i.MX51 EVK
Windows Embedded CE 6.0

User's Guide

Document Number: 924-76369
Rev. WCE600_MX51_ER_1104
05/2011

freescale™
semiconductor

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or
+1-480-768-2130
www.freescale.com/support

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)
www.freescale.com/support

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 China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 010 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
+1-800 441-2447 or
+1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Contents

About This Book

Chapter 1
BSP Installation

Chapter 2
BSP Contents and Organization

Chapter 3
Configuring OS Images

Chapter 4
Building OS Images

Chapter 5
Preparing for Downloading and Debugging

Chapter 6
Downloading and Debugging Images

Chapter 7
BSP Features

Appendix A
RealView Toolchain Configuration

About This Book

This document is a user's guide for the Freescale i.MX51 EVK 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.

Suggested Reading

Additional information regarding Microsoft Windows CE and the i.MX51 can be found in these documents:

- i.MX51 EVK Platform System HW User Guide
- Windows Embedded CE 6.0 BSP for i.MX51 EVK Reference Manual
- 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

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.

API	application programming interface
BSP	board support package
CSP	chip support package
DHCP	dynamic host configuration protocol
EBOOT	Ethernet bootloader
EVB	platform evaluation board
EVK	platform evaluation kit
FAL	flash abstraction layer
GDI	graphics display interface
ICE	in-circuit emulator
IDE	integrated development environment
IST	interrupt service thread
IPU	image processing unit
KITL	kernel independent transport layer
LVDS	low-voltage differential signaling
MAC	media access control
OAL	OEM adaptation layer
OEM	original equipment manufacturer
OS	operating system
PQOAL	production quality OEM adaptation layer
RVDEBUG	RealView debugger
RVI	RealView ICE
SDC	synchronous display controller
SDRAM	synchronous dynamic random access memory
SoC	system on a chip

References

The following documents were referenced to produce this document.

- *Windows Embedded CE 6.0 Online Help*
- *i.MX51 Applications Processor Reference Manual*
- *i.MX51 EVK Windows Embedded CE 6.0 Release Notes*
- *i.MX51 EVK Windows Embedded CE 6.0 Reference Manual*

Chapter 1
BSP Installation

The BSP is distributed as a single Microsoft® Installer (.msi) file that includes support for the i.MX51 EVK platform. See the *i.MX51 EVK Windows Embedded CE 6.0 Release Notes* for additional instructions and information before installing and using this BSP.

1.1 Installing Windows Embedded Tools and i.MX51 EVK BSP

To create the complete i.MX51 EVK development environment, install the Microsoft Windows Embedded CE 6.0 development tools and the Freescale i.MX51 EVK BSP as follows:

1. Install Visual Studio 2005 and the Windows Embedded CE 6.0 Platform Builder plugin from the installation discs. See the *i.MX51 EVK BSP Release Notes* on the Visual Studio and Platform Builder installation discs for installation instructions.

   NOTE

   When the platform builder installer asks for the operating system version, select the ARMV4I option. This option needs to be installed on the local hard drive. The other operating systems are not required.

   See the *i.MX51 EVK BSP Release Notes* for information about any additional Microsoft-supplied QFEs or Service Packs that also need to be installed.

2. If an earlier version of the Freescale i.MX51 EVK BSP has been previously installed, remove any existing BSP files by following the instructions in Section 1.2, "Uninstalling the Freescale i.MX51 EVK BSP."
3. Download the Freescale i.MX51 EVK BSP and install the contents in the existing WINCE600 top-level folder. The MSI installer automatically creates the necessary subfolders so all of the files are installed in the correct location.

1.2 Uninstalling the Freescale i.MX51 EVK BSP

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

NOTE

Uninstalling the BSP removes all files that were installed with the MSI installer. If any changes were made to these files, they are removed. Be sure to save any valuable modified files 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 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 the previous step because they contain object files or other generated files that were not part of the original BSP installation):

```
WINCE600\OSDesigns\iMX51-EVK-SmallFootprint
WINCE600\OSDesigns\iMX51-EVK-Mobility
WINCE600\PLATFORM\COMMON\SRC\SOC\COMMON_FSL_V2
WINCE600\PLATFORM\COMMON\SRC\SOC\MX51_FSL_V2
WINCE600\PLATFORM\iMX51-EVK
WINCE600\SUPPORT
```

4. Manually remove the residual BSP library files that were created after the BSP was installed. To find the libraries, use Windows Explorer with the search name `*FSL_V2*` for 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:

1. In the Platform Builder, select **File** > **Open** > **Project/Solution...**
2. Select the i.MX51 EVK BSP sample workspace by loading the following files:

```
WINCE600\OSDesigns\iMX51-EVK-Mobility\iMX51-EVK-Mobility.sln
WINCE600\OSDesigns\iMX51-EVK-SmallFootprint\iMX51-EVK-SmallFootprint.sln
```

