continued)

Jumper/Switch Setting	Configuration
SW1-1 = OFF SW1-2 = OFF SW1-3 = OFF SW1-4 = OFF	UART/FIR interfaces are not forced to be enabled. Enable/disable of these interfaces is under software control.
UART/PWM/WDOG Configuration Switches SW2	
SW2-1 = ON	Connect watchdog reset to MC13783.
SW2-2 = OFF	PWM output disconnected from buzzer.
SW2-3 = OFF	UARTC transceiver is enabled.
SW2-4 = ON	UARTC baud is limited to 250kbps
SW2-5 = OFF	External UARTA transceiver is enabled.
SW2-6 = OFF	External UARTA baud rate limited to 1Mbps.
SW2-7 = OFF	External UARTB transceiver is enabled.
SW2-8 = ON	External UARTB baud is limited to 250kbps.
I2C Connection Jumpers	
JP6 = 2-3 JP7 = 2-3	I2C1 is connected to the USB FS OTG transceiver.
User-Defined Switches SW3	
SW3-1 = ON	L2 cache configured for write-through mode. Ignored if SW3-7 is OFF.
SW3-1 = OFF	L2 cache configured for write-back mode. Ignored if SW3-7 is OFF
SW3-2 = ON	Currently unused.
SW3-3 = ON	Currently unused.
SW3-4 = ON	Normal clock configuration. 26 MHz CKIH or FPM will be used for generating the PLL output. CPU/bus/peripheral clock setting based on SW3-5 and SW3-6.
SW3-4 = OFF	Alternate (TV) clock configuration using 27 MHz CKIH for generating the PLL output. CPU/bus/peripheral clock setting based on SW3-5 and SW3-6. Requires TV daughter card.
SW3-5 = ON SW3-6 = ON	TURBO performance mode: • 528 MHz CPU clock • 528 MHz bus clock • 66 MHz peripheral clock
SW3-5 = ON SW3-6 = OFF	HIGH performance mode: • 396 MHz CPU clock • 132 MHz bus clock • 66 MHz peripheral clock This is the only recommended mode at this time.

Table 5-2. Base Board Settings (continued)

Jumper/Switch Setting	Configuration
SW3-5 = OFF SW3-6 = ON	MEDIUM performance mode: • 132 MHz CPU clock • 66 MHz bus clock • 66 MHz peripheral clock
SW3-5 = OFF SW3-6 = OFF	LOW performance mode: 132 MHz CPU clock 33 MHz bus clock 33 MHz peripheral clock
SW3-7 = ON	L2 cache enabled
SW3-7 = OFF	L2 cache disabled
SW3-8 = ON	For KITL-enabled OS images, enables active KITL mode. Use this mode to force a KITL connection to Platform Builder upon OS boot.
SW3-8 = OFF	For KITL-enabled OS images, enables passive KITL mode. Use this selection to boot the OS without establishing a KITL connection with Platform Builder. This is useful running OS images on boards without being tethered to Platform Builder.
Ethernet Jumper	
JP8 = 1-2	Enable NVRAM to Ethernet PHY.
I2C1 Connector	
JP13 = REMOVED	JP13 is the I2C1 connector. No jumper should be installed.
One-Wire Interface Jumper	
JP14 = INSTALLED	Enable One-wire.
MC13783 Power Source Selection Jumpers	
JP15 = 1-2 JP16 = 2-3 JP18 = 1-2 JP19 = REMOVED JP20 = 1-2 JP21 = 2-3 JP23 = 2-3 JP24 = 2-3 JP25 = 2-3	Supports MC13783 board installed on the base board and allows the CPU board to be powered from either the on-board regulators or MC13783 regulators.
Keypad Light Sense Jumper	
JP12 = 1-2	Select light sense.
Serial LCD CS Jumper	
JP3 = 1-2	Select LCS1.

5.1.6 MC13783 Board Settings

This section will describe the configuration for the MC13783 board.

