i.MX25 PDK Windows Embedded CE 6.0

User's Guide

Document Number: 924-76355
Rev. 2009.12
1/2010

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

Contents

About This Book

This User's Guide describes the Freescale i.MX25 3-Stack Platform System Windows Embedded CE 6.0 board support package (BSP). This document describes how to install the BSP and how to use Microsoft Platform Builder for Windows CE 6.0 to build, download, and debug OS images. Information on the configuration and usage of features included within the Freescale BSP is also provided.

Audience

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

Organization

This document is organized into the following chapters:

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

Suggested Reading

Additional information regarding Microsoft Windows CE and the i.MX25 can be found in these documents. These documents were also referenced to produce this document.

- i.MX25 PDK Windows Embedded CE 6.0 Reference Manual
- i.MX25 PDK Windows Embedded CE 6.0 Release Notes
- http://msdn.microsoft.com/embedded/windowsce
- Windows Embedded Training Resources: http://www.windowsembedded.com/training
- MCP Certification for Windows Embedded CE: http://www.windowsembedded.com/certification
- Windows Embedded CE 6.0 Online Help

Conventions

This document uses the following notational conventions:

- `Courier` indicate commands, command parameters, code examples, expressions, datatypes, and directives.
- *Italic* 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:

3DS	3-Stack 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
KITL	kernel independent transport layer
LVDS	low-voltage differential signaling
MAC	media access control
OAL	OEM adaptation layer
OEM	original equipment manufacturer
OS	operating system
PQOAL	production quality OEM adaptation layer
RVDEBUG	RealView debugger
RVI	RealView ICE
SDC	synchronous display controller
SDRAM	synchronous dynamic random access memory
SoC	system on a chip

Chapter 1
BSP Installation

The BSP is distributed as a Microsoft Installer (.msi) file that includes support for the i.MX25 3-Stack platform system. Refer to the *i.MX25 Windows Embedded CE 6.0 Reference Manual* for additional instructions and information before installing and using this BSP.

1.1 Install Windows Embedded Tools and i.MX25 3-Stack BSP

The following steps install the Microsoft Windows Embedded CE 6.0 development tools and the Freescale i.MX25 3-Stack BSP to provide a complete i.MX25 3-Stack development environment.

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

   NOTE

   During Platform Builder installation, be sure to select which versions of the operating system to include. The **ARMV4I** version must be installed on the local hard drive for this BSP. All other versions are not required. Refer to the *Release Notes* for information about any additional Microsoft-supplied QFEs or Service Packs that also need to be installed.

2. If you have previously installed an earlier version of the Freescale i.MX25 3-Stack BSP, remove any existing BSP files by following the instructions in Section 1.2, "Uninstall Freescale i.MX25 3-Stack BSP".
3. Download the Freescale i.MX25 3-Stack BSP and install the contents into the existing WINCE600 top-level folder (the MSI installer automatically creates the necessary subfolders so that all of the files are installed into the correct location).

1.2 Uninstall Freescale i.MX25 3-Stack BSP

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

NOTE

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

The following steps remove the BSP:

1. Close the Platform Builder.

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

```
WINCE600\OSDesigns\iMX25-3DS-Mobility-PDK1_7
WINCE600\PLATFORM\COMMON\SRC\SOC\COMMON_FSL_V2_PDK1_7
WINCE600\PLATFORM\COMMON\SRC\SOC\MX25_FSL_V2_PDK1_7
WINCE600\PLATFORM\iMX25-3DS-PDK1_7
WINCE600\SUPPORT_PDK1_7
```

4. Manually remove the residual BSP library files that were created after the BSP was installed. To find the libraries, use Windows Explorer with the search name `*FSL_V2*` and the following library path and remove all the `LIB`, `PDB`, and `DEF` files found by the search:

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

1.3 Load Sample OS Design Solution

After installing the BSP, use the sample OS solutions provided in the BSP package to build a Windows Embedded CE 6.0 OS image as follows:

1. In Platform Builder, select **File** > **Open** > **Project/Solution...**
2. Select the i.MX25 3-Stack BSP sample workspace by loading the following files:

```
WINCE600\OSDesigns\iMX25-3DS-PDK1_7-Mobility\iMX25-3DS-PDK1_7-Mobility.sln
WINCE600\OSDesigns\iMX25-3DS-PDK1_7-SmallFootprint\iMX25-3DS-PDK1_7-SmallFootprint.s
ln
```

The process of loading this solution also automatically loads the associated i.MX25 3-Stack BSP catalog. Select the **Catalog Items View** tab to see all of the available OS solution catalog items.

Chapter 2
BSP Contents and Organization

The Freescale BSP is a collection of code and support files that can be integrated into the Microsoft Platform Builder development environment to create Windows Embedded CE 6.0 OS images for the i.MX25 3-Stack platform system. The BSP contains the following elements:

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

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

The i.MX25 3-Stack system-on-a-chip (SoC) leverages a common Freescale ARM-based platform architecture. This platform is found in a series of ARM-based SoCs available from Freescale. In order to leverage source code that is portable across multiple Freescale ARM-based SoCs, a common directory called COMMON_FSL_V2_PDK1_7 is used to store shared OAL and driver components. This COMMON_FSL_V2_PDK1_7 common directory appears in the chip support package and PQOAL directories described below.

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

The Freescale SOC directory contains a collection of chipset-level code that can be leveraged to develop platforms based on the i.MX25 3-Stack SoC and the PQOAL components customized for the i.MX25. The code and definitions in the SOC directory can be reused in a new platform design without modification. To keep the SOC sources platform agnostic, source code in the SOC directory utilizes hardware abstraction routines that must be ported to a specific platform or board. The SOC source code is compiled into a set of static libraries that are ultimately linked with platform-specific libraries to create drivers for the system.

2.1.1 Production Quality Driver (PQD)

Windows Embedded CE 6.0 supports PQD components that simplify and shorten the process of developing a driver. For more information on PQQL development concepts, refer to the "Production-Quality Drivers" topic in the *Windows Embedded CE 6.0 Help*.

