About the QNX SDK for Apps and Media

QNX Software Systems Limited
1001 Farrar Road
Ottawa, Ontario
K2K 0B3
Canada

Voice: +1 613 591-0931
Fax: +1 613 591-3579
Email: info@qnx.com
Web: http://www.qnx.com/

Electronic edition published: Monday, February 24, 2014

Table of Contents

About This Guide

About the QNX SDK for Apps and Media describes how to get started using the SDK to build a target image for your apps. The following table may help you find information quickly:.

To find out about:	See:
An introduction to the SDK	*What's in the QNX SDK for Apps and Media?* (p. 9)
An overview of the procedure to create a target image	*Overview of Creating a Target Image* (p. 11)
An overview of the buildfiles used to create a target image	*Overview of the buildfiles* (p. 13)
Downloading, extracting, and building the BSP	*Build a BSP for QNX SDP 6.6* (p. 19)
Generating the final buildfile and target image	*Create an Apps and Media Buildfile and Image* (p. 23)
Including your own apps in an image	*Include Apps in an Image* (p. 25)
Transferring the image to the target	*Transfer an Image to an SD Card* (p. 29)

Typographical conventions

Throughout this manual, we use certain typographical conventions to distinguish technical terms. In general, the conventions we use conform to those found in IEEE POSIX publications.

The following table summarizes our conventions:

Reference	Example
Code examples	`if( stream == NULL )`
Command options	`-lR`
Commands	`make`
Environment variables	***PATH***
File and pathnames	`/dev/null`
Function names	*exit()*
Keyboard chords	**Ctrl –Alt –Delete**
Keyboard input	`Username`
Keyboard keys	`Enter`
Program output	`login:`
Variable names	*stdin*
Parameters	*parm1*
User-interface components	**Navigator**
Window title	**Options**

We use an arrow in directions for accessing menu items, like this:

You'll find the Other... menu item under **Perspective → Show View** .

We use notes, cautions, and warnings to highlight important messages:

Notes point out something important or useful.

Cautions tell you about commands or procedures that may have unwanted or undesirable side effects.

Warnings tell you about commands or procedures that could be dangerous to your files, your hardware, or even yourself.

Note to Windows users

In our documentation, we use a forward slash (/) as a delimiter in all pathnames, including those pointing to Windows files. We also generally follow POSIX/UNIX filesystem conventions.

Technical support

Technical assistance is available for all supported products.

To obtain technical support for any QNX product, visit the Support area on our website (*www.qnx.com*). You'll find a wide range of support options, including community forums.

Chapter 1
What's in the QNX SDK for Apps and Media?

The QNX SDK for Apps and Media is a development platform that delivers multimedia support, an application management framework, and a chromeless browser. In addition, a web browser application chrome written entirely in HTML5 demonstrates the capabilities of the browser engine. These features give you the tools to create a wide variety of native and web-based applications for embedded systems.

The SDK includes:

Multimedia components

The multimedia suite supports three main functions:

- Detecting mediastores attached to the system and retrieving their device information
- Synchronizing metadata from mediastores to embedded databases
- Playing audio and video files

The platform ships with these multimedia components:

- Metadata provider library (`libmd`)
- Multimedia playlist library (`libmmplaylist`)
- Multimedia renderer service (`mm-renderer`)
- Multimedia synchronizer service (`mm-sync`)
- Multimedia test utilities (`mmcli`, `mmrplay`, and `mm-pnp`)

For more information on how these components work together to perform the three main media tasks, see the *Multimedia Architecture Guide*. For examples of playing media through the multimedia test tools, see the *Multimedia Test Utilities Guide*.

Application management framework

The application management framework includes tools for installing and uninstalling applications in your embedded system, as well as services for:

- launching applications (`launcher`)
- controlling access to system services (`authman`)

For more information about packaging and installing apps, as well as details about the Application Launcher and Authorization Manager, see the *Application and Window Management* guide.

