
Windows CE BSP for MCIMX32 (i.MX32) ADS

User's Guide

WCE500-14
October 18, 2007

07-5220-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. 2007. All rights reserved.

Contents

Audience	v
Organization	v
Revision History	v
Suggested Reading	v
Conventions	vi
Definitions, Acronyms, and Abbreviations	vi
References	vii

Chapter 1 BSP Installation

1.1	Installing the BSP	1
1.2	Uninstall the BSP	1
1.3	Import BSP Catalog and Sample Workspace	2

Chapter 2 BSP Contents and Organization

2.1	Chip Support Package (CSP)	3
2.2	Production Quality OAL (PQOAL)	3
2.3	i.MX32 ADS Platform Files	4
2.4	Sample Workspace	4
2.5	Support Files	4

Chapter 3 Configuring OS Images

3.1	Image Build Type	5
3.2	BSP Environment Variables	5
3.2.1	Platform Settings Environment Variables	5
3.2.2	Catalog Environment Variables	7

Chapter 4 Building OS Images

4.1	Building Freescale CSP Libraries	8
4.2	Building Run-Time Images	8
4.2.1	Building the BSP for the First Time	8
4.2.2	Clean Build for the BSP	9
4.2.3	Incremental BSP Build	9

Chapter 5 Preparing for Downloading and Debugging

5.1	Serial Debug Messages	11
-----	-----------------------------	----

5.1.1	Desktop Workstation Serial Debug Port	11
5.1.2	Target Serial Debug Port	11
5.1.3	Target Serial KITL Port	11
5.2	i.MX32 ADS Board Configuration	11
5.2.1	i.MX32 CPU Board Settings	12
5.2.2	i.MX32 Base Board Settings	13
5.2.3	MC13783 Board Settings	16
5.3	RealView Toolchain Configuration	17
5.3.1	RVI Configuration	17
5.3.2	RVDEBUG Configuration	17
5.4	EBOOT Installation and Configuration	18
5.4.1	Building an EBOOT Image for NOR Flash	19
5.4.2	Building an EBOOT Image for NAND Flash	19
5.4.3	Building an EBOOT Image for SD/MMC Flash	20
5.4.4	Loading EBOOT into SDRAM using RVDEBUG	20
5.4.5	Initialize EBOOT Network Configuration	21
5.4.6	Program/Update EBOOT in NOR Flash using Platform Builder	21
5.4.7	Program/Update EBOOT in NAND Flash using Platform Builder	22
5.4.8	Program/Update EBOOT in SD/MMC Flash using Platform Builder	23
5.5	SBOOT Installation and Configuration	24
5.5.1	Target Serial Bootloader Port	24
5.5.2	Building a SBOOT Image for NOR Flash	25
5.5.3	Building a SBOOT Image for NAND Flash	25
5.5.4	Building a SBOOT Image for SD/MMC Flash	25
5.5.5	Loading SBOOT into SDRAM using RVDEBUG	26
5.5.6	Initialize SBOOT Network Configuration	26
5.5.7	Program/Update SBOOT in NOR Flash using Platform Builder	27
5.5.8	Program/Update SBOOT in NAND Flash using Platform Builder	28
5.5.9	Program/Update SBOOT in SD/MMC Flash using Platform Builder	29
5.6	XLDR Installation and Configuration	30
5.6.1	Building a NAND XLDR Image	30
5.6.2	Program XLDR into NAND Flash using Platform Builder	30
5.6.3	Building a SD/MMC XLDR Image	31
5.6.4	Program XLDR into SD/MMC Flash using Platform Builder	32
5.7	Configuring Ethernet Connection for Downloading and Debugging	33
5.8	Configuring Serial Connection for Downloading and Debugging	34

Chapter 6

Downloading and Debugging Images

6.1	Downloading an Image into SDRAM using EBOOT	36
6.2	Downloading an OS Image into NOR Flash using EBOOT	36
6.3	Running an OS Image from NOR Flash using EBOOT	37
6.4	Downloading an OS Image into NAND Flash using EBOOT	37
6.5	Running an OS Image from NAND Flash using EBOOT	38

6.6	Downloading an OS Image into SD/MMC Flash using EBOOT	38
6.7	Running an OS Image from SD/MMC Flash using EBOOT	39
6.8	Downloading an Image into SDRAM using SBOOT	40
6.9	Downloading an OS Image into NOR Flash using SBOOT	40
6.10	Running an OS Image from NOR Flash using SBOOT	41
6.11	Downloading an OS Image into NAND Flash using SBOOT	41
6.12	Running an OS Image from NAND Flash using SBOOT	42
6.13	Downloading an OS Image into SD/MMC Flash using SBOOT	42
6.14	Running an OS Image from SD/MMC Flash using SBOOT	43

Chapter 7

i.MX32 BSP Features



About This Book

This document is a user's guide for the Freescale MCIMX32 (i.MX32) Windows CE board support package (BSP). This document will describe how to install the BSP and how to use Microsoft Platform Builder for Windows CE 5.0 to build, download, and debug OS images. Information on the configuration and usage of features included within the Freescale i.MX32 BSP will also be provided.

Audience

This guide is intended for users of Microsoft Platform Builder who want to build and execute Windows CE 5.0 OS images based on the Freescale i.MX32 BSP. The audience also includes Windows CE application developers that want to leverage API interfaces provided by the Freescale i.MX32 BSP.

Organization

This document is organized into the following chapters.

Chapter 1	Describes how to install/uninstall the i.MX32 BSP.
Chapter 2	Describes the contents and organization of the i.MX32 BSP.
Chapter 3	Describes how to configure the i.MX32 BSP for building Windows CE OS images.
Chapter 4	Provides instructions on how to build Windows CE OS images using the i.MX32 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 i.MX32 BSP features included in this release.

Revision History

This is the first release of this document in this format.

Suggested Reading

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

- <http://msdn.microsoft.com/embedded/windowsce>
- *HW Getting Started - i.MX31 ADS*
- *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
SBOOT	Serial bootloader
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.

- *Platform Builder for Microsoft Windows CE 5.0 Help*
- *HW Getting Started - i.MX32 ADS*
- *Windows CE BSP Reference Guide*



Chapter 1

BSP Installation

The BSP is distributed as a single Microsoft Windows Installer (.msi) file. Please refer to the *i.MX32 BSP Release Notes* for additional instructions and information before installing and using the BSP.

1.1 Installing the BSP

The following steps will install the MCIMX32 (i.MX32) BSP into the Windows CE source code tree and the Platform Builder development environment.

1. Install Platform Builder from the installation discs. Please refer to the *Release Notes* on the Platform Builder installation discs for installation instructions.

NOTE

During Platform Builder installation, be sure to select the **Custom (Tools and OS)** option on **Setup Type** window. Use the **Custom Setup** window to select which versions of the operating system to include. The **ARMV4I** version must be installed on the local hard drive for the i.MX32 BSP. All other versions are not required.