The process of loading this solution also automatically loads the associated i.MX51 EVK BSP Catalog. Simply 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 i.MX51 EVK-based platforms. A BSP contains the following elements:

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

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

The i.MX51 EVK 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` is used to store shared OAL and driver components. This `COMMON_FSL_V2` 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 chipset-level code that can be leveraged to develop platforms based on the i.MX51 EVK SoC and the PQOAL components customized for the i.MX51. 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, the source code in the SoC directory utilizes hardware abstraction routine 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, see the topic **Production-Quality Drivers** in the *Windows Embedded CE 6.0 Help*.

The following directories contain the SoC driver source code for the i.MX51 EVK:

```
WINCE600\PLATFORM\COMMON\SRC\SOC\COMMON_FSL_V2
WINCE600\PLATFORM\COMMON\SRC\SOC\MX51_FSL_V2
```

The SoC code in the `COMMON_FSL_V2` directory is reusable across all Freescale ARM-based SoCs. The SoC driver code in the `MX51_FSL_V2` directory is reusable across all platforms based on the i.MX51 EVK.

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, see the topic **Production-Quality OAL** in the *Windows Embedded CE 6.0 Help*.

Where possible, the Freescale BSP leverages the PQOAL architecture and components provided by Microsoft to reduce the OAL code that needs to be modified and maintained by the OEM. In addition, PQOAL components customized for the i.MX51 EVK are available in the following directories:

```
WINCE600\PLATFORM\COMMON\SRC\SOC\COMMON_FSL_V2\OAL
WINCE600\PLATFORM\COMMON\SRC\SOC\MX51_FSL_V2\OAL
```

The PQOAL code in the `COMMON_FSL_V2\OAL` directory is reusable across all Freescale ARM-based SoCs. The PQOAL code in the `MX51_FSL_V2\OAL` directory is reusable across all platforms based on the i.MX51 EVK.

2.2 i.MX51 EVK Platform Files

The i.MX51 EVK BSP provides direct support for the interfaces and peripherals found on the i.MX51 EVK board. All of the driver and OAL content that is specific to the underlying hardware platform is located in the following directory:

```
WINCE600\PLATFORM\iMX51-EVK
```

The i.MX51 EVK platform directory implements the hardware abstraction routines invoked by driver code in the Freescale SoC directory. Additionally, 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 within the following directory:

```
WINCE600\OSDesigns\iMX51-EVK-Mobility
```

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

```
WINCE600\OSDesigns\iMX51-EVK-SmallFootprint
```