Embedded browser

The browser included with the SDK provides a mechanism for running web-based apps in your system. Based on WebKit, the browser provides support for HTML5 and related standards and technologies (including CSS3) and for JavaScript and associated standards, such as AJAX, JavaScript Object Notation (JSON), and XML. This browser is embedded in the target image for your board. For more information about the browser (also referred to as the browser engine), see "Browser Engine" in the *HTML5 Developer's Guide*.

Chapter 2
Overview of Creating a Target Image

To use the QNX SDK for Apps and Media, you first have to create an image for your target board. When you have your board booting with this image, you can modify the buildfiles to include additional packages and applications in the generated image.

You begin by downloading a BSP from the QNX website. Among other things, the BSP should include three buildfiles:

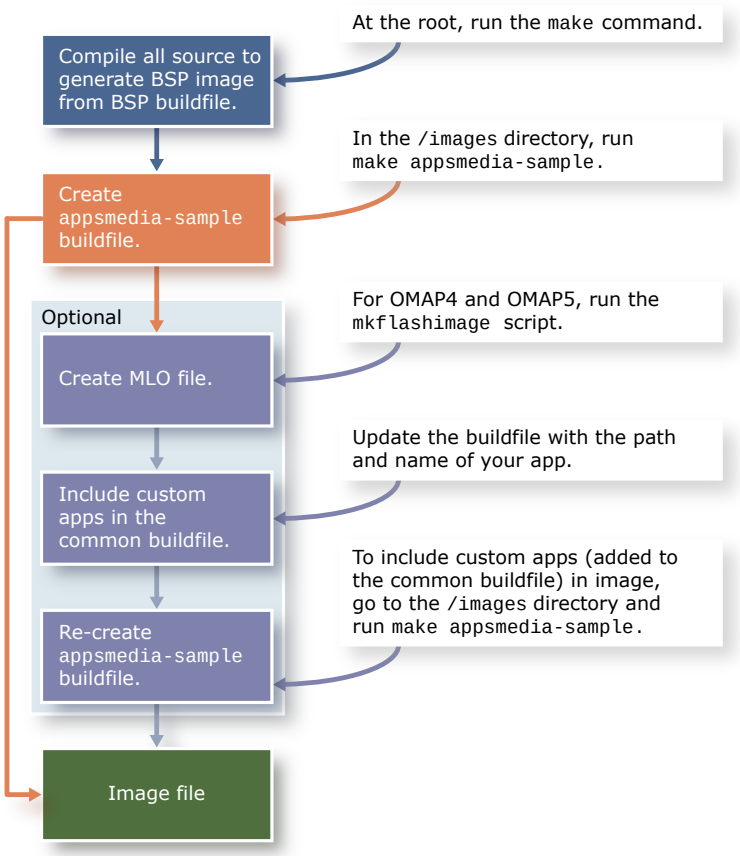
- a QNX SDP 6.6 BSP-specific buildfile
- a board-specific buildfile for Apps and Media (`appsmedia-sample-`*`board`*`.build`)
- a common buildfile for Apps and Media (`appsmedia-sample-common.build`)

You build the BSP for QNX SDP 6.6 to extract and compile the code for the board-specific drivers and libraries required by the OS.

You then use the `make appsmedia-sample` command, which uses all three buildfiles (i.e., BSP-specific buildfile, board-specific buildfile, and common buildfile) to generate a buildfile (`appsmedia-sample-`*`platform`*`.build`) and a final image (`ifs-appsmedia-sample-`*`platform`*`.bin`).