2. Close Platform Builder.
3. If you have previously installed the i.MX32 BSP, remove any existing BSP files by following the instructions in [Section 1.2, Uninstall the BSP](#).
4. Download and execute the following Microsoft Windows installer file:

```
WCE500-13-BSP.msi
```

NOTE

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

```
WINCE500\PLATFORM\COMMON\SRC\ARM\dirs  
WINCE500\PUBLIC\COMMON\OAK\CSP\ARM\dirs
```

Rename or move these files before executing the BSP installer file.

5. Follow the steps provided by the wizard to complete the BSP installation.

1.2 Uninstall the BSP

This section describes how to remove an installation of the i.MX32 BSP from the Windows CE source code tree and Platform Builder development environment.

NOTE

Uninstalling the BSP will remove all files that were populated by the Microsoft Windows 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. Right click on the Microsoft Windows Installer (.msi) file and select **Uninstall**.
3. Manually remove the remaining BSP files and directories that are not uninstalled by the previous step:

```
\WINCE500\PBWorkspaces\MX32ADSMobility
\WINCE500\PLATFORM\COMMON\SRC\ARM\dirs
\WINCE500\PLATFORM\COMMON\SRC\ARM\FREESCALE
\WINCE500\PLATFORM\MX32ADS
\WINCE500\PUBLIC\COMMON\OAK\CATALOG\CEC\mx32ads.cec
\WINCE500\PUBLIC\COMMON\OAK\CSP\ARM\dirs
\WINCE500\PUBLIC\COMMON\OAK\CSP\ARM\FREESCALE
\WINCE500\SUPPORT
```

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

```
\WINCE500\PUBLIC\COMMON\OAK\LIB\ARMV4I
\WINCE500\PLATFORM\COMMON\lib\ARMV4I
```

1.3 Import BSP Catalog and Sample Workspace

After installing the i.MX32 BSP, you can use the sample workspace provided in the BSP package to build a Windows CE OS image based on the i.MX32 BSP.

1. In Platform Builder, select **File** → **Manage Catalog Items**.
2. Within the **Manage Catalog Items** dialog, select **Import** and choose the following catalog file:

```
WINCE500\PUBLIC\COMMON\OAK\CATALOG\CEC\mx32ads.cec
```

3. Select the **Open** button and then select the **OK** button to complete the catalog import.
4. In Platform Builder, select **File** → **Open Workspace**.
5. Select the i.MX32 BSP sample workspace located as follows:

```
WINCE500\PBWorkspaces\MX32ADSMobility\MX32ADSMobility.pbxml
```

6. To view the workspace loaded, select **View** → **Workspace**.

Chapter 2

BSP Contents and Organization

The Freescale i.MX32 BSP is a collection of code and support files that can be integrated into the Microsoft Platform Builder development environment to create Windows CE OS images for 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 CE source tree. The BSP directory structure follows the production-quality OAL (PQOAL) and production-quality drivers (PQD) structure recommended by Microsoft. For more information on this directory structure layout, refer to the topic “*BSP and CSP Directory Layout*” in the *Platform Builder for Microsoft Windows CE 5.0 Help*.

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 Chip Support Package (CSP)

The Freescale CSP 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 CSP can be reused in a new platform design for i.MX32 without modification. To keep the CSP platform agnostic, drivers in the CSP directory utilize hardware abstraction routines that can be ported to a specific platform or board. The CSP 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 CSP source code for the i.MX32 CSP:

```
WINCE500\PUBLIC\COMMON\OAK\CSP\ARM\FREESCALE\MXARM11
WINCE500\PUBLIC\COMMON\OAK\CSP\ARM\FREESCALE\MX32
```

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

2.2 Production Quality OAL (PQOAL)

A new feature of Windows CE 5.0 is the availability of the 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 *Platform Builder for Microsoft Windows CE 5.0 Help*.

Where possible, the i.MX32 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:

```
WINCE500\PLATFORM\COMMON\SRC\ARM\FREESCALE\MXARM11
WINCE500\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.MX32 ADS 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:

```
WINCE500\PLATFORM\MX32ADS
```

The i.MX32 platform directory implements the hardware abstraction routines invoked by driver code in the Freescale CSP 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 Workspace

Workspaces are used by Platform Builder to encapsulate the OS components and build options necessary for the Windows CE tools to generate an OS image. For more information about the contents and creation of a Platform Builder workspace, refer to the topic “*Workspaces*” in the *Platform Builder for Microsoft Windows CE 5.0 Help*.

A default workspace has been included in the BSP within the following directory:

```
WINCE500\PBWorkspaces\MX32ADSMobility
```

This workspace was created using the **New Platform Wizard** feature within Platform Builder and utilizing the Mobile Handheld Design Template. For more information about creating Workspaces, refer to the topic “*Creating an OS Design with the New Platform Wizard*” in the *Platform Builder for Microsoft Windows CE 5.0 Help*.

2.5 Support Files

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

```
WINCE500\SUPPORT
```

This directory will be used to store applications and tests targeted for the i.MX32 BSP. 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 i.MX32 BSP sample workspace. These configurations control the type of OS image (debug, release, ship) 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 workspace 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 OS** menu and choosing **Set Active Configuration**. The sample workspace provided with the i.MX32 BSP provides the following two image build types:

- **Freescall MX32ADS: 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 MX32ADS: 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.

For more information about the build types available for Windows CE, refer to the topic “*Build Configurations*” in the *Platform Builder for Microsoft Windows CE 5.0 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 **Platform Settings** 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 Platform Settings Environment Variables

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

1. Open the sample workspace for the i.MX32 BSP.
2. From the **Platform** menu, choose **Settings**.
3. Select the **Environment** tab.

Table 3-1 provides a summary of BSP environment variables supported with the **Platform 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.
_TGTBOARD	Allows conditional compilation for the i.MX32 ADS hardware or the Virtio VPMX32 simulator.	_TGTBOARD = ADS Configures the build for i.MX32 ADS hardware. _TGTBOARD = VPMX32 Configures the build for the VPMX32 simulator. Note: A clean build of all BSP components is required after modifying this setting.
BUILD_OPTIONS	Specifies an optional set of BSP directories for the build tools	BUILD_OPTIONS = FREESCALE MXARM11 MX32 PMIC Required for proper build of BSP. DO NOT modify or remove from sample workspace.

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

Variable Name	Description	Settings
IMGNAND	Used by EBOOT/SBOOT and OS build files to determine if images are targeted for NAND flash.	<imgnand 1<br="" ==""></imgnand> Link EBOOT/SBOOT 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.
IMGMMC	Used by EBOOT/SBOOT and OS build files to determine if images are targeted for SD/MMC flash. This setting also maps images to CS4 memory region. With this mapping, we can flash all .bin and .nb0 files to the SD/MMC card.	<imgmmc 1<br="" ==""></imgmmc> Link EBOOT/SBOOT and OS images for SD/MMC 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.
IMGMMC_NORMAL	Used by EBOOT/SBOOT and OS build files to determine if images are targeted for SD/MMC flash. This setting indicates images are remapped to base address of SDHC1 peripheral. With this setting, .bin files only cannot be flashed into the SD/MMC card due to illegal memory access by the BLCOMMON code.	<imgmmc 1<br="" ==""></imgmmc> <imgmmc_normal 1<br="" ==""></imgmmc_normal> Link EBOOT/SBOOT and OS images for SD/MMC flash. Note: This environment variable takes effect only if <imgmmc=1 <="" is="" set="" td=""></imgmmc=1>
BSP_LC_MMC	Low Capacity MMC support. Used by XLDR targeted for standard capacity SD/MMC flash (<=2GB).	BSP_LC_MMC = 1 Builds XLDR to support MMC boot from a standard capacity SD/MMC flash. (<=2GB)
IMGSDBUS2	High Capacity SD support for the SDHC OS driver. (> 2GB).	<imgsdbus2 1<br="" ==""></imgsdbus2> Provides support for High capacity SD cards. (> 2GB)
BSP_OAL_TIMER32K	Used to include PQOAL support for OAL tick timer based on 32.768 kHz CKIL input clock. This specialized version of the OAL timer allows the system to enter low-power modes that require a transition of the peripheral clock rate.	BSP_OAL_TIMER32K = 1 Includes support for OAL tick timer with CKIL as input clock source. Note: A clean build of PLATFORM BSP components is required after modifying this setting.

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 CE OS images using the i.MX32 BSP.

4.1 Building Freescale CSP Libraries

The Freescale CSP libraries that support i.MX32 are not generated during the **Build OS** → **Sysgen** build procedure. Windows CE ships with pre-built CSP libraries for various ARM processors, but the libraries for i.MX32 must be built separately since they are not included with the standard Microsoft distribution.

Follow these steps to build the CSP libraries:

1. Open the sample workspace.
2. Select the desired build type discussed in [Section 3.1, Image Build Type](#).
3. Select the **File View** tab of the Platform Builder **Workspace** window.
4. Expand the **Favorites** group to view the BSP sub items.
5. Right-click on the **FREESCALE CSP** folder and enable the **Clean Before Building** option.
6. Right-click on the **FREESCALE CSP** folder again and select **Build Current Project**.

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