The solution was created using the Platform Builder **New** > **Project** feature utilizing the Custom Device Design Template. For more information about creating an OS solution, see the topic **Creating an OS Design with the Windows Embedded CE OS Design Wizard** 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
```

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

Chapter 3
Configuring OS Images

Use the Platform Builder to select one of the two default build configurations provided in the BSP sample solution. These configurations control the type of OS image (debug or release) that is generated by the Platform Builder tools. In addition, the build configuration encapsulates all of the platform environment variables and custom build instructions that are used during OS image creation. This section describes the configuration options available for the i.MX51 EVK BSP.

NOTE

The sample solution provided within the i.MX51 EVK BSP is properly configured to generate a default image targeted for the i.MX51 EVK hardware. It is not necessary to adjust any of the image configuration settings prior to building the BSP.

3.1 Image Build Type

The type of OS image generated by Platform Builder can be controlled using the **Build** > **Configuration Manager...** menu item and choosing the appropriate **Active Solution Configuration** item from the drop-down list. The sample workspace provided with the i.MX51 EVK BSP provides the following two image build types:

- **Freescale_i_MX51_EVK_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_MX51_EVK_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 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, see the topic **Build Configurations** 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
- IMGXXX

3.2.1 BSP_NOXXX Variables

The i.MX51 EVK 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 (for example SYSGEN_SMARTCARD) triggers the automatic selection of certain BSP drivers (for example SIM). The drivers that have a SYSGEN dependency 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\iMX51-EVK\iMX51-EVK.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 to 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	Setting
BSP_NOAUDIO	Excludes support for SSI Audio driver	BSP_NOAUDIO=1 Excludes Audio driver support from the OS image despite the SYSGEN_AUDIO dependency.
BSP_NOBATTERY	Excludes support for SSI Battery driver	BSP_NOBATTERY=1 Excludes Battery driver support from the OS image despite the SYSGEN_BATTERY dependency.
BSP_NOBLUETOOTH	Excludes support for Bluetooth® driver	BSP_NOBLUETOOTH=1 Excludes Bluetooth driver support from the OS image despite the SYSGEN_BTH dependency.
BSP_NOCSPDDK	Excludes 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_NOCSPI	Excludes support for eCSPI driver	BSP_NOCSPI=1 Excludes eCSPI driver support from the OS image despite the SYSGEN_DEVICE dependency.
BSP_NODISPLAY	Excludes support for Display driver	BSP_NODISPLAY=1 Excludes Display driver support from the OS image despite the SYSGEN_DISPLAY dependency.
BSP_NOETHER	Excludes support for ethernet driver	BSP_NOETHER=1 Excludes Ethernet driver support from the OS image despite the SYSGEN_NDIS dependency.

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

Variable Name	Description	Setting
BSP_NOKEYPAD	Excludes support for Keypad driver	BSP_NOKEYPAD=1 Excludes Keypad driver support from the OS image despite the SYSGEN_MININPUT dependency.
BSP_NONLED	Excludes support for NLED driver	BSP_NONLED=1 Excludes NLED driver support from the OS image despite the SYSGEN_NLED dependency.
BSP_NOPMIC	Excludes support for PMIC driver	BSP_NOPMIC=1 Excludes PMIC driver support from the OS image despite the SYSGEN_DEVICE dependency.
BSP_NOSDHC	Excludes support for eSDHC driver	BSP_NOSDHC=1 Excludes eSDHC driver support from the OS image despite the SYSGEN_SDBUS dependency.
BSP_NOSIM	Excludes support for SIM driver	BSP_NOSIM=1 Excludes SIM driver support from the OS image despite the SYSGEN_SMARTCARD dependency.
BSP_NOTOUCH	Excludes support for Touch driver	BSP_NOTOUCH=1 Excludes Touch driver support from the OS image despite the SYSGEN_TOUCH dependency.
BSP_NOUSB	Excludes 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 is either BSP_NOXXX =, which is set in the batch file by default, or no definition of the BSP_NOXXX variable at all. Thus, the driver is included by SYSGEN selection.

Although the drivers listed above have direct SYSGEN dependency, some of them may not be automatically selected by SYSGEN dependency because of the following possibilities:

1. The driver may have multiple subordinate selections; for example, when there are two selections (eSDHC1 and eSDHC2) for SD Host Controllers.
2. The driver may depend on another driver; for example, when Audio depends on the I2C2 driver.
3. The driver may conflict with other drivers.

3.2.2 BSP_XXX Variables

The i.MX51 EVK BSP uses the BSP_XXX variables defined in the catalog to configure the drivers that can not be automatically be selected by SYSGEN dependency. In these cases, users can use the Platform Builder Catalog UI to select or deselect drivers by clicking the catalog items. The *i.MX51 EVK Windows Embedded CE 6.0 Reference Manual* describes the BSP_XXX variables for individual drivers.

NOTE

Users might not always be able to successfully select the desired drivers in the catalog graphical user interface. They may see a small red **x** for some cases, which means there is some dependency or conflict involved in the selection. In this case, users should right-click the catalog item and see the details in **Reasons for Exclusion of Item**. To successfully make the selection, use the information shown to select the required SYSGEN components or drivers and to deselect conflicting components.

When creating a custom workspace using the Platform Builder wizard, manually select the desired drivers that may not be automatically selected by SYSGEN dependency using the BSP catalog items under **Third party** > **BSP** > **Freescale i.MX51 EVK: ARMV4I**.

Make sure BSP_SI_VER variable is set to 2.0 in the **Environment** dialog, launched from the Platform Builder menu **Project** > **Properties** > **Configuration Properties** > **Environment**.

3.2.3 IMG_XXX Variables

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

1. Open the sample solution for the i.MX51 EVK 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.MX51 EVK BSP.

Table 3-2. IMG_XXX Variables

Variable Name	Description	Setting
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
IMGRAM256	Configures the run-time image for 256 Mbytes of RAM	IMGRAM256 = 1 Configures the run-time image for 256 Mbytes of RAM Remove to configure the run-time image for 512 Mbytes of RAM
IMGRAM128	Configures the run-time image for 128 Mbytes of RAM	IMGRAM128 = 1 Configures the run-time image for 128 Mbytes of RAM Remove to configure the run-time image for 512 Mbytes of RAM
IMGSMALLNK	Configures the NK binary file to be 47MB	IMGSMALLNK = 1 Configures the NK binary file to be 47MB. In this case you should make sure the NK size will never exceed this size. Remove to configure the NK binary file to be 94MB

3.3 Catalog Environment Variables

The i.MX51 EVK BSP utilizes the Platform Builder Windows CE Catalog to configure BSP components in the OS design. The *i.MX51 EVK Windows Embedded CE 6.0 Reference Manual* describes the environment variables associated with each of the BSP features exposed in the i.MX51 EVK BSP Catalog.

Chapter 4
Building OS Images

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

4.1 Building the Freescale SoC Libraries

The Freescale SoC libraries that support the i.MX51 EVK are generated during the **Build** > **Advanced Build Commands** > **Sysgen** or **Build** > **Advanced Build Commands** > **Build Current BSP and Subprojects** build procedure. Windows CE ships with pre-built SoC libraries for various ARM processors, but the libraries for the i.MX51 EVK must be built from the sources since they are not included with the standard Microsoft distribution.

The sample OS design solution provided is already preconfigured to build the required Freescale SoC. That is, the sample i.MX51 EVK OS design solution automatically build the i.MX51 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 a Sysgen operation has never been performed. If a Sysgen has already been performed, it is only necessary to **Build Current BSP and Subprojects**, which rebuilds the Freescale SoC libraries.

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 Building Run-Time Images

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

4.2.1 Building the BSP for the First Time

1. Open the sample solution.
2. Select the desired build type discussed in Section 3.1, “Image Build Type.”
3. Using the **Build** > **Global Build Settings** menu, configure the build options as follows:
   - 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 OS image files are placed in the following directory depending on the OS design being used:

```
WINCE600\OSDesigns\iMX51-EVK-Mobility\RelDir\Freescale_i_MX51_EVK_ARMV4I_Release
```

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

```
WINCE600\OSDesigns\iMX51-EVK-Mobility\RelDir\Freescale_i_MX51_EVK_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 all the Freescale BSP components can be accomplished as follows:

1. Select **Copy Files to Release Directory After Build**.
2. Deselect **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, an incremental build of the Freescale BSP components can be performed 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 Downloading and Debugging

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. Additionally, 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.MX51 EVK BSP has been configured to use internal UART1 routed to the UARTC connector for serial debug messages. UARTC on the EVK board is the DB-9 connector next to the DVI port. Connect a standard serial cable between UARTC of the i.MX51 EVK board and the development workstation.

5.2 Board Configuration

5.2.1 i.MX51 EVK Board Configuration

This section describes the switch and jumper configuration required for proper operation of the BSP on the i.MX51 EVK board. For specific BSP features, the EVK board may need to be reconfigured to support the necessary interface. See the *i.MX51 EVK Windows Embedded CE 6.0 Reference Manual* for board configurations required by some BSP features.

5.2.2 i.MX51 EVK Boot Mode Settings

Table 5-1 shows the configuration for the i.MX51 EVK Boot Mode.

Table 5-1. CPU Boot Mode Settings

Boot Mode	Jumper/Switch Setting	Configuration
SPI Flash Boot	S1 setting	On: 3, 4, 5, 7, 8 Off: others
SD/MMC Boot	S1 setting	On: 7, 8 Off: others
Bootstrap	S1 setting	On: 1, 2, 7, 8, 10 Off: others

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. Furthermore, 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.

NOTE

If this is the first time to run the EBOOT on the i.MX51 EVK, the following message may appear during the boot:

```
INFO: Initialized SD Card
SPIFMD_Init: Manufacturer ID = 0x1f270100
Warning: Unsupported Page Size of 528 Bytes!
Reprogramming the Page Size is in progress.
Do NOT Power Down the board!
......
Reprogrammed the Page Size to be 512 Bytes!
Please Power Cycle the Board !
SpinForever...
```

After this message appears, Power Cycle (power down then power up) the i.MX51 EVK.

EBOOT that is stored in a SD/MMC card uses a reserved SD/MMC region to store the boot configuration. EBOOT that is resident in a SPI Flash uses a reserved SPI Flash region to store the boot configuration. Update the boot configuration as required when switching between the SD/MMC and SPI Flash versions of the bootloader.

5.3.1 Building an EBOOT Image for SD/MMC Card

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, the libraries needed by EBOOT are created.
2. Specify the targeted memory for the EBOOT image as a SD/MMC card. This is achieved by ensuring that the IMGSDMMC environment variable is set 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 **iMX51-EVK-Mobility** > **F:/WINCE600** > **PLATFORM** > **iMX51-EVK** > **src** > **BOOTLOADER**.
5. Right-click on the **EBOOT** folder again and select **Rebuild**.
6. The EBOOT binary image is created in the workspace release directory.

5.3.2 Building an XLDR Image for SD/MMC Card

Build XLDR for SD/MMC card 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, the libraries needed by XLDR are created.
2. Specify the targeted memory for the XLDR image as a SD/MMC card. This is achieved by ensuring that the IMGSDMMC environment variable is set within **Project** > **Properties** > **Configuration Properties** > **Environment**. This causes the XLDR image to be linked for a SD/MMC card.
3. Select the **Solution Explorer** tab of the Platform Builder window.
4. Expand **iMX51-EVK-Mobility** > **F:/WINCE600** > **PLATFORM** > **iMX51-EVK** > **src** > **BOOTLOADER** > **XLDR.**
5. Right-click on the **SD** folder again and select **Rebuild**.
6. The XLDR binary image is created in the workspace release directory.