The following directories contain the SOC driver source code for the i.MX25 3-Stack:

```
WINCE600\PLATFORM\COMMON\SRC\SOC\COMMON_FSL_V2_PDK1_7
WINCE600\PLATFORM\COMMON\SRC\SOC\MX25_FSL_V2_PDK1_7
```

SOC code in the `COMMON_FSL_V2_PDK1_7` directory is reusable across all Freescale ARM-based SoCs. SOC driver code in the `MX25_FSL_V2_PDK1_7` directory is reusable across all platforms based on the i.MX25 3-Stack.

2.1.2 Production Quality OAL (PQOAL)

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

Where possible, the Freescale BSP leverages the PQOAL architecture and components provided by Microsoft to reduce the OAL code that needs to be modified and maintained by the OEM. In addition, PQOAL components customized for the i.MX25 3-Stack are available in the following directories:

```
WINCE600\PLATFORM\COMMON\SRC\SOC\COMMON_FSL_V2_PDK1_7\OAL
WINCE600\PLATFORM\COMMON\SRC\SOC\MX25_FSL_V2_PDK1_7\OAL
```

PQOAL code in the `COMMON_FSL_V2_PDK1_7\OAL` directory is reusable across all Freescale ARM-based SoCs. PQOAL code in the `MX25_FSL_V2_PDK1_7\OAL` directory is reusable across all platforms based on the i.MX25 3-Stack.

2.2 i.MX25 3-Stack Platform Files

The i.MX25 3-Stack BSP provides direct support for the interfaces and peripherals found on the i.MX25 3-Stack board. All of the driver and OAL content that is specific to the underlying hardware platform is located in the following directory:

```
WINCE600\PLATFORM\iMX25-3DS-PDK1_7
```

The i.MX25 3-Stack platform directory implements the hardware abstraction routines invoked by driver code in the Freescale SOC directory. In addition, this directory implements certain aspects of the PQOAL that may need to be modified by the OEM for their specific platform.

2.3 Sample OS Design Solutions

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

```
WINCE600\OSDesigns\iMX25-3DS-PDK1_7-Mobility
```

There is another solution included in the BSP that provides the starting point for the smallest functional Windows Embedded CE 6.0 run-time image:

```
WINCE600\OSDesigns\iMX25-3DS-PDK1_7-SmallFootprint
```

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

2.4 Support Files

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

```
WINCE600\SUPPORT_PDK1_7
```

This directory is used to store applications and tests targeted for the supported platforms. Support files necessary to configure development tools are also placed in this directory.

Chapter 3
Configuring OS Images

The i.MX25 3-Stack BSP provides a set of configuration options, allowing users to configure the BSP and build the OS image per their specific needs. This section will describe how to configure these options, using either the sample workspace provided along with the i.MX25 3-Stack BSP or a custom workspace created through the Windows Embedded CE 6.0 OS Design Wizard (launched from Platform Builder menu item **File > New > Project...**)

NOTE

The sample workspace provided within the i.MX25 3-Stack BSP is configured to generate a default image targeted for the i.MX25 3-Stack 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 the Platform Builder can be controlled using the **Build** > **Configuration Manager...** menu item and choosing the appropriate **Active Solution Configuration** item from the drop-down list. The sample workspace provided with the i.MX25 3-Stack BSP provides the following two image build types:

- **Freescale_i_MX25_3DS_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.
- **Freescale_i_MX25_3DS_ARMV4I_Debug**—Debug build that includes KITL and kernel debugger support. This image type provides full debug capability at the expense of a larger image size and slower execution.

NOTE

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

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

3.2 BSP Environment Variables

There are three types of BSP environment variables used to configure the BSP: BSP_NOXXX, BSP_XXX and IMGXXX.

3.2.1 BSP_NOXXX Variables

The i.MX25 3-Stack BSP supports the Windows Embedded CE 6.0 dependency feature, which simplifies the BSP configuration process. Support for this feature means that the selection of certain SYSGEN components in the workspace (e.g. SYSGEN_DISPLAY, SYSGEN_AUDIO) will trigger the automatic selection of certain BSP drivers (e.g. Display, Audio). The drivers that have a SYSGEN dependency will have a corresponding BSP_NOXXX variable defined in the platform batch file listed below. These variables provide a means for excluding a driver from the OS image that might otherwise be included, due to a SYSGEN dependency.

```
WINCE600\PLATFORM\iMX25-3DS-PDK1_7\iMX25-3DS-PDK1_7.bat
```

NOTE

It is not possible to remove the selection of these drivers by clicking the items in Catalog UI. Instead, users need to set the corresponding BSP_NOXXX variables as ‘1’ in **Environment** dialog (launched from Platform Builder menu **Project > Properties > Configuration Properties > Environment**).

Table 3-1 provides a summary of the BSP_NOXXX environment variables defined in batch file.

Table 3-1. BSP_NOXXX Variables in Batch File

Variable Name	Description	Settings
BSP_NOAUDIO	Used to exclude support for SSI Audio driver.	BSP_NOAUDIO=1 Excludes audio driver support from the OS image despite the SYSGEN_AUDIO dependency.
BSP_NOBATTERY	Used to exclude support for SSI Battery driver.	BSP_NOBATTERY=1 Excludes audio driver support from the OS image despite the SYSGEN_BATTERY dependency.
BSP_NOCSPDDK	Used to exclude CSP driver development kit (CSPDDK) support. The CSPDDK is required for most BSP device drivers.	BSP_NOCSPDDK=1 Excludes CSPDDK from the OS image despite the SYSGEN_CEDDK dependency. Only use this configuration when building an OS design that does not include BSP device drivers.
BSP_NODISPLAY	Used to exclude support for Display driver.	BSP_NODISPLAY=1 Excludes display driver support from the OS image despite the SYSGEN_DISPLAY dependency.
BSP_NOFEC	Used to exclude support for FEC driver.	BSP_NOFEC=1 Excludes FEC driver support from the OS image despite the SYSGEN_NDIS dependency.
BSP_NOESDHC	Used to exclude support for eSDHC driver.	BSP_NOESDHC=1 Excludes eSDHC driver support from the OS image despite the SYSGEN_SDBUS dependency.
BSP_NOKEYPAD	Used to exclude support for Keypad driver.	BSP_NOKEYPAD=1 Excludes Keypad driver support from the OS image despite the SYSGEN_MININPUT dependency.