Table 5-3. MC13783 Board Settings

Jumper/Switch Setting	Configuration
Digital Audio Direction Switches S7	
S7-6 = ON S7-5 = ON S7-4 = OFF S7-3 = OFF S7-2 = OFF S7-1 = OFF	Enable audio TX, FS and BCL buffers. The MC13783 is the master (this configuration is required by the default audio driver build settings).
Audio Clock Switches S8	
S8-6 = OFF S8-5 = OFF S8-4 = ON S8-3 = ON S8-2 = OFF S8-1 = OFF	CLIB clock is connected to CLIA and CLIB MC13783 inputs.
Power Up Mode Switches S9	
S9-6 = OFF S9-5 = ON S9-4 = OFF S9-3 = ON S9-2 = OFF S9-1 = OFF	PUM3 = GND PUM2 = GND PUM1 = OPEN Refer to MC13783 specification for a description of the power up mode selection signals.
USB OTG/WDI Mode Switches S4	
S4-6 = ON S4-5 = OFF S4-4 = OFF S4-3 = ON S4-2 = OFF S4-1 = OFF	WDI signal is pulled up. USG OTG transceiver disabled. USB mode is differential, unidirectional (6-wire).
Li Cell Emulation Switches S5	
S5-4 = OFF S5-3 = OFF S5-2 = OFF S5-1 = OFF	Onboard super cap disabled for backup. External Li cell disabled for backup. Disable SC1 charging. SC1 not discharged.
USB Signal Direction Control Switches S6	
S6-4 = OFF S6-3 = OFF S6-2 = ON S6-1 = OFF	UDATVP and USEOVM flow toward i.MX32. UTXENB does not control signal direction.
Line Input (TXIN) Source Selection Jumper	

Table 5-3. MC13783 Board Settings (continued)

JMP4 = 1-2	Select TXOUT from MC13783.
SW2 Mode Jumpers	
JMP6 = 1-2 JMP8 = 1-2	Independent operation of SW2A and SW2B.
SW1 Mode Jumpers	
JMP5 = 2-3 JMP7 = 2-3	Combined operation of SW1A and SW1B.
BATT Power Source Selection Jumper	
JMP11 = 1-2	U4 regulator is power source.
Vibrator/LED Selection Jumper	
JMP2 = 1-2	Select vibrator.

5.2 RealView Toolchain Configuration

The RealView ICE (RVI) interface is used in conjunction with the RealView Debugger (RVDEBUG) to provide a way to program EBOOT into flash memory as well as offer hardware debug capability that is not possible with Platform Builder. This section will describe the configuration required to prepare the RealView tools for connection with the target.

5.2.1 RVI Configuration

1. Install the utility software provided with the RVI unit.
2. Use **RVI Config IP** to configure the RVI unit on your network. Refer to the RVI user documentation installed with the RVI software for more information on how to use **RVI Config IP**.
3. Use **RVI Update** to install the firmware update required for proper communication with the target device. Refer to the *i.MX32 BSP Release Notes* for the suggested firmware version. Refer to the RVI user documentation installed with the RVI software for more information on how to use **RVI Update**.
4. Use the LVDS probe connector and supplied cable to connect the RVI unit to the i.MX32ADS CPU board.

5.2.2 RVDEBUG Configuration

1. Follow the steps in [Section 5.2.1, RVI Configuration](#) to configure the RVI unit.
2. Install the RealView Developer Suite.
3. Launch RVDEBUG.
4. Select **File** → **Connection** → **Connect to Target**.
5. Expand **RealView Ice**.
6. Right-Click on **RealView Ice** and select **Configure Device Info**.

7. Click on **RealView ICE: (TCP/IP. . .)** on left window pane.
8. Click **Browse. . .** button on right window pane.
9. Choose the desired RVI unit and click OK.
10. Click **Connect**.
11. Under **JTAG Clock Speed** select **Adaptive**.

NOTE

Currently, the Auto Configure Scan Chain feature does not work correctly with the i.MX32. You need to use the **Add Device** and **Device Properties** buttons to manually create the necessary scan chain entries:

- **Add Device → ARM → ETMBUF**
- **Add Device → ARM11 → ARM1136JF-S, Device Properties → ETM, VFP checked**
- **Add Device → Custom Device → UNKNOWN, IR Length = 4**
- **Add Device → Custom Device → UNKNOWN, IR Length = 5**

12. Select **File → Save**.
13. Select **File → Exit**.
14. In the **Connection Control** window, expand **RealView ICE**
15. For the i.MX32: Click the **ARM1136JF-S** device to establish a connection.

NOTE

If you are using RVI and RVDEBUG to download and debug OS images, you will need to disable vector catching by setting **File → Connection → Connection Properties → Connection=RealView ICE → Advanced_Information → Default → ARM Config → Vector Catch** to FALSE.