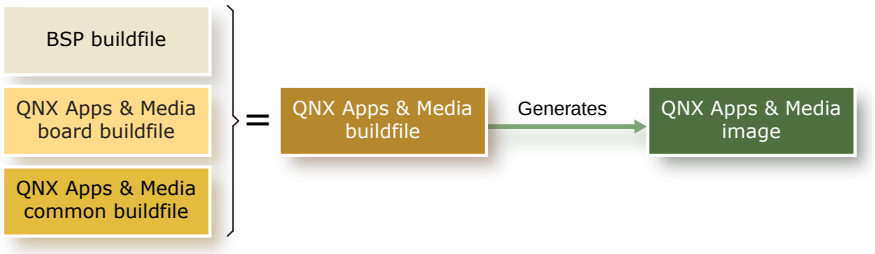
To change the contents of the generated buildfile and image, you modify the board-specific or common buildfile for Apps and Media and then regenerate the buildfile and image.

The following diagram shows an overview of the process used to create a target image for the QNX SDK for Apps and Media. This process is discussed in detail in the sections that follow.



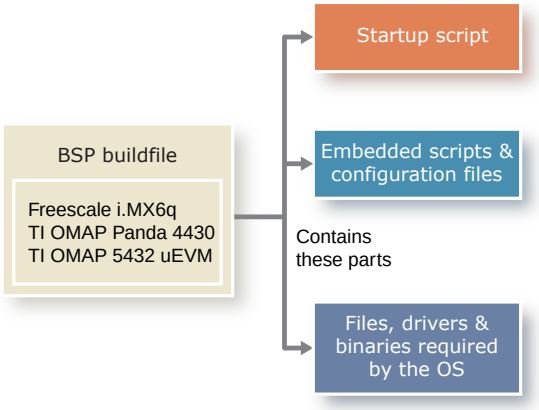
Overview of the buildfiles

After downloading your BSP, you'll have three buildfiles (a BSP-specific buildfile, a board-specific buildfile, and a common buildfile). You'll run a series of commands that combine these three files to create an Apps and Media buildfile that generates the final image, as shown in the following diagram.



The following diagram shows the main parts that make up the QNX SDP 6.6 BSP-specific buildfile. This file is found in one of the following locations:

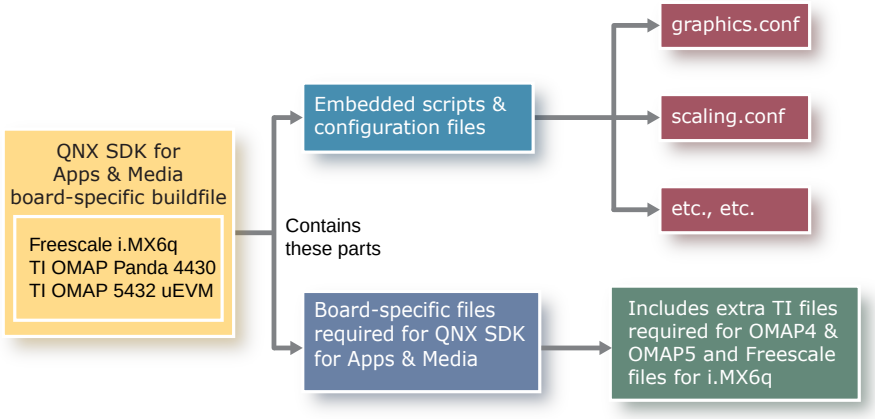
- OMAP4 Panda 4430:
  `src/hardware/startup/boards/omap4430/panda/build`
- OMAP5 5432uevm:
  `src/hardware/startup/boards/omap5-uevm5432/build`
- i.MX6q SABRELite:
  `src/hardware/startup/boards/imx6x/sabrelite/build`