Table 3-1. BSP_NOXXX Variables (continued)in Batch File

Variable Name	Description	Settings
BSP_NONANDFMD	Used to exclude support for NAND Flash driver.	BSP_NONANDFMD=1 Excludes NAND Flash driver support from the OS image despite the SYSGEN_FLASHMDD dependency.
BSP_NOSIM	Used to exclude support for SIM driver.	BSP_NOSIM=1 Excludes SIM driver support from the OS image despite the SYSGEN_SMARTCARD dependency.
BSP_NOTOUCH	Used to exclude support for Touch driver.	BSP_NOTOUCH=1 Excludes Touch driver support from the OS image despite the SYSGEN_TOUCH dependency.
BSP_NOUSB	Used to exclude support for all USB drivers.	BSP_NOUSB=1 Excludes all USB drivers support from the OS image despite the SYSGEN_USB and SYSGEN_USBFN dependency.

NOTE

The opposite setting of "BSP_NOXXX=1" would be either "BSP_NOXXX=" which is set in the batch file by default, or not to define the BSP_NOXXX variable at all. Thus, the driver will be included by SYSGEN selection.

Though the drivers listed above have direct SYSGEN dependency, not all of them can be automatically selected by SYSGEN dependency for the possible reasons below.

1. The driver has multiple subordinate selections, for example there are three selections K9LBG08U0D, K9LAG08U0M and K9LBG08U0M for NAND Flash.
2. The driver depends on another driver, for example Audio depends on I2C1 driver, and Touch depends on ADC driver.
3. The driver conflicts with other drivers, for example SIM driver conflicts with Camera and ESAI drivers.

3.2.2 BSP_XXX Variables

The i.MX25 3-Stack BSP uses BSP_XXX variables defined in Catalog to configure the drivers that can not automatically be selected by SYSGEN dependency. In these cases, users can use Platform Builder Catalog UI to select/deselect drivers by clicking the catalog items. The *Windows Embedded CE 6.0 BSP for i.MX25 3-Stack Reference Manual* describes the BSP_XXX variables for individual driver.

NOTE

Users might not always select the desired drivers successfully in Catalog UI. Instead, users will possibly get a small red 'x' for some cases, which means there are some dependency or conflict involved in the selection. In this case, users need to right-click the catalog item and see the details in 'Reasons for Exclusion of Item'. In order to get a successful selection, follow the info to select the required SYSGEN components or drivers and to deselect conflicting components.

NOTE

When creating custom workspace using Platform Builder wizard, users are always expected to go through BSP catalog items under **Third party > BSP > Freescale i.MX25 3DS: ARMV4I** to manually select those desired drivers which may not be automatically selected by SYSGEN dependency.

3.2.3 IMG_XXX Variables

The IMG_XXX variables are used to control Make Run-Time Image procedure, and can be viewed and configured within the **Environment** dialog as follows:

1. Open the sample solution for the i.MX25 3-Stack BSP.
2. From the **Project** menu, choose **Properties**.
3. Expand **Configuration Properties** if necessary and select the **Environment** item.

Table 3-2 provides a summary of the IMG_XXX environment variables available on i.MX25 3-Stack BSP.

Table 3-2. IMG_XXX Variables

Variable Name	Description	Settings
IMGNAND	Used by EBOOT and OS build files to determine if images are targeted for NAND Flash.	IMGNAND = 1 Link EBOOT and OS images for NAND Flash.
IMGSDMMC	Used by EBOOT and OS build files to determine if images are targeted for SD/MMC card.	IMGSDMMC = 1 Link EBOOT and OS images for SD/MMC card.
IMGCSPIFLASH	Used by EBOOT build files to determine if images are targeted for SPI Flash.	IMGCSPIFLASH = 1 Link EBOOT images for SPI Flash.

3.3 Catalog Environment Variables

The i.MX25 3-Stack BSP utilizes the Platform Builder Windows CE Catalog to allow users to configure BSP components in the OS design. The *i.MX25 Windows Embedded CE 6.0 Reference Manual* describes the environment variables associated with each of the BSP features exposed in the i.MX25 3-Stack BSP Catalog.

Chapter 4
Building OS Images

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

4.1 Build the Freescale SOC Libraries

The Freescale SOC libraries that support the i.MX25 3-Stack are generated during the **Build** > **Advanced Build Commands** > **Sysgen** or **Build** > **Advanced Build Commands** > **Build Current BSP and Subprojects** build procedures. Windows CE ships with pre-built SOC libraries for various ARM processors, but the libraries for the i.MX25 3-Stack must be built from the sources since they are not included with the standard Microsoft distribution.

The sample OS design solution that is provided is already preconfigured to build the Freescale SOC that is required. That is, the sample i.MX25 3-Stack OS design solution automatically builds the i.MX25 SOC sources.

Follow these steps to build the Freescale SOC libraries:

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

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

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

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

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

4.2 Build Run-Time Images

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

4.2.1 Build the BSP for the First Time

1. Open the sample solution.
2. Select the desired build type discussed in Section 3.1, “Image Build Type”.
3. Using the **Build** > **Global Build Settings** menu, configure the build options as follows:

   - Select **Copy Files to Release Directory After Build**
   - Select **Make Run-Time Image After Build**
4. Select **Build** > **Advanced Build Commands** > **Sysgen** to start the build process.

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

```
WINCE600\OSDesigns\iMX25-3DS-PDK1_7-Mobility\RelDir\Freescale_i_MX25_3DS_ARMV4I_Release
```

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

```
WINCE600\OSDesigns\iMX25-3DS-PDK1_7-Mobility\RelDir\Freescale_i_MX25_3DS_ARMV4I_Debug
```

4.2.2 Clean Build for the BSP

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

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

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

4.2.3 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 the **OS design**. Instead, the user can perform an incremental build of the Freescale BSP components to quickly build an updated OS image as follows:

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

Chapter 5
Preparing for Download and Debug

The target and development workstation must be properly configured and initialized before OS images can be downloaded and executed. This section discusses the steps required to prepare the target and development workstation so that the 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 describes the configuration of the desktop workstation and Freescale BSP to support serial debug messages.