```
WINCE500\PUBLIC\COMMON\OAK\LIB\ARMV4I\RETAIL
```

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

```
WINCE500\PUBLIC\COMMON\OAK\LIB\ARMV4I\DEBUG
```

4.2 Building Run-Time Images

After building the Freescale CSP libraries, 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 *Platform Builder for Microsoft Windows CE 5.0 Help*.

4.2.1 Building the BSP for the First Time

1. Open the sample workspace.
2. Select the desired build type discussed in [Section 3.1, Image Build Type](#).
3. Make sure the Freescale CSP libraries are available by following the instructions in [Section 4.1, Building Freescale CSP Libraries](#).
4. Using the **Build OS** menu, configure the build options as follows:
 - **Clean Before Building** unselected
 - **Copy Files to Release Directory After Build** selected
 - **Make Run-Time Image After Build** selected
5. Select **Build OS** → **Sysgen** to start the build process.

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

```
WINCE500\PBWorkspaces\MX32ADSMobility\RelDir\MX32ADS_ARMV4I_Release
```

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

```
WINCE500\PBWorkspaces\MX32ADSMobility\RelDir\MX32ADS_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 all the i.MX32ADS BSP components can be accomplished as follows:

1. Follow the instructions in [Section 4.1, Building Freescale CSP Libraries](#) to perform a clean build of the Freescale CSP libraries.
2. Using the Build OS menu, configure the build options as follows:
 - **Clean Before Building** selected
 - **Copy Files to Release Directory After Build** selected
 - **Make Run-Time Image After Build** unselected
3. Select **Build OS → Sysgen** to build the OS design and perform a clean build of the PQOAL libraries.
4. Using the Build OS menu, configure build options as follows:
 - **Clean Before Building** selected
 - **Copy Files to Release Directory After Build** selected
 - **Make Run-Time Image After Build** selected
2. Select **Build OS → Build and Sysgen Current BSP** to perform a clean build of the BSP platform directory and complete the creation of an OS image.

4.2.3 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 BSP components to quickly build an updated OS image as follows:

1. Open a workspace.
2. Select the desired build type discussed in [Section 3.1, Image Build Type](#).
3. Select the **File View** tab of the Platform Builder **Workspace** window.
4. Expand the **Favorites** group to view the BSP sub items.
5. Right-click on any of the **Favorites** group folders and disable the **Clean Before Building** option.
6. If any of the Freescale CSP source files have been modified, perform an incremental build of the CSP libraries by right-clicking on the **FREESCALE CSP** folder selecting **Build Current Project**.
7. If any of the PQOAL source files have been modified, perform an incremental build of the PQOAL libraries by right-clicking on the **FREESCALE PQOAL** folder and selecting **Build Current Project**.
8. Using the Build OS menu, configure the build options as follows:

- **Clean Before Building** unselected
 - **Copy Files to Release Directory After Build** selected
 - **Make Run-Time Image After Build** selected
9. Select **Build OS → Build and Sysgen Current BSP** to perform an incremental build of the BSP platform directory 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 i.MX32 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.MX32 BSP has been configured to use internal UART1 routed to the UARTC ADS board connector for serial debug messages. UARTC on the ADS is the upper DB-9 connector next to the Ethernet port. Connect a standard serial cable between UARTC of the MX32 ADS and your development workstation.

5.1.3 Target Serial KITL Port

The i.MX32 BSP has been configured to use internal UART3 routed to the UARTB ADS board connector for serial KITL messages. UARTB on the ADS is the lower male DB-9 connector next to the MC13783 base board connector. Connect a NULL modem serial cable between UARTB of the MX32 ADS and your development workstation.

5.2 i.MX32 ADS Board Configuration

This section will describe the switch and jumper configuration required for proper operation of the BSP on the i.MX32 ADS 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.MX32 ADS documentation for more information on the full set of jumper and switch configurations available.

5.2.1 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/SBOOT.

Table 5-1. CPU Board Settings (continued)

Jumper/Switch Setting	Configuration
SW2-5 = OFF SW2-4 = ON SW2-3 = OFF SW2-2 = ON SW2-1 = ON	Direct external boot from NOR. Used to execute EBOOT/SBOOT or OS images previously programmed into NOR flash memory.
SW2-5 = OFF SW2-4 = ON SW2-3 = ON SW2-2 = ON SW2-1 = OFF	Direct external boot from NAND. Used to execute EBOOT/SBOOT or OS images previously programmed into NAND flash memory.
SW2-5 = ON SW2-4 = ON SW2-3 = OFF SW2-2 = OFF SW2-1 = OFF	Direct external boot from SD/MMC. Used to execute EBOOT/SBOOT or OS images previously programmed into SD/MMC 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.

5.2.2 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	
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 = OFF	Alternate (TV) clock configuration is based on 27 MHz CKIH for generating the PLL output. CPU/bus/peripheral clock setting based on SW3-5 and SW3-6. Requires TVOUT daughter board to supply external 27MHz CKIH.
SW3-4 = ON	Clock configuration is based on 26 MHz CKIH for generating the PLL output. CPU/bus/peripheral clock setting based on SW3-5 and SW3-6.
SW3-5 = ON SW3-6 = ON	TURBO performance mode: • 528 MHz CPU clock • 132 MHz bus clock • 66 MHz peripheral clock

Table 5-2. Base Board Settings (continued)

Jumper/Switch Setting	Configuration
SW3-5 = ON SW3-6 = OFF	HIGH performance mode: 396MHz CPU clock 132 MHz bus clock 66 MHz peripheral clock Note: You must configure the system for this mode when including the DVFC driver. If the DVFC driver detects the system is not in this configuration upon initialization, it will fail to load.
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.