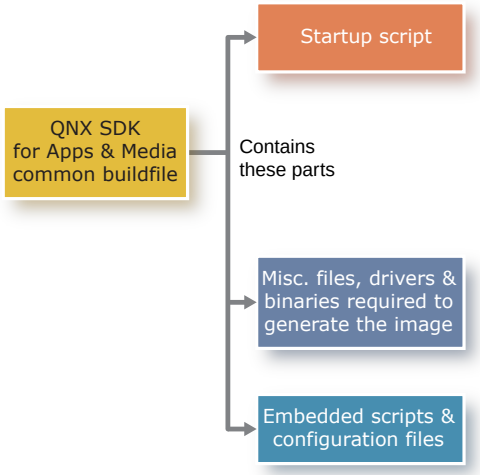
The following diagram shows the main parts that make up the board-specific Apps and Media buildfile. This file (called `appsmedia-sample-`*`board`*`.build`) is found in one of the following locations:

- OMAP4 Panda 4430:
  `src/hardware/support/appsmedia-sample/appsmedia-sample-omap4.build`
- OMAP5 5432uevm:
  `src/hardware/support/appsmedia-sample/appsmedia-sample-omap5.build`

- i.MX6q SABRELite:
  `src/hardware/support/appsmedia-sample/appsmedia-sample-imx6.build`



The following diagram shows the main parts that make up the Apps and Media common buildfile (called `appsmedia-sample-common.build`). This common buildfile contains those elements, such as network, multimedia, and browser-specific instructions, that are common to all board types.



The buildfile startup script starts:

- drivers
- remote debug tools (`qconn` and `pdebug`)
- RAM disk filesystem
- PPS
- network driver
- system date (set using `ntp`)
- `sshd` server
- touchscreen and keyboard drivers
- multimedia (`mm-renderer`)

- environment variables
- `authman` (identifies apps allowed access to services)
- `launcher` (a process to launch apps)
- `certmgr_pps` (to manage SSL certificates)

The browser chrome app is automatically installed (using `bar-install`).

The common buildfile is found at `/src/hardware/support/appsmedia-sample/appsmedia-sample-common.build` on all platforms.

Chapter 3
Prerequisites

Before you begin the process of creating an image for the QNX SDK for Apps and Media, ensure that you:

- Installed QNX Software Development Platform 6.6 for either a Windows or Linux host.
- Have `root` privilege on your host system, to avoid any issues with permissions when attempting to edit, copy, move, or delete files.

When running on a Windows host, `mkifs` can't get the *execute*(*x*), *setuid* (set user ID), or *setgid* (set group ID) permissions from the file. Use the `perms` attribute to specify these permissions explicitly. You might also have to use the `uid` and `gid` attributes to set the ownership correctly. To determine whether a utility needs to have the *setuid* or *setgid* permission set, see the utility's entry in the *Utilities Reference*.

ELF executables and shared objects are automatically marked as executable (unless you specify `[+raw]`).

- Can build and run existing QNX SDP 6.6 BSPs without any issues.
- Are familiar with the structure and layout of QNX BSPs in general.
- Understand the process of generating an image and building an embedded system. For more information, see *Building Embedded Systems*.
- Are using a QNX SDP 6.6 development environment to build your image containing the embedded browser (`appsmedia-sample` image) from the command line.
- Have a terminal-emulation program for your Windows or Linux OS (such as Qtalk, QNX Momentics IDE Terminal, minicom, Hyperterminal, and so on).
- Have a target with a touchscreen and USB keyboard for using with the browser app.

- On the Freescale i.MX6q SABRELite platform, video playback requires installation of the Freescale video codecs. You can obtain the necessary files in the `freescale-`*`datestamp`*`.zip` package from the *QNX Download Center*. Install them according to installation instructions included in the archive.
- On the OMAP5 5432uevm platform, video playback requires installation of the Texas Instruments video codecs. You can obtain the necessary files

in the `ti-`*`datestamp`*`.zip` package from the *QNX Download Center*. Install them according to installation instructions included in the archive.

- Video playback is not supported on the OMAP4 Panda 4430.
- After you unzip a BSP, a prebuilt IFS image is available in the BSP's `/images` directory. This prebuilt image is configured for the various BSP device drivers already running. When you build the BSP, the prebuilt image will be overwritten with a new image that is generated by the BSP build process, so you may want to make a copy of the prebuilt image for future reference. However, if you forget to make a copy of the prebuilt image, you can still recover the original one—simply extract the BSP from the `.zip` archive into a new directory.