5.1.1 Desktop Workstation Serial Debug Port

Any terminal emulation application can be used to display messages sent from the serial port of the target. Configure the 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.MX25 3-Stack BSP has been configured to use internal UART1 routed to the UARTC debugger board connector for serial debug communication. UARTC on the 3-Stack debugger board is the DB-9 connector next to the Ethernet port. Connect a standard serial cable between UARTC of the i.MX25 3-Stack debugger board and the development workstation.

5.2 Board Configuration

5.2.1 i.MX25 3-Stack Board Configuration

This section describes the switch and jumper configuration required for proper operation of the BSP on the i.MX25 3-Stack board. For specific BSP features, the 3-Stack board may need to be reconfigured to support the necessary interface. Refer to the *i.MX25 Windows Embedded CE 6.0 Reference Manual* for board configuration required by some BSP features.

5.2.2 i.MX25 3-Stack Boot Mode Settings

Table 5-1 describes the configuration for the i.MX25 3-Stack Boot Mode.

Table 5-1. CPU Boot Mode Settings

Jumper/Switch Setting	Configuration
NAND Flash (K9LBG08U0D) Boot Configuration	
SW9 on Debug board	0
SW10 on Debug board	0
SW21 on Personality board	1, 4, 6 ON, others OFF
SW22 on Personality board	3, 4 ON, others OFF
NAND Flash (K9LAG08U0M) Boot Configuration	
SW9 on Debug board	0
SW10 on Debug board	0
SW21 on Personality board	1, 4, 5 ON, others OFF
SW22 on Personality board	3 ON, others OFF
SD/MMC Boot Configuration	
SW9 on Debug board	0
SW10 on Debug board	0
SW21 on Personality board	1, 2 ON, others OFF
SW22 on Personality board	All OFF
SPI Flash Boot Configuration	
SW9 on Debug board	0
SW10 on Debug board	0
SW21 on Personality board	1, 2, 3, 4, 7 ON, others OFF
SW22 on Personality board	All OFF
Bootstrap Configuration Used by ATK and ICE	
SW9 on Debug board	1
SW10 on Debug board	1
SW21 on Personality board	IGNORED
SW22 on Personality board	IGNORED

5.3 EBOOT Installation and Configuration

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

5.3.1 Build an EBOOT Image for SD/MMC Card

Build EBOOT for a SD/MMC card by following these steps:

1. Follow the steps in Section 4.2, “Build Run-Time Images” to build a retail OS image. During the process of building the OS image, the libraries needed by EBOOT are created.
2. Specify the targeted memory for the EBOOT image as SD/MMC card. This is achieved by setting the environment variable IMGSDMMC=1 within **Project** > **Properties** > **Configuration Properties** > **Environment**. This causes the EBOOT image to be linked for a SD/MMC card.
3. Select the **Solution Explorer** tab of the Platform Builder window.
4. Expand **iMX25-3DS-PDK1_7-Mobility** > **D:/WINCE600** > **PLATFORM** > **iMX25-3DS-PDK1_7** > **src** > **BOOTLOADER**.

NOTE

The drive letter **D** may vary per the actual Platform Builder installation path.

5. Right-click on the **EBOOT** folder and select **Rebuild**.
6. The EBOOT binary image is created in the workspace release directory.

5.3.2 Build an XLDR Image for SD/MMC Card

Build XLDR for a SD/MMC card by following these steps:

1. Follow the steps in Section 4.2, “Build Run-Time Images” to build a retail OS image. During the process of building the OS image, the libraries needed by XLDR are created.
2. Select the **Solution Explorer** tab of the Platform Builder window.
3. Expand **iMX25-3DS-PDK1_7-Mobility** > **D:/WINCE600** > **PLATFORM** > **iMX25-3DS-PDK1_7** > **src** > **BOOTLOADER** > **XLDR.**
4. Right-click on the **SD** folder and select **Rebuild**.
5. The XLDR binary image is created in the workspace release directory.

5.3.3 Build an EBOOT Image for NAND Flash

Build EBOOT for NAND Flash by following these steps:

1. Follow the steps in Section 4.2, “Build Run-Time Images” to build a retail OS image. During the process of building the OS image, the libraries needed by EBOOT are created.
2. Specify the targeted memory for the EBOOT image as NAND Flash. This is achieved by setting the environment variable IMGNAND=1 within **Project** > **Properties** > **Configuration Properties** > **Environment**. This causes the EBOOT image to be linked for NAND Flash.
3. Select the **Solution Explorer** tab of the Platform Builder window.
4. Expand **iMX25-3DS-PDK1_7-Mobility** > **D:/WINCE600** > **PLATFORM** > **iMX25-3DS-PDK1_7** > **src**
5. Right-click on the **COMMON** folder and select **Rebuild**.
6. Expand **iMX25-3DS-PDK1_7-Mobility** > **D:/WINCE600** > **PLATFORM** > **iMX25-3DS-PDK1_7** > **src** > **BOOTLOADER**.

7. Right-click on the **EBOOT** folder again and select **Rebuild**.
8. The EBOOT binary image is created in the workspace release directory.

5.3.4 Build an XLDR Image for NAND Flash

Build XLDR for NAND Flash by following these steps:

1. Follow the steps in Section 4.2, "Build Run-Time Images" to build a retail OS image. During the process of building the OS image, the libraries needed by XLDR are created.
2. Select the **Solution Explorer** tab of the Platform Builder window.
3. Expand **iMX25-3DS-PDK1_7-Mobility** > **D:/WINCE600** > **PLATFORM** > **iMX25-3DS-PDK1_7** > **src** > **BOOTLOADER** > **XLDR.**
4. Right-click on the **NAND** folder and select **Rebuild**.
5. The XLDR binary image is created in the workspace release directory.