Table 5-2. Base Board Settings (continued)

Jumper/Switch Setting	Configuration
Serial LCD CS Jumper	
JP3 = 1-2	Select LCS1.

5.2.3 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. MC13783 is the master.
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.

Table 5-3. MC13783 Board Settings (continued)

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	
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.

5.3 RealView Toolchain Configuration

The RealView ICE (RVI) interface is used in conjunction with the RealView Debugger (RVDEBUG) to provide a way to program EBOOT/SBOOT 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.

5.3.1 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.MX32 ADS CPU board.

5.3.2 RVDEBUG Configuration

1. Follow the steps in [Section 5.3.1, 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 the **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 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. 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.

5.4 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 on the target hardware.

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

EBOOT for MX32 supports both low and high capacity MMC cards. But it has support only for SD low capacity cards since BOOTROM code does not support SD High capacity boot.

5.4.1 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 and IMG MMC environment variables are not set within **Platform** → **Settings** → **Environment**. This causes the EBOOT image to be linked for NOR flash.
3. Select the **File View** tab of the Platform Builder **Workspace** window.
4. Expand **Favorites** → **MX32ADS PLATFORM** → **src** → **BOOTLOADER**.
5. Right-click on the **BOOTLOADER** folder and enable the **Clean Before Building** option.
6. Right-click on the **COMMON** folder and select **Build Current Project**.
7. Right-click on the **EBOOT** folder and select **Build Current Project**.
8. The EBOOT binary image will be created in the workspace release directory.

5.4.2 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 **Platform** → **Settings** → **Environment**. Also ensure that the IMG MMC environment variable is not set within **Platform** → **Settings** → **Environment**. This causes the EBOOT image to be linked for NAND flash.
3. Select the **File View** tab of the Platform Builder **Workspace** window.
4. Expand **Favorites** → **MX32ADS PLATFORM** → **src** → **BOOTLOADER**.
5. Right-click on the **BOOTLOADER** folder and enable the **Clean Before Building** option.
6. Right-click on the **COMMON** folder and select **Build Current Project**.
7. Right-click on the **EBOOT** folder and select **Build Current Project**.

8. The EBOOT binary image will be created in the workspace release directory.

5.4.3 Building an EBOOT Image for SD/MMC Flash

Build EBOOT for SD/MMC 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 SD/MMC flash. This is achieved by setting the environment variable `IMGMMC = 1` within **Platform** → **Settings** → **Environment**. Also ensure that the `IMGNAND` environment variable is not set within **Platform** → **Settings** → **Environment**. This causes the EBOOT image to be linked for SD/MMC flash.
3. Select the **File View** tab of the Platform Builder **Workspace** window.
4. Expand **Favorites** → **MX32ADS PLATFORM** → **src** → **BOOTLOADER**.
5. Right-click on the **BOOTLOADER** folder and enable the **Clean Before Building** option.
6. Right-click on the **COMMON** folder and select **Build Current Project**.
7. Right-click on the **EBOOT** folder and select **Build Current Project**.
8. The EBOOT binary image will be created in the workspace release directory.

5.4.4 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 5.2, i.MX32 ADS Board Configuration](#). Set the i.MX32 CPU board for internal ROM boot.
3. Configure the RealView toolchain by following the instructions in [Section 5.3, RealView Toolchain Configuration](#).
4. Locate the RVDEBUG initialization script in the following BSP directory:
`\WINCE500\Support\Tools\RVD\`
5. Edit the RVD script (`RVD_MX31_DDR.inc`) 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.

5.4.5 Initialize EBOOT Network Configuration

To initialize EBOOT network configuration, follow these steps:

1. Follow the steps in [Section 5.4.4, 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.

5.4.6 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 5.4.1, 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 5.4.4, 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:
`WINCE500\PBWorkspaces\MX32ADSMobility\RelDir\MX32ADS_ARMV4I_Release\EBOOT.bin`
5. Follow the steps in [Section 5.7, 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 5.2, i.MX32 ADS Board Configuration](#) to configure the i.MX32 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.

5.4.7 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 5.4.2, Building an EBOOT Image for NAND Flash](#).
2. If EBOOT and XLDR are not resident in flash or the resident EBOOT is not compatible with the current BSP, you must first follow the steps in [Section 5.4.4, 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:
`WINCE500\PBWorkspaces\MX32ADSMobility\RelDir\MX32ADS_ARMV4I_Release\EBOOT.bin`
5. Follow the steps in [Section 5.7, 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/SBOOT 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 5.2, i.MX32 ADS Board Configuration](#) to configure the i.MX32 CPU board for direct external boot from NAND flash.

NOTE

Booting from NAND requires the XLDR and EBOOT images to be resident in the NAND device. Be sure to follow the steps in [Section 5.4.7, Program/Update EBOOT in NAND Flash using Platform Builder](#) and [Section 5.6, XLDR Installation and Configuration](#) before attempting to boot from NAND.

14. Reset the target hardware. If you see new EBOOT messages appear on the terminal, EBOOT has been properly programmed into NAND flash.

5.4.8 Program/Update EBOOT in SD/MMC Flash using Platform Builder

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

1. Build EBOOT for SD/MMC flash following the steps in [Section 5.4.3, Building an EBOOT Image for SD/MMC Flash](#).
2. If EBOOT and XLDR are not resident in flash or the resident EBOOT is not compatible with the current BSP, you must first follow the steps in [Section 5.4.4, 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:
`WINCE500\PBWorkspaces\MX32ADSMobility\RelDir\MX32ADS_ARMV4I_Release\EBOOT.bin`
5. Follow the steps in [Section 5.7, 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/SBOOT 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 5.2, i.MX32 ADS Board Configuration](#) to configure the i.MX32 CPU board for direct external boot from SD/MMC flash.

NOTE

Booting from SD/MMC requires the XLDR and EBOOT images to be resident in the SD/MMC device. Be sure to follow the steps in [Section 5.4.8, Program/Update EBOOT in SD/MMC Flash using Platform Builder](#) and [Section 5.6, XLDR Installation and Configuration](#) before attempting to boot from SD/MMC.

14. Reset the target hardware. If you see new EBOOT messages appear on the terminal, EBOOT has been properly programmed into SD/MMC flash.

5.5 SBOOT Installation and Configuration

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

NOTE

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

SBOOT occupies the same region as EBOOT resident in NOR/NAND/MMC flash. So only either SBOOT or EBOOT can reside in that flash region. So flashing SBOOT would erase EBOOT completely and similarly vice-versa.

SBOOT for MX32 supports both low and high capacity MMC cards. But it has support only for SD low capacity cards since BOOTROM code does not support SD High capacity boot.

5.5.1 Target Serial Bootloader Port

The i.MX32 BSP has been configured to use internal UART3 routed to the UARTB ADS board connector for Serial BOOT messages by default. UARTB on the ADS is the lower male DB-9 connector next to the MC13783 base board connector. Connect a NULL modem serial cable between UARTB of the MX32 ADS and your development workstation. Follow the steps in [Section 5.5.6, Initialize SBOOT Serial Configuration](#) to configure SBOOT port to any other valid serial port.