5.3 EBOOT Installation and Configuration

The Ethernet boot loader (EBOOT) is used to download and execute OS images. EBOOT is typically programmed into flash memory on the target hardware and executes immediately out of reset. Initially, the target hardware will not have EBOOT resident in the flash memory. In addition, the target hardware has non-volatile storage for the EBOOT network configuration (DHCP/static, MAC address, etc.) that must be initialized before using the boot loader with Platform Builder. This section will describe the procedure for building, installing, and configuring EBOOT.

NOTE

EBOOT that is resident in NOR flash will use a reserved NOR region to store the boot configuration. EBOOT that is resident in NAND flash will use a reserved NAND region to store the boot configuration. Be sure to update the boot configuration as required when switching between NOR and NAND versions of the bootloader.

5.3.1 Building an EBOOT Image for NOR Flash

Build EBOOT for NOR flash by following these steps:

1. Follow the steps in [Section 4.2, Building Run-Time Images](#) to build a retail OS image. During the process of building the OS image, libraries needed by EBOOT will be created.
2. Specify the targeted memory for the EBOOT image as NOR flash . This is achieved by ensuring that the IMG_NAND environment variable is not set within **Project → Properties → Configuration Properties → Environment**. This causes the EBOOT image to be linked for NOR flash.
3. Select the **Solution Explorer** tab of the Platform Builder window.
4. Expand **iMX32ADSMobility → F:/WINCE600 → PLATFORM → iMX32ADS → src → BOOTLOADER**.
5. Right-click on the **EBOOT** folder again and select **Rebuild**.
6. The EBOOT binary image will be created in the workspace release directory.

5.3.2 Building an EBOOT Image for NAND Flash

Build EBOOT for NAND flash by following these steps:

1. Follow the steps in [Section 4.2, Building Run-Time Images](#) to build a retail OS image. During the process of building the OS image, libraries needed by EBOOT will be created.
2. Specify the targeted memory for the EBOOT image as NAND flash . This is achieved by setting the environment variable IMG_NAND = 1 within **Project → Properties → Configuration Properties → Environment**. This causes the EBOOT image to be linked for NAND flash.
3. Select the **Solution Explorer** tab of the Platform Builder window.
4. Expand **iMX32ADSMobility → F:/WINCE600 → PLATFORM → iMX32ADS → src → BOOTLOADER**.
5. Right-click on the **EBOOT** folder again and select **Rebuild**.
6. The EBOOT binary image will be created in the workspace release directory.

5.3.3 Loading EBOOT into SDRAM using RVDEBUG

1. Configure the target and desktop workstation for serial debug messages. Refer to [Section 5.1, Serial Debug Messages](#) for more information.
2. Prepare the target hardware by following the instructions in [Section 5.1.3, i.MX32ADS Board Configuration](#). Set the board for internal ROM boot.
3. Configure the RealView toolchain by following the instructions in [Section 5.2, RealView Toolchain Configuration](#).
4. Locate the RVDEBUG initialization script in the following BSP directory:
`\WINCE600\Support\Tools\RVD\`
5. Edit the RVD script (RVD_MX31_DDR.inc which can also be used for the i.MX32ADS) and update the path for EBOOT.nb0 to point to the workspace release directory. Save the script.

6. Within RVDEBUG, select **Debug** → **Include Commands from File** and choose the previously edited and saved EBOOT initialization script.
7. The script initializes the CPU, loads EBOOT into SDRAM, and sets the program counter to the starting address. Select the **Go** button within RVDEBUG or hit **F5** to start the EBOOT execution.
8. You should see EBOOT serial debug messages on your desktop terminal emulation application.

5.3.4 Initialize EBOOT Network Configuration

To initialize EBOOT network configuration, follow these steps:

1. Follow the steps in [Section 5.3.3, Loading EBOOT into SDRAM using RVDEBUG](#) to execute EBOOT on the target.
2. Quickly switch over to your terminal emulation application and wait for the debug message “Press [ENTER] to download now or [SPACE] to cancel.” to appear.
3. Hit the space bar to bring up the EBOOT configuration menu.

NOTE

If the flash memory has not been previously programmed or has been erased due to reprogramming, EBOOT will automatically display the configuration menu without prompting the user and steps 2-3 above are skipped.

4. Specify an Ethernet MAC address by selecting the **MAC Address** menu option and then entering the 12 digit hexadecimal MAC address delimited by periods.
5. Hit enter to return to the EBOOT configuration menu. The specified MAC address should now be displayed in the EBOOT menu.
6. By default, DHCP is enabled and there is no need to specify a static IP or static subnet mask. If you need static networking parameters, use the **IP Address** and **Subnet Mask** menu options.
7. Once the desired network configuration appears in the EBOOT menu, select the **Save configuration** menu option to save the configuration to non-volatile memory.