5.3.3 Building an EBOOT Image for SPI Flash

Build EBOOT for SPI 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, the libraries needed by EBOOT are created.
2. Specify the targeted memory for the EBOOT image as SPI Flash. This is achieved by removing the environment variable IMGSDMMC within **Project** > **Properties** > **Configuration Properties** > **Environment**. This causes the EBOOT image to be linked for SPI Flash (that is, the default targeted memory for the EBOOT image is SPI Flash).
3. Select the **Solution Explorer** tab of the Platform Builder window.
4. Expand **iMX51-EVK-Mobility** > **F:/WINCE600** > **PLATFORM** > **iMX51-EVK** > **src** > **BOOTLOADER**.

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

5.3.4 Building an XLDR Image for SPI Flash

Build XLDR for SPI 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, the libraries needed by XLDR are created.
2. Specify the targeted memory for the XLDR image as SPI Flash. This is achieved by removing the environment variable IMGSDMMC within **Project** > **Properties** > **Configuration Properties** > **Environment**. This causes the XLDR image to be linked for SPI Flash (that is, the default targeted memory for the XLDR image is SPI Flash).
3. Select the **Solution Explorer** tab of the Platform Builder window.
4. Expand **iMX51-EVK-Mobility** > **F:/WINCE600** > **PLATFORM** > **iMX51-EVK** > **src** > **BOOTLOADER** > **XLDR.**
5. Right-click on the **CSPI** folder again and select **Rebuild**.
6. The XLDR binary image is created in the workspace release directory.

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

See the *i.MX51 EVK BSP 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.MX51 EVK Boot Mode Settings," to configure bootstrap mode.
2. Power on the board.
3. Launch the ATK Tool.
4. Select **i.MX51_TO2** in **i.MX CPU**.
5. Select **DDR** in **Device Memory Initial**.
6. Select **Serial Port: COM1** in **Communication Channel**. USB may be selected as well. See the *i.MX Advanced Toolkit Standard 1.68 User's Guide* to configure the USB connection.
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 the **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.MX51 EVK 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.

5.3.6 Program XLDR and EBOOT in SD/MMC Card Using Manufacture Tool

See the *i.MX51 EVK BSP Release Notes* for the suggested Manufacture Tool version. Refer to Manufacture Tool document to program XLDR and EBOOT in a SD/MMC card using the Manufacture Tool.

5.3.7 Initialize EBOOT Network Configuration

To initialize EBOOT network configuration, follow these steps:

1. Follow the steps in Section 5.3.5, "Program XLDR and EBOOT in SD/MMC Card Using ATK Tool," to execute 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.

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 and 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 needed, use the **IP Address** and **Subnet Mask** menu options.
7. After the desired network configuration appears in the EBOOT menu, select the **Save configuration** menu option to save the configuration to nonvolatile memory.

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

EBOOT running from SDRAM can be used to program or 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, "Building 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, follow the steps in Section A.3, "Loading EBOOT to SDRAM Using RVDEBUG," to obtain a valid copy of EBOOT running from RAM.

3. From the **File** menu of the Platform Builder choose **Open Project/Solution**.
4. Select XLDR.bin found in the appropriate workspace release directory:

```
WINCE600\OSDesigns\iMX51-EVK-Mobility\RelDir\Freescale_i_MX51_EVK_ARMV4I_Release\XLDR.bin
```

5. Follow the steps in Section 5.4, "Configuring Ethernet Connection for Downloading and Debugging," to establish an Ethernet connection between the target and 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 provide status using serial debug messages. After the programming is complete, the following messages appears:

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

11. Close the Platform Builder workspace for XLDR.bin. It is not necessary to save any workspace changes.
12. If EBOOT was downloaded to SDRAM using RVDEBUG, disconnect from the RVDEBUG by selecting **File** > **Connection** > **Disconnect**.
13. Use the instructions provided in Section 5.2.2, "i.MX51 EVK Boot Mode Settings," to configure the EVK board for boot from SD/MMC card if EBOOT was updated in the SD/MMC card. Otherwise, follow the steps in Section 5.3.9, "Program/Update EBOOT in SD/MMC Card Using Platform Builder," to update EBOOT in SD/MMC card and skip step 14.
14. Reset the target hardware. If new EBOOT messages appear on the terminal, XLDR has been properly programmed into the 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 or update the EBOOT image resident in a 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, "Building 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, follow the steps in Section A.3, "Loading EBOOT to SDRAM Using RVDEBUG," to get a valid copy of EBOOT running from RAM.
3. From the **File** menu of the Platform Builder choose **Open Workspace**.
4. Select EBOOT.bin found in the appropriate workspace release directory:

```
WINCE600\OSDesigns\iMX51-EVK-Mobility\RelDir\Freescale_i_MX51_EVK_ARMV4I_Release\EBOOT.
bin
```