5.5.2 Building a SBOOT Image for NOR Flash

Build SBOOT 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 SBOOT will be created.
2. Specify the targeted memory for the SBOOT image as NOR flash . This is achieved by ensuring that the IMGNAND and IMGMMC environment variables are not set within **Platform** → **Settings** → **Environment**. This causes the SBOOT image to be linked for NOR flash.
3. Select the **File View** tab of the Platform Builder **Workspace** window.
4. Expand **Favorites** → **MX32ADS PLATFORM** → **src** → **BOOTLOADER**.
5. Right-click on the **BOOTLOADER** folder and enable the **Clean Before Building** option.
6. Right-click on the **COMMON** folder and select **Build Current Project**.
7. Right-click on the **SBOOT** folder and select **Build Current Project**.
8. The SBOOT binary image will be created in the workspace release directory.

5.5.3 Building a SBOOT Image for NAND Flash

Build SBOOT 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 SBOOT will be created.
2. Specify the targeted memory for the SBOOT image as NAND flash . This is achieved by setting the environment variable IMGNAND = 1 within **Platform** → **Settings** → **Environment**. Also ensure that the IMGMMC environment variable is not set within **Platform** → **Settings** → **Environment**. This causes the SBOOT image to be linked for NAND flash.
3. Select the **File View** tab of the Platform Builder **Workspace** window.
4. Expand **Favorites** → **MX32ADS PLATFORM** → **src** → **BOOTLOADER**.
5. Right-click on the **BOOTLOADER** folder and enable the **Clean Before Building** option.
6. Right-click on the **COMMON** folder and select **Build Current Project**.
7. Right-click on the **SBOOT** folder and select **Build Current Project**.
8. The SBOOT binary image will be created in the workspace release directory.

5.5.4 Building a SBOOT Image for SD/MMC Flash

Build SBOOT for SD/MMC 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 SBOOT will be created.
2. Specify the targeted memory for the SBOOT image as SD/MMC flash . This is achieved by setting the environment variable IMGMMC = 1 within **Platform** → **Settings** → **Environment**. Also ensure that the IMGNAND environment variable is not set within **Platform** → **Settings** → **Environment**. This causes the SBOOT image to be linked for SD/MMC flash.
3. Select the **File View** tab of the Platform Builder **Workspace** window.

4. Expand **Favorites** → **MX32ADS PLATFORM** → **src** → **BOOTLOADER**.
5. Right-click on the **BOOTLOADER** folder and enable the **Clean Before Building** option.
6. Right-click on the **COMMON** folder and select **Build Current Project**.
7. Right-click on the **SBOOT** folder and select **Build Current Project**.
8. The SBOOT binary image will be created in the workspace release directory.

5.5.5 Loading SBOOT 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 5.2, i.MX32 ADS Board Configuration](#). Set the i.MX32 CPU board for internal ROM boot.
3. Configure the RealView toolchain by following the instructions in [Section 5.3, RealView Toolchain Configuration](#).
4. Locate the RVDEBUG initialization script in the following BSP directory:
`\WINCE500\Support\Tools\RVD\`
5. Edit the RVD script (RVD_MX31_DDR) and update the path for SBOOT.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 SBOOT initialization script.
7. The script initializes the CPU, loads SBOOT into SDRAM, and sets the program counter to the starting address. Select the **Go** button within RVDEBUG or hit **F5** to start the SBOOT execution.
8. You should see SBOOT serial debug messages on your desktop terminal emulation application.

5.5.6 Initialize SBOOT Serial Configuration

To initialize SBOOT serial configuration, follow these steps:

1. Follow the steps in [Section 5.5.5, Loading SBOOT into SDRAM using RVDEBUG](#) to execute SBOOT 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 SBOOT configuration menu.

NOTE

If the flash memory has not been previously programmed or has been erased due to reprogramming, SBOOT 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 SBOOT configuration menu. The specified MAC address should now be displayed in the SBOOT menu.

6. Specify the necessary Serial configurations like UART channel, Baud Rate, Parity, Data Bits, Stop Bit and Flow Control. The serial configurations configured should match with the Platform Builder serial settings.
7. Once the desired serial configuration appears in the SBOOT menu, select the **Save configuration** menu option to save the configuration to non-volatile memory.

5.5.7 Program/Update SBOOT in NOR Flash using Platform Builder

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

1. Build SBOOT for NOR flash following the steps in [Section 5.5.2, Building a SBOOT Image for NOR Flash](#).
2. If SBOOT is not resident in flash or the resident SBOOT is not compatible with the current BSP, you must first follow the steps in [Section 5.5.5, Loading SBOOT into SDRAM using RVDEBUG](#) to get a valid copy of SBOOT running from RAM.
3. Use the **File** menu of Platform Builder and choose **Open Workspace**.
4. Select SBOOT.bin found in the workspace release directory:
`WINCE500\PBWorkspaces\MX32ADSMobility\RelDir\MX32ADS_ARMV4I_Release\SBOOT.bin`
5. Follow the steps in [Section 5.8, Configuring Serial Connection for Downloading and Debugging](#) to establish an Serial 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, SBOOT 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. SBOOT 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 SBOOT.bin. It is not necessary to save any workspace changes.
12. If you downloaded SBOOT to SDRAM using RVDEBUG, disconnect from the RVDEBUG by selecting **File** → **Connection** → **Disconnect**.
13. Use the instructions provided in [Section 5.2, i.MX32 ADS Board Configuration](#) to configure the i.MX32 CPU board for direct external boot from NOR flash.
14. Reset the target hardware. If you see new SBOOT messages appear on the terminal, SBOOT has been properly programmed into NOR flash.