5.3.5 Build an EBOOT Image for SPI Flash

Build EBOOT for SPI Flash by following these steps:

1. Follow the steps in Section 4.2, "Build Run-Time Images" to build a retail OS image. During the process of building the OS image, the libraries needed by EBOOT are created.
2. Specify the targeted memory for the EBOOT image as SPI Flash. This is achieved by setting the environment variable IMGCSPIFLASH=1 within **Project** > **Properties** > **Configuration Properties** > **Environment**. This causes the EBOOT image to be linked for SPI Flash.
3. Select the **Solution Explorer** tab of the Platform Builder window.
4. Expand **iMX25-3DS-PDK1_7-Mobility** > **D:/WINCE600** > **PLATFORM**> **iMX25-3DS-PDK1_7** > **src**
5. Right-click on the **COMMON** folder and select **Rebuild**.
6. Expand **iMX25-3DS-PDK1_7-Mobility** > **D:/WINCE600** > **PLATFORM** > **iMX25-3DS-PDK1_7** > **src** > **BOOTLOADER**.
7. Right-click on the **EBOOT** folder and select **Rebuild**.
8. The EBOOT binary image is created in the workspace release directory.

5.3.6 Build an XLDR Image for SPI Flash

Build XLDR for SPI Flash by following these steps:

1. Follow the steps in Section 4.2, "Build Run-Time Images" to build a retail OS image. During the process of building the OS image, the libraries needed by XLDR are created.
2. Select the **Solution Explorer** tab of the Platform Builder window.
3. Expand **iMX25-3DS-PDK1_7-Mobility** > **D:/WINCE600** > **PLATFORM** > **iMX25-3DS-PDK1_7** > **src** > **BOOTLOADER** > **XLDR.**
4. Right-click on the **CSPI** folder and select **Rebuild**.
5. The XLDR binary image is created in the workspace release directory.

5.3.7 Initialize EBOOT Network Configuration

Initialize the EBOOT network configuration, by following these steps:

1. Follow the steps in Section 5.3.15, "Program XLDR and EBOOT in SD/MMC Card Using ATK Tool" or Section 5.3.16, "Program XLDR and EBOOT in NAND Flash using ATK Tool" 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. Press 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 automatically displays 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. Press 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 static networking parameters are required, use the **IP Address** and **Subnet Mask** menu options.
7. Once the desired network configuration appears in the EBOOT menu, select the **Save Configuration** menu option to save the configuration to non-volatile memory.

5.3.8 Program/Update XLDR in SD/MMC Card Using Platform Builder

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

1. Build XLDR for SD/MMC card following the steps in Section 5.3.2, "Build an XLDR Image for SD/MMC Card".
2. If EBOOT is not resident in Flash or the resident EBOOT is not compatible with the current BSP, first follow the steps in Section 5.3.15, "Program XLDR and EBOOT in SD/MMC Card Using ATK Tool" to get a valid copy of EBOOT running from SD/MMC.
3. Use the **File** menu of the Platform Builder and choose **Project/Solution**.
4. Select XLDR.bin found in the appropriate workspace release directory:

```
WINCE600\OSDesigns\iMX25-3DS-PDK1_7-Mobility\RelDir\Freescale_i_MX25_3DS_ARMV4I_Rel
ease\
XLDR.bin
```

5. Follow the steps in Section 5.4, "Configure Ethernet Connection for Download and Debug" to establish an Ethernet connection between the target and the 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 to the 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, press the “y” key to begin programming the Flash.
10. EBOOT programs the image into SD/MMC card and provides status using serial debug messages. Once the programming is complete, the following messages are seen:
    ```
    INFO: Update of XLDR completed successfully.
    Reboot the device manually...
    SpinForever...
    Do you want to reset [Y\N]
    ```
11. Close the Platform Builder workspace for XLDR.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. If you updated EBOOT in SD/MMC card, use the instructions provided in Section 5.2.2, “i.MX25 3-Stack Boot Mode Settings” to configure the i.MX25 3-Stack board for direct external boot from a SD/MMC card. Otherwise follow the steps in Section 5.3.8, “Program/Update XLDR in SD/MMC Card Using Platform Builder” to update EBOOT in SD/MMC card and skip step 14.
14. Reset the target hardware. If you see new EBOOT messages appear on the terminal, XLDR has been properly programmed into SD/MMC card.

5.3.9 Program/Update EBOOT in SD/MMC Card Using Platform Builder

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

1. Build EBOOT for SD/MMC card following the steps in Section 5.3.1, “Build an EBOOT Image for SD/MMC Card”.
2. If EBOOT is not resident in Flash or the resident EBOOT is not compatible with the current BSP, first follow the steps in Section 5.3.15, “Program XLDR and EBOOT in SD/MMC Card Using ATK Tool” to get a valid copy of EBOOT running from SD/MMC.
3. Use the **File** menu of the Platform Builder and choose **Project/Solution**.
4. Select EBOOT.bin found in the workspace release directory:
    ```
    WINCE600\OSDesigns\iMX25-3DS-PDK1_7-Mobility\RelDir\Freescale_i_MX25_3DS_ARMV4I_Rel
    ease\
    EBOOT.bin
    ```
5. Follow the steps in Section 5.4, “Configure Ethernet Connection for Download and Debug” 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 to the 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, press the “y” key to begin programming the Flash.
10. EBOOT programs the image into SD/MMC card and provides status using serial debug messages. Once the programming is complete, the following messages are seen:

```
INFO: Update of EBOOT/SBOOT completed successfully.
Reboot the device manually...
SpinForever...
Do you want to reset [Y\N]
```

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 5.2.2, "i.MX25 3-Stack Boot Mode Settings" to configure the i.MX25 3-Stack board for direct external boot from SD/MMC card.
14. Reset the target hardware. If you see new EBOOT messages appear on the terminal, EBOOT has been properly programmed into SD/MMC card.