5. Follow the steps in Section 5.4, "Configuring Ethernet Connection for Downloading and Debugging," to establish an Ethernet connection between the target and 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. After the programming is complete, the following messages appears:

```
INFO: Update to 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 EBOOT was downloaded to SDRAM using RVDEBUG, disconnect from the RVDEBUG by selecting **File** > **Connection** > **Disconnect**.
13. Use the instructions provided in Section 5.2.2, "i.MX51 EVK Boot Mode Settings," to configure the EVK board for boot from the SD/MMC card.
14. Reset the target hardware. If new EBOOT messages appear on the terminal, EBOOT has been properly programmed into SD/MMC card.

5.3.10 Using cfimager Utility to Flash SD/MMC Card

cfimager.exe is a host PC tool used to flash boot images (xldr.nb0, eboot.nb0, nk.nb0) and create a FAT partition on SD/MMC cards for i.MX51. Users can avoid programming the card on the device. The tool for i.MX51 can only flash *.nb0 files. A card reader on the PC is required. The utility, accompanied by a readme file, is provided under `<%WINCEROOT%>\SUPPORT\TOOL\COMMON\CFIMAGER`. Any data on the card may be lost by flashing it with cfimager. Follow these steps to flash the card:

1. Build the XLDR for SD. See Section 5.3.2, "Building an XLDR Image for SD/MMC Card," section for instructions.
2. To flash the XLDR, use the following command in a build window:

```
cfimager -f xldr.nb0 -d <card reader drive letter without colon> -imx51
```

3. To flash the bootloader, use the following command:

```
cfimager -f eboot.nb0 -d <card reader drive letter without colon> -imx51
```

4. To flash the OS image, use the following command:

```
cfimager -f nk.nb0 -d <card reader drive letter without colon> -imx51
```

   Flashing a debug nk.nb0 image does not work because it is too big.
5. To add the FAT partition, add `-a` option to any of the above command lines.
   When creating a FAT partition, any previous data in any existing FAT partition is erased.

5.3.11 Program/Update XLDR in SPI Flash Using Platform Builder

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

1. Build XLDR for SPI Flash following the steps in Section 5.3.4, "Building 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, follow the steps in Section A.3, "Loading EBOOT to SDRAM Using RVDEBUG," to get a valid copy of EBOOT running from RAM.
3. From the **File** menu of the Platform Builder choose **Open Workspace**.
4. Select XLDR.bin found in the appropriate workspace release directory:

   ```
   WINCE600\OSDesigns\iMX51-EVK-Mobility\RelDir\Freescale_i_MX51_EVK_ARMV4I_Release\XLDR.bin
   ```

5. Follow the steps in Section 5.4, "Configuring Ethernet Connection for Downloading and Debugging," to establish an Ethernet connection between the target and Platform Builder.
6. From the **Target** menu, select **Attach Device**.
7. After the download is complete, from the **Target** menu, select **Detach Device**.
8. Switch over 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 provide status using serial debug messages. After the programming is complete, the following messages appear:

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

11. Close the Platform Builder workspace for XLDR.bin. It is not necessary to save any workspace changes.
12. If EBOOT was downloaded to SDRAM using RVDEBUG, disconnect from the RVDEBUG by selecting **File** > **Connection** > **Disconnect**.
13. Use the instructions provided in Section 5.2.2, "i.MX51 EVK Boot Mode Settings," to configure the EVK board for direct external boot from SPI Flash if EBOOT was updated in the SPI Flash. Otherwise, follow the steps in Section 5.3.12, "Program/Update EBOOT in SPI Flash Using Platform Builder," to update EBOOT in the SPI Flash.
14. Reset the target hardware. If new EBOOT messages appear on the terminal, XLDR has been properly programmed into SPI Flash.

5.3.12 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 5.3.3, "Building 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, "Loading EBOOT to SDRAM Using RVDEBUG," to get a valid copy of EBOOT running from RAM.
3. Use the **File** menu of Platform Builder and choose **Open Workspace**.
4. Select EBOOT.bin found in the workspace release directory:

```
WINCE600\OSDesigns\iMX51-EVK-Mobility\RelDir\Freescale_i_MX51_EVK_ARMV4I_Release\EBOOT.bin
```

5. Follow the steps in Section 5.4, "Configuring Ethernet Connection for Downloading and Debugging," to establish an Ethernet connection between the target and Platform Builder.
6. From the **Target** menu, select **Attach Device**.
7. After the download is complete, from the **Target** menu, select **Detach Device**.
8. Switch over 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. After the programming is complete, the following messages appear:

```
INFO: Update to 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 EBOOT was downloaded to SDRAM using RVDEBUG, disconnect from the RVDEBUG by selecting **File** > **Connection** > **Disconnect**.
13. Use the instructions provided in Section 5.2.2, "i.MX51 EVK Boot Mode Settings," to configure the i.MX51 EVK board for direct external boot from SPI Flash.
14. Reset the target hardware. If new EBOOT messages appear on the terminal, EBOOT has been properly programmed into SPI Flash.