5.5.8 Program/Update SBOOT in NAND Flash using Platform Builder

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

1. Build SBOOT for NAND flash following the steps in [Section 5.5.3, Building a SBOOT Image for NAND Flash](#).
2. If SBOOT and XLDR are not resident in flash or the resident SBOOT is not compatible with the current BSP, you must first follow the steps in [Section 5.5.5, Loading SBOOT into SDRAM using RVDEBUG](#) to get a valid copy of SBOOT running from RAM.
3. Use the **File** menu of Platform Builder and choose **Open Workspace**.
4. Select SBOOT.bin found in the workspace release directory:
`WINCE500\PBWorkspaces\MX32ADSMobility\RelDir\MX32ADS_ARMV4I_Release\SBOOT.bin`
5. Follow the steps in [Section 5.8, Configuring Serial Connection for Downloading and Debugging](#) to establish an Serial 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, SBOOT 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. SBOOT 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/SBOOT completed successfully.
Reboot the device manually...
SpinForever...
```
11. Close the Platform Builder workspace for SBOOT.bin. It is not necessary to save any workspace changes.
12. If you downloaded SBOOT to SDRAM using RVDEBUG, disconnect from the RVDEBUG by selecting **File** → **Connection** → **Disconnect**.
13. Use the instructions provided in [Section 5.2, i.MX32 ADS Board Configuration](#) to configure the i.MX32 CPU board for direct external boot from NAND flash.

NOTE

Booting from NAND requires the XLDR and SBOOT images to be resident in the NAND device. Be sure to follow the steps in [Section 5.5.8, Program/Update SBOOT in NAND Flash using Platform Builder](#) and [Section 5.6, XLDR Installation and Configuration](#) before attempting to boot from NAND.

14. Reset the target hardware. If you see new SBOOT messages appear on the terminal, SBOOT has been properly programmed into NAND flash.

5.5.9 Program/Update SBOOT in SD/MMC Flash using Platform Builder

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

1. Build SBOOT for SD/MMC flash following the steps in [Section 5.5.4, Building a SBOOT Image for SD/MMC Flash](#).
2. If SBOOT and XLDR are not resident in flash or the resident SBOOT is not compatible with the current BSP, you must first follow the steps in [Section 5.5.5, Loading SBOOT into SDRAM using RVDEBUG](#) to get a valid copy of SBOOT running from RAM.
3. Use the **File** menu of Platform Builder and choose **Open Workspace**.
4. Select SBOOT.bin found in the workspace release directory:
`WINCE500\PBWorkspaces\MX32ADSMobility\RelDir\MX32ADS_ARMV4I_Release\SBOOT.bin`
5. Follow the steps in [Section 5.8, Configuring Serial Connection for Downloading and Debugging](#) to establish an Serial 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, SBOOT 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. SBOOT 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/SBOOT completed successfully.
Reboot the device manually...
SpinForever...
```
11. Close the Platform Builder workspace for SBOOT.bin. It is not necessary to save any workspace changes.
12. If you downloaded SBOOT to SDRAM using RVDEBUG, disconnect from the RVDEBUG by selecting **File** → **Connection** → **Disconnect**.
13. Use the instructions provided in [Section 5.2, i.MX32 ADS Board Configuration](#) to configure the i.MX32 CPU board for direct external boot from SD/MMC flash.

NOTE

Booting from SD/MMC requires the XLDR and EBOOT images to be resident in the SD/MMC device. Be sure to follow the steps in [Section 5.4.8, Program/Update EBOOT in SD/MMC Flash using Platform Builder](#) and [Section 5.6, XLDR Installation and Configuration](#) before attempting to boot from SD/MMC.

14. Reset the target hardware. If you see new SBOOT messages appear on the terminal, SBOOT has been properly programmed into SD/MMC flash.

5.6 XLDR Installation and Configuration

XLDR is a boot loader that is resident in the first 4K bytes of the external memory flash device. An external flash memory could be NAND, SD/MMC, etc., which does not support XIP. XLDR is the first code to execute on the system when booting directly from external flash memory. Based on the boot switch settings (either NAND boot or SD/MMC boot), the BOOT ROM code is responsible for copying XLDR (first 4Kbytes) from the external memory flash to the internal processor RAM after reset. After the XLDR has been copied, the CPU program counter is set to the starting base of the external memory flash and begins executing the XLDR. The responsibility of the XLDR is to then load a secondary bootloader (EBOOT, Windows Mobile IPL, etc.) that can provide additional capabilities for booting OS images. This section will describe the procedure for building and installing XLDR.

NOTE

XLDR for MX32 supports both low and high capacity MMC cards. But it has support only for SD low capacity cards since BOOTROM code does not support SD High capacity boot.

5.6.1 Building a NAND XLDR Image

The steps to build a XLDR image NAND for flash are as follows:

1. Select the **File View** tab of the Platform Builder **Workspace** window.
2. Expand **Favorites** → **MX32ADS PLATFORM** → **src** → **BOOTLOADER**.
3. Right-click on the **XLDR** folder and enable the **Clean Before Building** option.
4. Expand **XLDR** folder and Right-click on the **NAND** folder again and select **Build Current Project**.
5. The NAND XLDR binary image will be created in the workspace release directory.

5.6.2 Program XLDR into NAND Flash using Platform Builder

EBOOT supports downloading of the XLDR image from Platform Builder and programming it into the NAND flash. To update the XLDR image resident in NAND flash, follow these steps:

1. Build XLDR following the steps in [Section 5.6.1, Building a NAND XLDR Image](#).
2. If EBOOT and XLDR are not resident in flash or the resident EBOOT is not compatible with the current BSP, you must first follow the steps in [Section 5.4.4, 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 XLDR.nb0 found in the workspace release directory:

```
WINCE500\PBWorkspaces\MX32ADSMobility\RelDir\MX32ADS_ARMV4I_Release\XLDR.nb0
```

NOTE

Platform Builder will report an error stating that you will not be able to debug the kernel file. You can ignore this error message and select **OK**.

5. Follow the steps in [Section 5.7, 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**.

NOTE

Due to the special contents of the XLDR image, Platform Builder will not indicate that the entire image has been downloaded. This does not effect the success of programming the XLDR.

8. Switch over your terminal emulation application. At this point, XLDR 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 XLDR completed successfully.
Reboot the device manually...
SpinForever...
```

11. Close the Platform Builder workspace for XLDR.nb0. 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 5.2, i.MX32 ADS Board Configuration](#) to configure the i.MX32 CPU board for direct external boot from NAND flash.

NOTE

Booting from NAND requires the XLDR and EBOOT images to be resident in the NAND device. Be sure to follow the steps in [Section 5.4.7, Program/Update EBOOT in NAND Flash using Platform Builder](#) and [Section 5.6, XLDR Installation and Configuration](#) before attempting to boot from NAND.

14. Reset the target hardware. If you see new EBOOT messages appear on the terminal, XLDR is properly booting EBOOT from NAND flash.

5.6.3 Building a SD/MMC XLDR Image

The steps to build a XLDR image SD/MMC for flash are as follows:

1. Select the **File View** tab of the Platform Builder **Workspace** window.
2. Expand **Favorites** → **MX32ADS PLATFORM** → **src** → **BOOTLOADER**.
3. Right-click on the **XLDR** folder and enable the **Clean Before Building** option.
4. Expand **XLDR** folder and Right-click on the **SD** folder again and select **Build Current Project**.
5. The SD/MMC XLDR binary image will be created in the workspace release directory.

5.6.4 Program XLDR into SD/MMC Flash using Platform Builder

EBOOT supports downloading of the XLDR image from Platform Builder and programming it into the SD/MMC flash. To update the XLDR image resident in SD/MMC flash, follow these steps:

1. Build XLDR following the steps in [Section 5.6.3, Building a SD/MMC XLDR Image](#).
2. If EBOOT and XLDR are not resident in flash or the resident EBOOT is not compatible with the current BSP, you must first follow the steps in [Section 5.4.4, 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 XLDR.nb0 found in the workspace release directory:

```
WINCE500\PBWorkspaces\MX32ADSMobility\RelDir\MX32ADS_ARMV4I_Release\XLDR.nb0
```

NOTE

Platform Builder will report an error stating that you will not be able to debug the kernel file. You can ignore this error message and select **OK**.

5. Follow the steps in [Section 5.7, 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**.

NOTE

Due to the special contents of the XLDR image, Platform Builder will not indicate that the entire image has been downloaded. This does not effect the success of programming the XLDR.

8. Switch over your terminal emulation application. At this point, XLDR 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 XLDR completed successfully.
Reboot the device manually...
SpinForever...
```

11. Close the Platform Builder workspace for XLDR.nb0. 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 5.2, i.MX32 ADS Board Configuration](#) to configure the i.MX32 CPU board for direct external boot from SD/MMC flash.