Chapter 4
Build a BSP for QNX SDP 6.6

To build a BSP for QNX SDP 6.6 follow these steps.

If you're using a Windows host, run the commands in the `bash` shell.

1. Download a QNX SDP 6.6 BSP from the QNX website at *http://community.qnx.com/sf/sfmain/do/viewProject/projects.bsp* to a new directory within the QNX SDP 6.6 host environment (the archive unzips to the current directory).

   For example, you can use the following directory structure:
   $QNX_TARGET`/root/bsps/`*`my_bsp/`*

   The BSP file will be named something similar to the following:
   `BSP_`*`board_name`*`_SVN`*`xxxxxx`*`_JBN`*`yy`*`.zip`

   where *`board_name`* is the name of the board, *`xxxxxx`* is the SVN ID for the BSP, and *`yy`* is a unique ID for the BSP.

2. Navigate to the directory where you saved the BSP and extract the BSP archive file using the following command at the prompt:
   `unzip` *`bsp_filename`*

   The unzip process extracts files to the following directories:

Directory	Description
`/images`	Contains the QNX IFS, which is the image suitable for running on the target device. Any other related binaries (such as an IPL or combined IPL/IFS image) are also created in this directory. In addition, the generated IFS buildfile will also reside in this directory after the BSP builds. By default, this directory also contains a prebuilt OS image.
`/install`	When the BSP is built, the contents of the `/prebuilt` directory are copied to this location. Any binaries generated as a result of building the BSP source (contained in the BSP's `/src` directory) are also copied to the `/install` directory. The directory structure within the `/install` directory reflects the structure of a typical QNX host machine. However, when you build the IFS image, the `/install` directory is searched first for any required binaries or libraries, before the remaining

Directory	Description
	host system directories are searched. This ensures that the intended BSP components come from the IFS rather than the host system, which may have older versions of these components that existed prior to the installation of the BSP.
`/prebuilt`	Contains various header files necessary to build the source components of the BSP, as well as any prebuilt binaries or libraries for which the source code is not included with the BSP.
`/src`	Contains the source code for device drivers, libraries, and utilities.

3. To build the QNX 6.6 BSP, change to the root directory of the unzipped BSP and type the following command at the prompt:

   ```
   make
   ```

   After you build the BSP, you'll find key files in the following locations, where $***BSP_ROOT_DIR*** is the name of the directory you extracted the BSP archive in, and $***CPU_VARIANT*** is the CPU architecture the BSP is targeted for (e.g., `armle-v7` or `x86`):

File	Location
Buildfile	$***BSP_ROOT_DIR***`/images`
IPL and/or startup	$***BSP_ROOT_DIR***`/install/`$***CPU_VARIANT***`/boot/sys`
Prebuilt libraries (DLL drivers), such as audio, graphics, and network	$***BSP_ROOT_DIR***`/install/`$***CPU_VARIANT***`/lib/dll`
Generic header files (not architecture specific)	$***BSP_ROOT_DIR***`/install/usr/include`
Source code for different drivers (`sbin` drivers), such as serial, flash, block, PCI, PCMCIA, and USB	$***BSP_ROOT_DIR***`/install/`$***CPU_VARIANT***`/sbin`

Building a BSP from the IDE

When you install a BSP by importing it into the QNX Momentics Integrated Development Environment, you'll find the source code for individual BSP components in directories at the root of your IDE workspace directory.

To build a BSP from the IDE:

1. Open the QNX Momentics IDE.
2. Click **File → Import**.
3. Expand **QNX**, select **QNX Source Package and BSP** and click **Next**.
4. Click **Browse** to choose a `.zip` archive file for your BSP.
5. Select a file, click **Open**, and then click **Next**.