5.3.10 Program/Update XLDR in NAND Flash Using Platform Builder

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

1. Build XLDR for NAND following the steps in Section 5.3.4, "Build an XLDR Image for NAND Flash".
2. If EBOOT is not resident in Flash or the resident EBOOT is not compatible with the current BSP, first follow the steps in Section 5.3.16, "Program XLDR and EBOOT in NAND Flash using ATK Tool" to get a valid copy of EBOOT running from NAND Flash.
3. Use the **File** menu of Platform Builder and choose **Project/Solution**.
4. Select XLDR.bin found in the appropriate workspace release directory:

```
WINCE600\OSDesigns\iMX25-3DS-PDK1_7-Mobility\RelDir\Freescale_i_MX25_3DS_ARMV4I_Rel
ease\
XLDR.bin
```

5. Follow the steps in Section 5.4, "Configure Ethernet Connection for Download and Debug" 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 to the 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, press the "y" key to begin programming the Flash.
10. EBOOT programs the image into Flash and provides status using serial debug messages. Once the programming is complete, the following messages are seen:

```
INFO: Update of XLDR completed successfully.
Reboot the device manually...
SpinForever...
Do you want to reset [Y\N]
```

11. Close the Platform Builder workspace for XLDR.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. If you updated EBOOT in NAND Flash, use the instructions provided in Section 5.2.2, "i.MX25 3-Stack Boot Mode Settings" to configure the i.MX25 3-Stack board for direct external boot from NAND Flash. Otherwise follow the steps in Section 5.3.10, "Program/Update XLDR in NAND Flash Using Platform Builder" to update EBOOT in NAND Flash.
14. Reset the target hardware. If you see new EBOOT messages appear on the terminal, XLDR has been properly programmed into NAND Flash.

5.3.11 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 lash following the steps in Section 5.3.3, "Build an EBOOT Image for NAND Flash".
2. If EBOOT is not resident in Flash or the resident EBOOT is not compatible with the current BSP, first follow the steps in Section 5.3.16, "Program XLDR and EBOOT in NAND Flash using ATK Tool" to get a valid copy of EBOOT running from NAND Flash.
3. Use the **File** menu of Platform Builder and choose **Project/Solution**.
4. Select EBOOT.bin found in the workspace release directory:

```
WINCE600\OSDesigns\iMX25-3DS-PDK1_7-Mobility\RelDir\Freescale_i_MX25_3DS_ARMV4I_Rel
ease\
EBOOT.bin
```

5. Follow the steps in Section 5.4, "Configure Ethernet Connection for Download and Debug" 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 to the 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, press the "y" key to begin programming the Flash.
10. EBOOT programs the image into Flash and provides status using serial debug messages. Once the programming is complete, the following messages are seen:

```
INFO: Update of EBOOT/SBOOT completed successfully.
Reboot the device manually...
SpinForever...
Do you want to reset [Y\N]
```

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 5.2.2, "i.MX25 3-Stack Boot Mode Settings" to configure the i.MX25 3-Stack board for direct external boot from NAND Flash.
14. Reset the target hardware. If you see new EBOOT messages appear on the terminal, EBOOT has been properly programmed into NAND Flash.

5.3.12 Program/Update XLDR in SPI Flash Using Platform Builder

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

1. Build XLDR for SPI following the steps in Section 5.3.6, “Build an XLDR Image for SPI Flash”.
2. If EBOOT is not resident in Flash or the resident EBOOT is not compatible with the current BSP, first follow the steps in Section A.3, “Load 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 **Project/Solution**.
4. Select XLDR.bin found in the appropriate workspace release directory:
```
WINCE600\OSDesigns\iMX25-3DS-PDK1_7-Mobility\RelDir\Freescale_i_MX25_3DS_ARMV4I_Rel
ease\
XLDR.bin
```
5. Follow the steps in Section 5.4, “Configure Ethernet Connection for Download and Debug” 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 to the 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, press the “y” key to begin programming the Flash.
10. EBOOT programs the image into Flash and provides status using serial debug messages. Once the programming is complete, the following messages are seen:
```
INFO: Update of XLDR completed successfully.
Reboot the device manually...
SpinForever...
Do you want to reset [Y\N]
```
11. Close the Platform Builder workspace for XLDR.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. If you updated EBOOT in SPI Flash, use the instructions provided in Section 5.2.2, “i.MX25 3-Stack Boot Mode Settings” to configure the i.MX25 3-Stack board for direct external boot from SPI Flash. Otherwise follow the steps in Section 5.3.13, “Program/Update EBOOT in SPI Flash Using Platform Builder” to update EBOOT in SPI Flash.
14. Reset the target hardware. If you see new EBOOT messages appear on the terminal, XLDR has been properly programmed into SPI Flash.

5.3.13 Program/Update EBOOT in SPI Flash Using Platform Builder

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

1. Build EBOOT for SPI Flash following the steps in Section 5.3.5, “Build an EBOOT Image for SPI Flash”.

2. If EBOOT is not resident in Flash or the resident EBOOT is not compatible with the current BSP, first follow the steps in Section A.3, "Load 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 **Project/Solution**.
4. Select EBOOT.bin found in the workspace release directory:

```
WINCE600\OSDesigns\iMX25-3DS-PDK1_7-Mobility\RelDir\Freescale_i_MX25_3DS_ARMV4I_Rel
ease\
EBOOT.bin
```

5. Follow the steps in Section 5.4, "Configure Ethernet Connection for Download and Debug" 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 to the 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, press the "y" key to begin programming the Flash.
10. EBOOT programs the image into Flash and provides status using serial debug messages. Once the programming is complete, the following messages are seen:

```
INFO: Update of EBOOT/SBOOT completed successfully.
Reboot the device manually...
SpinForever...
Do you want to reset [Y\N]
```

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 5.2.2, "i.MX25 3-Stack Boot Mode Settings" to configure the i.MX25 3-Stack board for direct external boot from SPI Flash.
14. Reset the target hardware. If you see new EBOOT messages appear on the terminal, EBOOT has been properly programmed into SPI Flash.

5.3.14 SD/MMC Boot Card with File System Support

To use the SD/MMC card for file system support that is older than spec v2.1 (for SD) and v4.3 (for MMC) and that is already flashed with boot images as described in Section 5.3.8, "Program/Update XLDR in SD/MMC Card Using Platform Builder" and Section 5.2.9, Program/Update EBOOT in SD/MMC Card Using Platform Builder", use the following steps:

1. Reset the 3-Stack board to launch EBOOT on the target.
2. Quickly switch over to the terminal emulation application and wait for the debug message "Press [ENTER] to download now or [SPACE] to cancel." to appear.
3. Press the space bar or enter key to bring up the EBOOT configuration menu.
4. Select "M" to enter the submenu for MMC and SD utilities.