NOTE

Booting from SD/MMC requires the XLDR and EBOOT images to be resident in the SD/MMC device. Be sure to follow the steps in [Section 5.4.8, Program/Update EBOOT in SD/MMC Flash using Platform Builder](#) and [Section 5.6, XLDR Installation and Configuration](#) before attempting to boot from SD/MMC.

14. Reset the target hardware. If you see new EBOOT messages appear on the terminal, XLDR is properly booting EBOOT from SD/MMC flash.

5.7 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 5.4.4, 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**.

5.8 Configuring Serial Connection for Downloading and Debugging

To configure an Serial 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 **Serial** as the download service.
5. Launch SBOOT on the target. After SBOOT initialization completes, you should begin to see BOOTME messages appear on the serial debug output.

NOTE

If SBOOT has already been programmed into the target, a hardware reset will start SBOOT execution. If SBOOT is being programmed into the target, refer to [Section 5.5.5, Loading SBOOT into SDRAM using RVDEBUG](#) for the steps on how to launch SBOOT.

6. To the right of the **Download** box, choose **Settings**. Configure the Serial settings, ensure it conforms to the settings as configured for the target in [Section 5.5.6, Initialize SBOOT Serial Configuration](#) and then choose **OK**.
7. In the **Transport** box, choose **Serial** as a kernel transport.
8. To the right of the **Transport** box, choose **Settings**. The Serial configuration will be the same as the settings done in Step 6 and then choose **OK**.
9. 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**.
10. Choose **Core Service Settings**.
11. To instruct Platform Builder to download a run-time image each time Platform Builder connects with the target device, under **Download Image**, choose **Always**.

12. Select **Enable KITL on device boot**.
13. Select **Clear memory on soft reset**.
14. Select **Enable access to desktop files**.
15. Choose **Apply**.
16. Choose **Close**.

Chapter 6

Downloading and Debugging Images

This section describes the procedures for downloading and debugging OS images on the i.MX32 ADS 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 5.4, EBOOT Installation and Configuration](#) to program EBOOT into the target.
2. Prepare the target hardware by following the instructions in [Section 5.2, i.MX32 ADS Board Configuration](#). Configure the i.MX32 CPU board for direct external boot from NOR/NAND flash.
3. Open the desired workspace within Platform Builder.
4. **Deselect Platform** → **Settings** → **Build Options** → **Write Run-time Image to Flash Memory**.
5. Within **Platform** → **Settings** → **Environment**, remove the IMG_NAND and IMG_MMC environment variables if it exists.
6. Build the image following the steps provided in Chapter 4.
7. Reset the i.MX32 ADS to launch EBOOT on the target.
8. If a target device connection has not been created within Platform Builder, follow the steps in [Section 5.7, 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 5.4, EBOOT Installation and Configuration](#) to program EBOOT into the target.
2. Prepare the target hardware by following the instructions in [Section 5.2, i.MX32 ADS Board Configuration](#). Configure the i.MX32 CPU board for direct external boot from NOR flash.
3. Open the desired workspace within Platform Builder.
4. **Select Platform** → **Settings** → **Build Options** → **Write Run-time Image to Flash Memory**.
5. Within **Platform** → **Settings** → **Environment**, remove the IMG_NAND and IMG_MMC environment variables if it exists.
6. Build the image following the steps provided in Chapter 4.
7. Reset the i.MX32 ADS to launch EBOOT on the target.
8. If a target device connection has not been created within Platform Builder, follow the steps in [Section 5.7, 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 i.MX32 ADS 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 “NK from 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 i.MX32 ADS to launch EBOOT again. EBOOT will automatically jump to the OS image in NOR flash after the specified boot delay.

NOTE

If the i.MX32 ADS is configured for active KITL, the image will wait for a Platform Builder connection before booting the OS image. Refer to [Table 5-2](#) for 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 and XLDR is not resident on the target, follow the procedure in [Section 5.4, EBOOT Installation and Configuration](#) and [Section 5.6.2, Program XLDR into NAND Flash using Platform Builder](#) to program EBOOT and XLDR into NAND flash memory.
2. Prepare the target hardware by following the instructions in [Section 5.2, i.MX32 ADS Board Configuration](#). Configure the i.MX32 CPU board for direct external boot from NAND flash.
3. Open the desired workspace within Platform Builder.
4. **Deselect Platform → Settings → Build Options → Write Run-time Image to Flash Memory.**
5. Within **Platform → Settings → Environment**, use the **New** button to add **IMG_NAND = 1**, remove the **IMGMMC** environment variable if it exists.
6. Build the image following the steps provided in Chapter 4.

7. Reset the i.MX32 ADS to launch EBOOT on the target.
8. If a target device connection has not been created within Platform Builder, follow the steps in [Section 5.7, 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 i.MX32 ADS 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 “NK from 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 i.MX32 ADS 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 i.MX32 ADS is configured for active KITL, the image will wait for a Platform Builder connection before booting the OS image. Refer to [Table 5-2](#) for active/passive KITL configuration.

6.6 Downloading an OS Image into SD/MMC Flash using EBOOT

To download an image into the SD/MMC Flash of the target, follow these steps:

1. If EBOOT and XLDR is not resident on the target, follow the procedure in [Section 5.4, EBOOT Installation and Configuration](#) and [Section 5.6.4, Program XLDR into SD/MMC Flash using Platform Builder](#) to program EBOOT and XLDR into SD/MMC flash memory.
2. Prepare the target hardware by following the instructions in [Section 5.2, i.MX32 ADS Board Configuration](#). Configure the i.MX32 CPU board for direct external boot from SD/MMC flash.
3. Open the desired workspace within Platform Builder.