   The package contains the source and build files for targeting QNX Neutrino on your selected reference board.
6. Select the BSP and verify that the package is correct by reviewing the information in the **Description** field.
7. Click **Next** to specify any project-specific settings you want to set for the BSP project.

   Now you can change the name of the BSP project in the IDE, set the default location to use as the workspace for the BSP project files, and specify whether this BSP belongs to a working set.
8. Click **Finish** to import the BSP.

   You will see that the IDE created a QNX project to configure and build the BSP. To build or make modifications (e.g., to the drivers) in this BSP, the `/src` directory contains the source code to all BSP components for which the source is provided, including device drivers, libraries, and utilities.
9. To build the BSP, right-click the project file name in the project explorer, and then select **Build Project**.

The IDE generates a buildfile named *`board_BSP`*`.build` from the BSP.

Chapter 5
Create an Apps and Media Buildfile and Image

After you've built the BSP for QNX SDP 6.6, you're ready to generate the final image file for the QNX SDK for Apps and Media.

If you're using a Windows host, run the commands in the `bash` shell.

1. Change to the `/images` directory:

   ```
   cd /images
   ```

2. To generate the final Apps and Media image, type the following command:

   ```
   make appsmedia-sample
   ```

   We recommend that you redirect the output to a file and look for any warning and error messages about missing files, such as:

   ```
   Warning: host file filename missing.
   ```

   Running the `make appsmedia-sample` command generates the Apps and Media buildfile and image file:

   - `appsmedia-sample-`*`platform`*`.build` (the buildfile)
   - `ifs-appsmedia-sample-`*`platform`*`.bin` (the image file that includes an embedded browser)

The generated files are located in the `/images` directory.

- If you want to make a permanent change to the final buildfile, change the Apps and Media board-specific buildfile or common buildfile.
- If you run the command `make clean`, the generated files `appsmedia-sample` buildfile and `appsmedia-sample-`*`board`*`.build` image file will be removed.

For information about adding your own apps to the `appsmedia-sample` image, see *Include Apps in an Image* (p. 25).

Chapter 6
Include Apps in an Image

To include your apps in the `appsmedia-sample` image on a target device, you'll need to modify the buildfile you used to generate the final `appsmedia-sample` image.

If you're using a Windows host, run the commands in the `bash` shell.

To include apps in an image:

1. Change directory to `/src/hardware/support/appsmedia-sample` in the location where you extracted the BSP.

   In this location, you'll notice the following two buildfiles:

   - `appsmedia-sample-`*board*`.build`
   - `appsmedia-sample-common.build`

2. In an editor on the host machine, open the file `appsmedia-sample-common.build`.
3. Locate the section named `Browser Related`.

   This section is where you'll include information about your specific app.

4. Add the following information about your app into the `Browser Related` section of the `appsmedia-sample-common.build` file:

   ```
   [search={directory_containing_barfile}]
   /appinstall/myapp.bar=myapp.bar
   ```

   where *directory_containing_barfile* is the path to the directory that contains the `.bar` file for your app. You can use environment variables like ${***QNX_TARGET***} in the search path specification.

   For information about creating `.bar` files, see "Packaging, Installing, and Launching Apps" in the *Application and Window Management* guide.

5. Change to the `/images` directory.
6. Run the following command:

   ```
   make clean
   ```

7. To generate a new image that includes your app(s), run the following command:

   ```
   make appsmedia-sample
   ```

To install an app to a target device each time the image boots:

- In the `#Bar-install the browser app` section of the `appsmedia-sample-common.build` file, add the following command:
  `/scripts/bar-install /appinstall/myapp.bar`

The app will be installed each time the image boots. You can then run the app from the command line or from a script. For more information about launching apps, see "Launching an app on the target" in the *Application and Window Management* guide.