5.3.5 Program/Update EBOOT in NOR Flash using Platform Builder

EBOOT running from SDRAM can be used to program/update the EBOOT image resident in NOR flash memory. Follow these steps to program/update the EBOOT image:

1. Build EBOOT for NOR flash following the steps in [Section 5.3.1, Building an EBOOT Image for NOR Flash](#).
2. If EBOOT is not resident in flash or the resident EBOOT is not compatible with the current BSP, you must first follow the steps in [Section 5.3.3, Loading EBOOT into SDRAM using RVDEBUG](#) to get a valid copy of EBOOT running from RAM.
3. Use the **File** menu of Platform Builder and choose **Open Workspace**.
4. Select EBOOT.bin found in the appropriate workspace release directory:

```
WINCE600\OSDesigns\iMX32ADSMobility\RelDir\Freescale_i_MX32_ADS_ARMV4I_Release\EBOOT.bin
```

5. Follow the steps in [Section 5.4, Configuring Ethernet Connection for Downloading and Debugging](#) to establish an Ethernet connection between the target and Platform Builder.
6. From the **Target** menu, select **Attach Device**.
7. After the download is complete, from the **Target** menu, select **Detach Device**.
8. Switch over your terminal emulation application. At this point, EBOOT has downloaded the image temporarily into SDRAM and is ready to begin the flash programming procedure.
9. In the terminal emulation application, hit the 'y' key to begin programming the flash.
10. EBOOT will program the image into flash and provide status using serial debug messages. Once the programming is complete, you will see the following messages:

```
INFO: Flashing sequence complete.
Reboot the device manually...
SpinForever...
```

11. Close the Platform Builder workspace for EBOOT.bin. It is not necessary to save any workspace changes.
12. If you downloaded EBOOT to SDRAM using RVDEBUG, disconnect from the RVDEBUG by selecting **File** → **Connection** → **Disconnect**.
13. Use the instructions provided in [Section 5.1.3, i.MX32ADS Board Configuration](#) to configure the CPU board for direct external boot from NOR flash.
14. Reset the target hardware. If you see new EBOOT messages appear on the terminal, EBOOT has been properly programmed into NOR flash.

5.3.6 Program/Update EBOOT in NAND Flash using Platform Builder

EBOOT running from SDRAM can be used to program/update the EBOOT image resident in NAND flash memory. Follow these steps to program/update the EBOOT image:

1. Build EBOOT for NAND flash following the steps in [Section 5.3.2, Building an EBOOT Image for NAND Flash](#).
2. If EBOOT is not resident in flash or the resident EBOOT is not compatible with the current BSP, you must first follow the steps in [Section 5.3.3, Loading EBOOT into SDRAM using RVDEBUG](#) to get a valid copy of EBOOT running from RAM.
3. Use the **File** menu of Platform Builder and choose **Open Workspace**.
4. Select EBOOT.bin found in the workspace release directory:

```
WINCE600\OSDesigns\iMX32ADSMobility\RelDir\Freescale_i_MX32_ADS_ARMV4I_Release\EBOOT.bin
```

5. Follow the steps in [Section 5.4, Configuring Ethernet Connection for Downloading and Debugging](#) to establish an Ethernet connection between the target and Platform Builder.
6. From the **Target** menu, select **Attach Device**.
7. After the download is complete, from the **Target** menu, select **Detach Device**.
8. Switch over your terminal emulation application. At this point, EBOOT has downloaded the image temporarily into SDRAM and is ready to begin the flash programming procedure.
9. In the terminal emulation application, hit the 'y' key to begin programming the flash.

10. EBOOT will program the image into flash and provide status using serial debug messages. Once the programming is complete, you will see the following messages:

```
INFO: Update of EBOOT completed successfully.
Reboot the device manually...
SpinForever...
```

11. Close the Platform Builder workspace for EBOOT.bin. It is not necessary to save any workspace changes.
12. If you downloaded EBOOT to SDRAM using RVDEBUG, disconnect from the RVDEBUG by selecting **File** → **Connection** → **Disconnect**.
13. Use the instructions provided in [Section 5.1.3, i.MX32ADS Board Configuration](#) to configure the i.MX32 CPU board for direct external boot from NAND flash.
14. Reset the target hardware. If you see new EBOOT messages appear on the terminal, EBOOT has been properly programmed into NAND flash.