4. *Deselect Platform* → **Settings** → **Build Options** → **Write Run-time Image to Flash Memory**.
5. Within **Platform** → **Settings** → **Environment**, use the **New** button to add **IMGMMC = 1**, remove the **IMGNAND** environment variable if it exists.
6. Build the image following the steps provided in Chapter 4.
7. Reset the i.MX32 ADS to launch EBOOT on the target.
8. If a target device connection has not been created within Platform Builder, follow the steps in [Section 5.7, 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...
```

6.7 Running an OS Image from SD/MMC Flash using EBOOT

To execute an OS image from SD/MMC flash that has been previously programmed using the procedure described in [Section 6.6, Downloading an OS Image into SD/MMC Flash using EBOOT](#), follow these steps.

1. Reset the i.MX32 ADS 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 “NK from SD/MMC” 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 i.MX32 ADS to launch EBOOT again. EBOOT will automatically load the OS image from SD/MMC to RAM and execute it after the specified boot delay.

NOTE

If the i.MX32 ADS is configured for active KITL, the image will wait for a Platform Builder connection before booting the OS image. Refer to [Table 5-2](#) for active/passive KITL configuration.

6.8 Downloading an Image into SDRAM using SBOOT

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

1. If SBOOT is not resident on the target, follow the procedure in [Section 5.5, SBOOT Installation and Configuration](#) to program SBOOT into the target.
2. Prepare the target hardware by following the instructions in [Section 5.2, i.MX32 ADS Board Configuration](#). Configure the i.MX32 CPU board for direct external boot from NOR/NAND flash.
3. Open the desired workspace within Platform Builder.
4. *Deselect Platform* → **Settings** → **Build Options** → **Write Run-time Image to Flash Memory**.
5. Within **Platform** → **Settings** → **Environment**, remove the IMG_NAND and IMG_MMC environment variables if it exists.
6. Build the image following the steps provided in Chapter 4.
7. Reset the i.MX32 ADS to launch SBOOT on the target.
8. If a target device connection has not been created within Platform Builder, follow the steps in [Section 5.8, Configuring Serial Connection for Downloading and Debugging](#) to establish a connection.
9. From the Platform Builder **Target** menu, select **Attach Device** to begin the download.

6.9 Downloading an OS Image into NOR Flash using SBOOT

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

1. If SBOOT is not resident on the target, follow the procedure in [Section 5.5, SBOOT Installation and Configuration](#) to program SBOOT into the target.
2. Prepare the target hardware by following the instructions in [Section 5.2, i.MX32 ADS Board Configuration](#). Configure the i.MX32 CPU board for direct external boot from NOR flash.
3. Open the desired workspace within Platform Builder.
4. *Select Platform* → **Settings** → **Build Options** → **Write Run-time Image to Flash Memory**.
5. Within **Platform** → **Settings** → **Environment**, remove the IMG_NAND and IMG_MMC environment variables if it exists.
6. Build the image following the steps provided in Chapter 4.
7. Reset the i.MX32 ADS to launch SBOOT on the target.
8. If a target device connection has not been created within Platform Builder, follow the steps in [Section 5.8, Configuring Serial 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...
```

6.10 Running an OS Image from NOR Flash using SBOOT

To execute an OS image from NOR flash that has been previously programmed using the procedure described in [Section 6.9, Downloading an OS Image into NOR Flash using SBOOT](#), follow these steps.

1. Reset the i.MX32 ADS to launch SBOOT 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 SBOOT configuration menu.
4. Continue selecting the **Autoboot** option of the SBOOT menu until “NK from NOR” is selected.
5. Specify the desired boot delay using the **Boot Delay** menu option.
6. Save the configuration using the SBOOT menu.
7. Reset the i.MX32 ADS to launch SBOOT again. SBOOT will automatically jump to the OS image in NOR flash after the specified boot delay.

NOTE

If the i.MX32 ADS is configured for active KITL, the image will wait for a Platform Builder connection before booting the OS image. Refer to [Table 5-2](#) for active/passive KITL configuration.

6.11 Downloading an OS Image into NAND Flash using SBOOT

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

1. If SBOOT and XLDR is not resident on the target, follow the procedure in [Section 5.5, SBOOT Installation and Configuration](#) and [Section 5.6.2, Program XLDR into NAND Flash using Platform Builder](#) to program SBOOT and XLDR into NAND flash memory.
2. Prepare the target hardware by following the instructions in [Section 5.2, i.MX32 ADS Board Configuration](#). Configure the i.MX32 CPU board for direct external boot from NAND flash.
3. Open the desired workspace within Platform Builder.
4. **Deselect Platform** → **Settings** → **Build Options** → **Write Run-time Image to Flash Memory**.
5. Within **Platform** → **Settings** → **Environment**, use the **New** button to add **IMG_NAND = 1**, remove the **IMGMMC** environment variable if it exists.
6. Build the image following the steps provided in Chapter 4.
7. Reset the i.MX32 ADS to launch SBOOT on the target.
8. If a target device connection has not been created within Platform Builder, follow the steps in [Section 5.8, Configuring Serial 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 SBOOT has downloaded the image temporarily into SDRAM and is ready to begin the flash programming procedure.
12. SBOOT 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...
```

6.12 Running an OS Image from NAND Flash using SBOOT

To execute an OS image from NAND flash that has been previously programmed using the procedure described in [Section 6.11, Downloading an OS Image into NAND Flash using SBOOT](#), follow these steps.

1. Reset the i.MX32 ADS to launch SBOOT 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 SBOOT configuration menu.
4. Continue selecting the **Autoboot** option of the SBOOT menu until “NK from NAND” is selected.
5. Specify the desired boot delay using the **Boot Delay** menu option.
6. Save the configuration using the SBOOT menu.
7. Reset the i.MX32 ADS to launch SBOOT again. SBOOT will automatically load the OS image from NAND to RAM and execute it after the specified boot delay.

NOTE

If the i.MX32 ADS is configured for active KITL, the image will wait for a Platform Builder connection before booting the OS image. Refer to [Table 5-2](#) for active/passive KITL configuration.

6.13 Downloading an OS Image into SD/MMC Flash using SBOOT

To download an image into the SD/MMC Flash of the target, follow these steps:

1. If SBOOT and XLDR is not resident on the target, follow the procedure in [Section 5.5, SBOOT Installation and Configuration](#) and [Section 5.6.4, Program XLDR into SD/MMC Flash using Platform Builder](#) to program SBOOT and XLDR into SD/MMC flash memory.
2. Prepare the target hardware by following the instructions in [Section 5.2, i.MX32 ADS Board Configuration](#). Configure the i.MX32 CPU board for direct external boot from SD/MMC flash.
3. Open the desired workspace within Platform Builder.
4. **Deselect Platform → Settings → Build Options → Write Run-time Image to Flash Memory.**
5. Within **Platform → Settings → Environment**, use the **New** button to add IMGMMC = 1, remove the IMG NAND environment variable if it exists.
6. Build the image following the steps provided in Chapter 4.
7. Reset the i.MX32 ADS to launch SBOOT on the target.

8. If a target device connection has not been created within Platform Builder, follow the steps in [Section 5.7, 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 SBOOT has downloaded the image temporarily into SDRAM and is ready to begin the flash programming procedure.
12. SBOOT 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...
```

6.14 Running an OS Image from SD/MMC Flash using SBOOT

To execute an OS image from SD/MMC flash that has been previously programmed using the procedure described in [Section 6.13, Downloading an OS Image into SD/MMC Flash using SBOOT](#), follow these steps.

1. Reset the i.MX32 ADS to launch SBOOT 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 SBOOT configuration menu.
4. Continue selecting the **Autoboot** option of the SBOOT menu until “NK from SD/MMC” is selected.
5. Specify the desired boot delay using the **Boot Delay** menu option.
6. Save the configuration using the SBOOT menu.
7. Reset the i.MX32 ADS to launch SBOOT again. SBOOT will automatically load the OS image from SD/MMC to RAM and execute it after the specified boot delay.

NOTE

If the i.MX32 ADS is configured for active KITL, the image will wait for a Platform Builder connection before booting the OS image. Refer to [Table 5-2](#) for active/passive KITL configuration.

Chapter 7

i.MX32 BSP Features

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 this document for more information regarding the requirements, implementation, and testing for each of the i.MX32 BSP features.