5. Select “C” to create boot and file system partitions on the card. Select “y” when prompted by the warning message. This step does not harm the boot images already flashed on the card. This step only successfully completes on a card that is larger than 64 Mbytes because first 64 Mbytes on the card is saved off for the boot image partition.
6. Remove the card from the device and plug it into an SD card reader on a Windows XP PC.
7. Format the card using WinXP. This step does not harm the boot images already flashed on the card.
8. This card can now be used on the Windows CE system as a file system storage device and also continue to be used as a boot device.

NOTE

Do not format the card on the Windows CE system, as it erases both the boot images as well as the file system partition. Running any kind of CETK Storage Device tests also erases the boot and file system on the card because these tests format the card as needed by the tests. The only way to format the card safely (without erasing the boot images) on the CE device is to do the following: From the Storage Manager, select the **PART01** partition on the card, and select **Properties**. Dismount this partition, then select **Format**. On the next dialog box, be sure to select **FAT32** as the file system from the drop-down menu, and complete the format.

Once the boot card is partitioned to also support file system, it is not recommended to run CETK Storage Device tests until the card is formatted on the device using the Storage Manager (in order to wipe out boot images and file system partition).

5.3.15 Program XLDR and EBOOT in SD/MMC Card Using ATK Tool

Refer to the *i.MX25 Windows Embedded CE 6.0 Release Notes* for the suggested ATK Tool version. Follow these steps to program XLDR and EBOOT in a SD/MMC card using the ATK Tool:

1. Use the instructions provided in Section 5.2.2, “i.MX25 3-Stack Boot Mode Settings” to configure bootstrap mode.
2. Power on the board.
3. Launch the ATK Tool.
4. Select **i.MX25_TO1.1** in **i.MX CPU**.
5. Select **DDR2** in **Device Memory Initial**.
6. Select **Serial Port: COM1** in **Communication Channel**.
7. Click **Next** > **Flash Tool** > **Go**.
8. Select **Program** in **Operation type**.
9. Select **MMC/SD** in **Flash Model**.
10. Set **Operation Settings** as 0x00000400.
11. Specify the XLDR.nb0 in **Image**.
12. Press the **Program** button to program XLDR in SD/MMC card

13. Set **Operation Settings** as 0x00020000.
14. Specify the EBOOT.nb0 in **Image**.
15. Press **Program** button to program EBOOT in SD/MMC card.
16. Exit the ATK Tool.
17. Power off the board.
18. Use the instructions provided in Section 5.2.2, "i.MX25 3-Stack Boot Mode Settings" to configure the SD/MMC card boot mode.
19. Power on the board. The EBOOT menu should be seen on the serial terminal.

5.3.16 Program XLDR and EBOOT in NAND Flash using ATK Tool

Refer to the *i.MX25 Windows Embedded CE 6.0 Release Notes* for the suggested ATK Tool version. Follow these steps to program XLDR and EBOOT in NAND Flash using the ATK Tool:

1. Use the instructions provided in Section 5.2.2, "i.MX25 3-Stack Boot Mode Settings" to configure bootstrap mode.
2. Power on the board.
3. Launch the ATK Tool.
4. Select **i.MX25_TO1.1** in **i.MX CPU**.
5. Select **DDR2** in **Device Memory Initial**.
6. Select **Serial Port: COM1** in **Communication Channel**.
7. Click **Next** > **Flash Tool** > **Go**.
8. Select **Program** and check on **BI Swap** in **Operation type**.
9. Select **NAND** in **Flash Model**.
10. Set **Operation Settings** as 0x00000000.
11. Specify the XLDR.nb0 in **Image**.
12. Press **Program** button to program XLDR in NAND Flash.
13. Set **Operation Settings** as 0x00180000.
14. Specify the EBOOT.nb0 in **Image**.
15. Press **Program** button to program EBOOT in NAND Flash.
16. Exit ATK Tool.
17. Power off the board.
18. Use the instructions provided in Section 5.2.2, "i.MX25 3-Stack Boot Mode Settings" to configure NAND Flash boot mode.
19. Power on the board. The EBOOT menu should be seen on the serial terminal.

NOTE

LBA in **Operation type** must be selected when programing EBOOT and NK images or dumping data to or from the NAND.

5.4 Configure Ethernet Connection for Download and Debug

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 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 starts EBOOT execution. If EBOOT has not been programmed into the target, refer to Section 5.3.15, "Program XLDR and EBOOT in SD/MMC Card Using ATK Tool" or Section 5.3.16, "Program XLDR and EBOOT in NAND Flash using ATK Tool".

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

16. Select **Enable access to desktop files**.
17. Choose **Apply**.
18. Choose **Close**.

Chapter 6
Downloading and Debugging Images

This section describes the procedures for downloading and debugging OS images on the i.MX25 3-Stack 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 Download an OS Image into SDRAM Using EBOOT

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

1. If EBOOT is not resident on the target, follow the procedure in Section 5.3, “EBOOT Installation and Configuration” to program EBOOT into the target.
2. Prepare the target hardware by following the instructions in Section 5.2.1, “i.MX25 3-Stack Board Configuration”. Configure the CPU board for direct external boot from NAND Flash.
3. Open the desired workspace within Platform Builder.
4. Deselect **Project** > **Properties** > **Configuration Properties** > **Build Options** > **Write Run-time Image to Flash Memory**.
5. Within **Build** > **Properties** > **Configuration Properties** > **Environment**, remove the IMGNAND environment variable if it exists.
6. Build the image following the steps provided in Chapter 4, “Building OS Images”.
7. Reset the 3-Stack board to launch EBOOT on the target.
8. If a target device connection has not been created within the Platform Builder, follow the steps in Section 5.4, “Configure Ethernet Connection for Download and Debug” to establish a connection.
9. From the Platform Builder **Target** menu, select **Attach Device** to begin the download.

6.2 Download an OS Image into SD/MMC Card Using EBOOT

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

1. If EBOOT is not resident on the target, follow the procedure in Section 5.3, “EBOOT Installation and Configuration” to program EBOOT into the target.
2. Prepare the target hardware by following the instructions in Section 5.2.1, “i.MX25 3-Stack Board Configuration”. Configure the CPU board for direct external boot from SD/MMC card.
3. Open the desired workspace within Platform Builder.
4. Within **Build** > **Properties** > **Configuration Properties** > **Environment**, use the **New** button to add IMGSDMMC = 1.
5. Build the image following the steps provided in Chapter 4, “Building OS Images”.
6. Reset the 3-Stack board to launch EBOOT on the target.