5.3.13 Using SD/MMC Boot Card for File System Support

To use a SD/MMC card 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 5.3.9, "Program/Update EBOOT in SD/MMC Card Using Platform Builder," and which is older than specification versions 2.1 (for SD) and 4.3 (for MMC), follow these steps for storing file system data:

1. Reset the EVK 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 completes successfully only on a card larger than 64 Mbytes, because first 64 Mbytes on the card is saved 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 Windows XP. This step does not harm the boot images already flashed on the card.

   This card can now be used on the Windows CE system as a file system storage device as well as continue to be used as a boot device.

NOTE

Do not format the card on the Windows CE system, as this erases the boot images as well as the file system partition. Running any kind of CETK Storage Device tests erases the boot and file system on the card, because they format the card as needed by the tests. The only way to format the card safely (that is, 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. 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 the file system, it is not recommended to run CETK Storage Device tests until the card is formatted on the device using the Storage Manager (to wipe out boot images and file system partition).

5.3.14 Using eSD v2.1 and eMMC v4.3 as Boot Devices

On eSD v2.1 and eMMC v4.3 devices, multiple physical partitions can be created to avoid having to write an MBR as described in Section 5.3.13, “Using SD/MMC Boot Card for File System Support.” One partition can be used to store the boot images, while the other partition can be used to store the file system. EBOOT provides menu options to create a 64 Mbyte boot partition. Assuming the eSD or eMMC device is plugged into the SD slot, run EBOOT (either using JTAG or from SPI Flash), and follow these steps to prepare the device for booting from the boot partition:

1. Select **M** from the EBOOT menu to bring up the MMC and SD Utilities submenu.
2. For eMMC select **B**, or for eSD select **D**, to change the boot partition size. If the previous value was 0 Kbytes, then a 65,536 Kbyte (64 Mbyte) boot partition is created. If the previous value was 65,536 Kbytes, then the boot partition is removed (0 Kbytes). Data on all partitions is subject to being lost when either of these menu options is selected.

NOTE

Be sure to have a non-zero boot partition size before continuing.

3. For eMMC select **A** to enable boot partition 1. Select **A** again if boot partition 2 is to be enabled. Selecting **A** a third time disables both boot partitions. Toggle between these three options by repeatedly selecting **A**.

4. For eSD, if the boot partition is disabled, then select **S** to enable it. If **S** is selected again, then the boot partition is disabled. There is only one boot partition on an eSD device.

NOTE

Be sure to enable the boot partition before continuing. Enabling the boot partition means that all subsequent reads, writes, and erasures (including flashing of boot images) happens only to the boot partition. If the boot partition is disabled, then all reads, writes, and erasures (including flashing of boot images) affect only the user (or file system) partition. For eSD version 2.1, ROM code always tries to boot from the boot partition if the size is non-zero. Therefore, boot images flashed to the user partition do not work unless boot partition size is 0 Kbytes prior to flashing. For eMMC v4.3, when the boot partition is disabled, it is still possible to boot from the user partition even when the size is non-zero (provided the boot images were flashed to the user partition), unless the **A** menu option shows **Boot part 1, r/w usr part** or **Boot part 2, r/w usr part** prior to flashing the boot images.

5. Select **R** to return to main menu, and follow the instructions in Section 5.3.8, "Program/Update XLDR in SD/MMC Card Using Platform Builder," and Section 5.3.9, "Program/Update EBOOT in SD/MMC Card Using Platform Builder," to flash the eMMC or eSD device with boot images. The boot partition enable/disable and size settings persist through a warm or cold reset of the system, except in the case where eSD boot partition size is non-zero and boot partition is disabled. In this case, the boot partition is enabled on reset.

NOTE

Only eMMC v4.3 from Samsung is currently supported on this platform. It uses a custom command to change boot partition size. This command does not work on eMMC devices from other manufacturers.

5.4 Configuring Ethernet Connection for Downloading and Debugging

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

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

NOTE

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

4. In the **Download** box, choose **Ethernet** as the download service.

5. Launch EBOOT on the target. After EBOOT initialization completes, BOOTME messages begin to 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 is being programmed into the target, see Section A.3, "Loading EBOOT to SDRAM Using RVDEBUG," for steps to launch EBOOT.