5.4 Configuring Ethernet Connection for Downloading and Debugging

To configure an Ethernet connection that can be used for downloading and debugging images, follow these steps:

1. From the Platform Builder **Target** menu, choose **Connectivity Options**.
2. Choose **Kernel Service Map**.
3. In the **Target Device** box, choose a target device.

NOTE

If a connection to a target device is already configured, the settings associated with the connection to the target device appear in the **Download** box and the **Transport** box.

4. In the **Download** box, choose **Ethernet** as the download service.
5. Launch EBOOT on the target. After EBOOT initialization completes, you should begin to see BOOTME messages appear on the serial debug output. Observe the device name created by EBOOT on the serial debug output.

NOTE

If EBOOT has already been programmed into the target, a hardware reset will start EBOOT execution. If EBOOT is being programmed into the target, refer to [Section 5.3.3, Loading EBOOT into SDRAM using RVDEBUG](#) for the steps on how to launch EBOOT.

6. To the right of the **Download** box, choose **Settings**. The device name of your target should appear in the **Active Devices** box.
7. Select your target from the **Active Devices** box, and then choose **OK**.
8. In the **Transport** box, choose **Ethernet** as a kernel transport.
9. To the right of the **Transport** box, choose **Settings**.
10. Check the box next to **Use device name from bootloader** and then choose **OK**.

11. If your run-time image includes support for the kernel debugger stub, KdStub, from the **Debugger** box, choose **KdStub**. If your run-time image does not include support for a debugger, from the **Debugger** box, choose **None**.
12. Choose **Core Service Settings**.
13. To instruct Platform Builder to download a run-time image each time Platform Builder connects with the target device, under **Download Image**, choose **Always**.
14. Select **Enable KITL on device boot**.
15. Select **Clear memory on soft reset**.
16. Select **Enable access to desktop files**.
17. Choose **Apply**.
18. Choose **Close**.

Chapter 6

Downloading and Debugging Images

This section describes the procedures for downloading and debugging OS images on the i.MX32ADS board. It is assumed that you have followed the steps to build an OS image and configured your development hardware prior to attempting a download and debug session.

6.1 Downloading an Image into SDRAM using EBOOT

To download an image into the SDRAM of the target, follow these steps:

1. If EBOOT is not resident on the target, follow the procedure in [Section 5.3, EBOOT Installation and Configuration](#) to program EBOOT into the target.
2. Prepare the target hardware by following the instructions in [Section 5.1.3, i.MX32ADS Board Configuration](#). Configure the CPU board for direct external boot from NOR/NAND flash.
3. Open the desired workspace within Platform Builder.
4. **Deselect Build → Properties → Configuration Properties → Build Options → Write Run-time Image to Flash Memory.**
5. Within **Build → Properties → Configuration Properties → Environment**, remove the IMG_NAND environment variable if it exists.
6. Build the image following the steps provided in [Chapter 4, “Building OS Images”](#).
7. Reset the ADS board to launch EBOOT on the target.
8. If a target device connection has not been created within Platform Builder, follow the steps in [Section 5.4, Configuring Ethernet Connection for Downloading and Debugging](#) to establish a connection.
9. From the Platform Builder **Target** menu, select **Attach Device** to begin the download.

6.2 Downloading an OS Image into NOR Flash using EBOOT

To download an image into the NOR Flash of the target, follow these steps:

1. If EBOOT is not resident on the target, follow the procedure in [Section 5.3, EBOOT Installation and Configuration](#) to program EBOOT into the target.
2. Prepare the target hardware by following the instructions in [Section 5.1.3, i.MX32ADS Board Configuration](#). Configure the CPU board for direct external boot from NOR flash.
3. Open the desired workspace within Platform Builder.
4. **Select Build → Properties → Configuration Properties → Build Options → Write Run-time Image to Flash Memory.**
5. Within **Build → Properties → Configuration Properties → Environment**, remove the IMG_NAND environment variable if it exists.
6. Build the image following the steps provided in [Chapter 4, “Building OS Images”](#).
7. Reset the ADS board to launch EBOOT on the target.