7. If a target device connection has not been created within Platform Builder, follow the steps in Section 5.4, “Configure Ethernet Connection for Download and Debug” to establish a connection.
8. From the Platform Builder **Target** menu, select **Attach Device** to begin the download.
9. After the download is complete, switch over to the terminal emulation application.
10. At this point EBOOT has downloaded the image temporarily into SDRAM and is ready to begin the Flash programming procedure.
11. EBOOT programs the image into SD/MMC card and provides status using serial debug messages. Once the programming is complete, following messages are seen:

```
INFO: Update of NK completed successfully.
Reboot the device manually...
SpinForever...
Do you want to reset [Y\N]
```

6.3 Run an OS Image from SD/MMC Card Using EBOOT

To execute an OS image from a SD/MMC card that has been previously programmed using the procedure described in Section 6.2, “Download an OS Image into SD/MMC Card Using EBOOT”, follow these steps.

1. Reset the 3-Stack board to launch EBOOT on the target.
2. Quickly switch over to the terminal emulation application and wait for the debug message “Press [ENTER] to download now or [SPACE] to cancel.” to appear.
3. Press the space bar to bring up the EBOOT configuration menu.
4. Continue selecting the **Select Boot Device** option of the EBOOT menu until 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 3-Stack board to launch EBOOT again. EBOOT automatically loads the OS image from the SD/MMC card to RAM and executes it after the specified boot delay.

NOTE

If the 3-Stack board is configured for active KITL, the image waits for a Platform Builder connection before booting the OS image. Since the OS image is already resident in Flash memory, you must reconfigure Platform Builder to jump directly to the image by selecting **Target** > **Connectivity Options** > **Core Service Settings** > **Download Image: Never (jump to image)**.

6.4 Download an OS Image into NAND Flash Using EBOOT

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

1. If EBOOT is not resident on the target, follow the procedure in Section 5.3, “EBOOT Installation and Configuration” to program EBOOT into NAND Flash memory.
2. Prepare the target hardware by following the instructions in Section 5.2.1, “i.MX25 3-Stack Board Configuration”. Configure the CPU board for direct external boot from NAND Flash.
3. Open the desired workspace within Platform Builder.
4. Deselect **Project** > **Properties** > **Configuration Properties** > **Build Options** > **Write Run-time Image to Flash Memory**.
5. Within **Build** > **Properties** > **Configuration Properties** > **Environment**, use the **New** button to add IMGNAND = 1.
6. Build the image following the steps provided in Chapter 4, “Building OS Images”.
7. Reset the 3-Stack board to launch EBOOT on the target.
8. If a target device connection has not been created within the Platform Builder, follow the steps in Section 5.4, “Configure Ethernet Connection for Download and Debug” 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 the 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 programs the image into Flash and provides status using serial debug messages. Once the programming is complete, the following messages are seen:

```
INFO: Update of NK completed successfully.
Reboot the device manually...
SpinForever...
Do you want to reset [Y\N]
```

6.5 Run 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, “Download an OS Image into NAND Flash Using EBOOT”, follow these steps.

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

NOTE

If the 3-Stack board is configured for active KITL, the image waits for a Platform Builder connection before booting the OS image. Since the OS image is already resident in Flash memory, you must reconfigure Platform Builder to jump directly to the image by selecting **Target** > **Connectivity Options** > **Core Service Settings** > **Download Image: Never (jump to image)**.

Chapter 7 BSP Features

7.1 Supported Features and Device Drivers

Details of the drivers and kernel support provided in the i.MX25 3-Stack BSP are described in the *i.MX25 Windows Embedded CE 6.0 Reference Manual*. Refer to this document for more information regarding the requirements, implementation, and testing for each of the i.MX25 3-Stack BSP features.

Appendix A
RealView Toolchain

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

A.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.MX25 Windows Embedded CE 6.0 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.MX25 3-Stack Debugger board.

A.2 RVDEBUG Configuration

The following is the RVDEBUG configuration example for RVI v3.1.

1. Follow the steps in Section A.1, "RVI Configuration" to configure the RVI unit.
2. Install the RealView Developer Suite.
3. Launch RVDEBUG.
4. Select **File** > **Connection** > **Connect to Target**.
5. Expand **RealView Ice**.
6. Right-Click on **RealView Ice** and select **Configure Device Info**.
7. Click on **RealView ICE: (TCP/IP...)** on left window pane.
8. Click **Browse...** button on right window pane.
9. Choose the desired RVI unit and click OK.
10. Click **Connect**.
11. Under **JTAG Clock Speed** select **Adaptive**.

NOTE

Currently, the Auto Configure Scan Chain feature does not work correctly with the i.MX25 3-Stack. Use the **Add Device** and **Device Properties** buttons to manually create the necessary scan chain entries:

- **Add Device** > **Custom Device** > **UNKNOWN** > **IR Length = 5**
- **Add Device** > **Custom Device** > **UNKNOWN** > **IR Length = 4**

- **Add Device** > **ARM9** > **ARM926EJ-S**
- **Add Device** > **ARM** > **ETMBUF**

The last step of adding **ETMBUF** is for RVI v3.1. For RVI v3.2 add **ETB**.

12. Select **File** > **Save**.
13. Select **File** > **Exit**.
14. In the **Connection Control** window, expand **RealView ICE**
15. For the i.MX25 3-Stack, click the **ARM926EJ-S** device to establish a connection.

NOTE

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

A.3 Load 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 5.2.1, “i.MX25 3-Stack Board Configuration”. Refer to Section 5.2.2, “i.MX25 3-Stack Boot Mode Settings” to set the board to Bootstrap boot mode.
3. Configure the RealView toolchain by following the instructions in Section A.1, “RVI Configuration”.
4. Locate the RVDEBUG initialization script in the following BSP directory:
   `WINCE600\SUPPORT_PDK1_7\TOOL\iMX25-3DS\RVD\`
5. Edit the RVD script (RVD_MX25_MDDR.inc for the i.MX25 3-Stack) 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 press **F5** to start the EBOOT execution.
8. You should see EBOOT serial debug messages on your desktop terminal emulation application.