6. To the right of the **Download** box, choose **Settings**. The device name of the target should appear in the **Active Devices** box.
7. Select the 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 the run-time image includes support for the kernel debugger stub, KdStub, from the **Debugger** box, choose **KdStub**. If the run-time image does not include support for a debugger, from the **Debugger** box, choose **None**.
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, under **Download Image**, choose **Always**.
14. Select **Enable KITL on device boot**.
15. Select **Clear memory on soft reset**.
16. Select **Enable access to desktop files**.
17. Choose **Apply**.
18. Choose **Close**.

Chapter 6
Downloading and Debugging Images

This section describes the procedures for downloading and debugging OS images on the i.MX51 EVK board. It is assumed that the steps to build an OS image have been followed and that development hardware has been configured prior to attempting a download and debug session.

6.1 Downloading OS Image to 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.MX51 EVK Board Configuration." Configure the EVK board for boot from SPI Flash or SD/MMC card.
3. Open the desired workspace within the 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 IMGSDMMC environment variable if it exists.
6. Build the image following the steps provided in Chapter 4, "Building OS Images."
7. Reset the EVK 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, "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 OS Image to 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.MX51 EVK Board Configuration." Configure the EVK board for boot from SD/MMC card.
3. Open the desired workspace within the Platform Builder.
4. Within **Build** > **Properties** > **Configuration Properties** > **Environment**, set the IMGSDMMC environment variable if it does not exist.
5. Build the image following the steps provided in Chapter 4, "Building OS Images."
6. Reset the EVK board to launch EBOOT on the target.
7. If a target device connection has not been created within the Platform Builder, follow the steps in Section 5.4, "Configuring Ethernet Connection for Downloading and Debugging," 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 to a SD/MMC card and provides status using serial debug messages. After the programming is complete, the following messages appear:

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

6.3 Running 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, "Downloading OS Image to SD/MMC Card Using EBOOT," follow these steps:

1. Reset the EVK 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 EVK board to launch EBOOT again. EBOOT automatically loads the OS image from SD/MMC to RAM and executes it after the specified boot delay.

NOTE

If the EVK board is configured for active KITL, the image waits for a Platform Builder connection before booting the OS image. Because the OS image is already resident in Flash memory, it is necessary to reconfigure the 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 the kernel support provided in the i.MX51 EVK BSP are described in the *Windows Embedded CE 6.0 BSP for i.MX51 EVK Reference Manual*. This document contains detailed information regarding the requirements, implementation, and testing for each of the i.MX51 EVK BSP features.

Appendix A
RealView Toolchain Configuration

The RealView ICE (RVI) interface is used in conjunction with the RealView Debugger (RVDEBUG) to enable programming of EBOOT into Flash memory, as well as offer hardware debug capability that would otherwise not be possible in the Platform Builder. This section describes the configuration required to prepare the RealView tools for connection with the target.

A.1 RVI Configuration

The following steps configure the RVI:

1. Install the utility software provided with the RVI unit.
2. Use **RVI Config IP** to configure the RVI unit on the network. See 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. See the *i.MX51 EVK BSP Release Notes* for the suggested firmware version. See 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 the supplied cable to connect the RVI unit to the i.MX51 EVK board.

A.2 RVDEBUG Configuration

The following steps configure the RVDEBUG:

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.MX51 EVK. 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** > **Registered Device** > **CoreSight** > **ARMCS-DP**
- **Add Device** > **Registered Device** > **Cortex** > **Cortex-A8**

12. Update the CoreSight base address as follows:
    — Right-click on **Cortex-A8 Device**
    — Select **Configuration**
    — Set **CoreSight base address = 0xE0008000**
13. Select **File** > **Save**.
14. Select **File** > **Exit**.
15. In the **Connection Control** window, expand **RealView ICE**
16. For the i.MX51 EVK, click the **Cortex-A8** device to establish a connection.

NOTE

If 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 Loading EBOOT to SDRAM Using RVDEBUG

Load EBOOT to SDRAM using RVDEBUG following these steps:

1. Configure the target and desktop workstation for serial debug messages. See Section 5.1, “Serial Debug Messages,” for more information.
2. Prepare the target hardware by following the instructions in Section 5.2.1, “i.MX51 EVK Board Configuration.” See Section 5.2.2, “i.MX51 EVK 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.” and Section A.2, “RVDEBUG Configuration.”
4. Locate the RVDEBUG initialization script in the following BSP directory:
   `WINCE600\Support\TOOL\iMX51-EVK\RVD\`
5. Edit the RVD script (`RVD_MX51_DDR2.inc` for the i.MX51 EVK) 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. EBOOT serial debug messages should appear on the desktop terminal emulation application.