8. If a target device connection has not been created within Platform Builder, follow the steps in [Section 5.4, Configuring Ethernet Connection for Downloading and Debugging](#) to establish a connection.
9. From the Platform Builder **Target** menu, select **Attach Device** to begin the download.
10. After the download is complete, switch over to your terminal emulation application.
11. At this point EBOOT has downloaded the image temporarily into SDRAM and is ready to begin the flash programming procedure.
12. EBOOT will program the image into flash and provide status using serial debug messages. Once the programming is complete, you will see the following messages:

```
INFO: Flashing sequence complete.
Reboot the device manually...
SpinForever...
```

6.3 Running an OS Image from NOR Flash using EBOOT

To execute an OS image from NOR flash that has been previously programmed using the procedure described in [Section 6.2, Downloading an OS Image into NOR Flash using EBOOT](#), follow these steps.

1. Reset the ADS board to launch EBOOT on the target.
2. Quickly switch over to your terminal emulation application and wait for the debug message “Press [ENTER] to download now or [SPACE] to cancel.” to appear.
3. Hit the space bar to bring up the EBOOT configuration menu.
4. Continue selecting the **Autoboot** option of the EBOOT menu until NOR is selected.
5. Specify the desired boot delay using the **Boot Delay** menu option.
6. Save the configuration using the EBOOT menu.
7. Reset the ADS board to launch EBOOT again. EBOOT will automatically jump to the OS image in NOR flash after the specified boot delay.

NOTE

If the ADS board is configured for active KITL, the image will wait for a Platform Builder connection before booting the OS image. Since the OS image is already resident in flash memory, you must reconfigure Platform Builder to jump directly to the image by selecting **Target → Connectivity Options → Core Service Settings → Download Image: Never (jump to image)**. For the i.MX32ADS, refer to [Table 5-2](#) for how to select the active/passive KITL configuration.

6.4 Downloading an OS Image into NAND Flash using EBOOT

To download an image into the NAND Flash of the target, follow these steps:

1. If EBOOT is not resident on the target, follow the procedure in [Section 5.3, EBOOT Installation and Configuration](#) to program EBOOT into NAND flash memory.
2. Prepare the target hardware by following the instructions in [Section 5.1.3, i.MX32ADS Board Configuration](#). Configure the CPU board for direct external boot from NAND flash.

3. Open the desired workspace within Platform Builder.
4. **Deselect Build → Properties → Configuration Properties → Build Options → Write Run-time Image to Flash Memory.**
5. Within **Build → Properties → Configuration Properties → Environment**, use the **New** button to add `IMG_NAND = 1`.
6. Build the image following the steps provided in [Chapter 4, “Building OS Images”](#).
7. Reset the ADS board to launch EBOOT on the target.
8. If a target device connection has not been created within Platform Builder, follow the steps in [Section 5.4, Configuring Ethernet Connection for Downloading and Debugging](#) to establish a connection.
9. From the Platform Builder **Target** menu, select **Attach Device** to begin the download.
10. After the download is complete, switch over to your terminal emulation application.
11. At this point EBOOT has downloaded the image temporarily into SDRAM and is ready to begin the flash programming procedure.
12. EBOOT will program the image into flash and provide status using serial debug messages. Once the programming is complete, you will see the following messages:

```
INFO: Update of NK completed successfully.
Reboot the device manually...
SpinForever...
```

6.5 Running an OS Image from NAND Flash using EBOOT

To execute an OS image from NAND flash that has been previously programmed using the procedure described in [Section 6.4, Downloading an OS Image into NAND Flash using EBOOT](#), follow these steps.

1. Reset the ADS board to launch EBOOT on the target.
2. Quickly switch over to your terminal emulation application and wait for the debug message “Press [ENTER] to download now or [SPACE] to cancel.” to appear.
3. Hit the space bar to bring up the EBOOT configuration menu.
4. Continue selecting the **Autoboot** option of the EBOOT menu until NAND is selected.
5. Specify the desired boot delay using the **Boot Delay** menu option.
6. Save the configuration using the EBOOT menu.
7. Reset the ADS board to launch EBOOT again. EBOOT will automatically load the OS image from NAND to RAM and execute it after the specified boot delay.

NOTE

If the ADS board is configured for active KITL, the image will wait for a Platform Builder connection before booting the OS image. Since the OS image is already resident in flash memory, you must reconfigure Platform Builder to jump directly to the image by selecting **Target → Connectivity Options → Core Service Settings → Download Image: Never (jump to image)**. For the i.MX32ADS, refer to [Table 5-1](#) for how to select the active/passive KITL configuration.

Chapter 7

BSP Features

7.1 Supported Features and Device Drivers

Details of the drivers and kernel support provided in the i.MX32 BSP are described in the *Windows CE BSP Reference Guide*. Please refer to that document for more information regarding the requirements, implementation, and testing for each of the i.MX32 BSP features.