Chapter 7
Generate an IPL/MLO file

When some OMAP4 and OMAP5 boards start up, they require a corresponding `MLO` file with the image. If your board requires an `MLO file`, you'll need to generate an `MLO` file and include it on the SD card in order to initialize your board's boot loader. Since these boards are specifically looking for a file called `MLO`, you'll use the IPL found in the `/images` directory (e.g., for OMAP5 the name is `sd-ipl-omap5-uevm5432.bin`) and simply rename it.

To generate an `MLO` file:

1. Change directory to `/images`.
2. Run the following command:
   ```
   mkflashimage
   ```

   If you're using a Windows host, you must run the `mkflashimage` command in the `bash` shell.

   The `mkflashimage` script generates an IPL for use with flash media.
3. Rename the generated IPL file from:
   ```
   sd-ipl-board-ID_number.bin
   ```
   to:
   ```
   MLO
   ```
4. To copy the `MLO` file to the SD card:
   - For Linux, type the following command:
     ```
     cp MLO SD_card_location
     ```

     For Ubuntu, the location of the SD card will be `/media/username`.
   - For Windows, type the following command:
     ```
     copy MLO SD_card_location
     ```

- If the card is freshly formatted, the MLO file should be the first file copied to the card. Once the file is on the card, subsequent copying order doesn't matter. If the corresponding `MLO` file is not included on the SD card along with the image file, you'll receive an error message when the image attempts to load on the target device.

- Instead of using the copy command, you can use drag-and-drop to copy these files directly onto your SD card.

Chapter 8
Transfer an Image to an SD Card

After you've created the image file, you'll need to transfer it to an SD card so you can use it to boot your target device. For information about creating this buildfile and corresponding image, see *Create an Apps and Media Buildfile and Image* (p. 23).

To transfer an image to an SD card:

1. Format your SD card.

   Each BSP that you download from the QNX website comes with complete instructions on how to prepare an SD card for images.

2. Navigate to the directory where your image file is located. By default, it will be located in the `/images` directory.
3. Change the name of the file `ifs-appsmedia-sample-`*`board`*`.bin` to `qnx-ifs`.
4. Type the following command to copy the `qnx-ifs` image file to your SD card:

   - For Linux, type the following command:
     ```
     cp qnx-ifs SD_card_location
     ```

     For Ubuntu, the location of the SD card will be `/media/`*`username`*.

   - For Windows, type the following command:
     ```
     copy qnx-ifs SD_card_location
     ```

   Instead of using the `copy` command, you can use drag-and-drop to copy these files directly onto your SD card.

   To initialize a boot loader for OMAP4 and OMAP5 boards, you'll need to copy the `MLO` file as well. For information on generating an `MLO` file, see *"Generate an IPL/MLO file"* (p. 27).

Chapter 9
Load and Run the Image on the Target

Instructions for loading and running the image vary depending on your board.

Freescale i.MX6q SABRELite board

1. Insert the SD card you prepared earlier into the *lower slot* (i.e., the large, full-size SD card slot) on the SABRELite board.
2. Press the **Reset** button.
3. Interrupt the countdown by pressing any key at the first boot on this system.
4. Enter the following commands to prepare the ***U-Boot*** environment variables for booting the system:

```
MX6Q SABRELITE U-Boot> setenv sdslot '0'
MX6Q SABRELITE U-Boot> setenv loadaddr '0x10800000'
MX6Q SABRELITE U-Boot> setenv bootifs 'qnx-ifs'
MX6Q SABRELITE U-Boot> setenv bootcmd_fatload 'mmc dev ${sdslot}; fatload mmc
 ${sdslot}:1 ${loadaddr} ${bootifs}; go ${loadaddr}'
MX6Q SABRELITE U-Boot> setenv bootcmd 'run bootcmd_fatload'
MX6Q SABRELITE U-Boot> saveenv
```

5. Type `boot` or press the **Reset** button.
6. Follow the instructions to calibrate the screen.

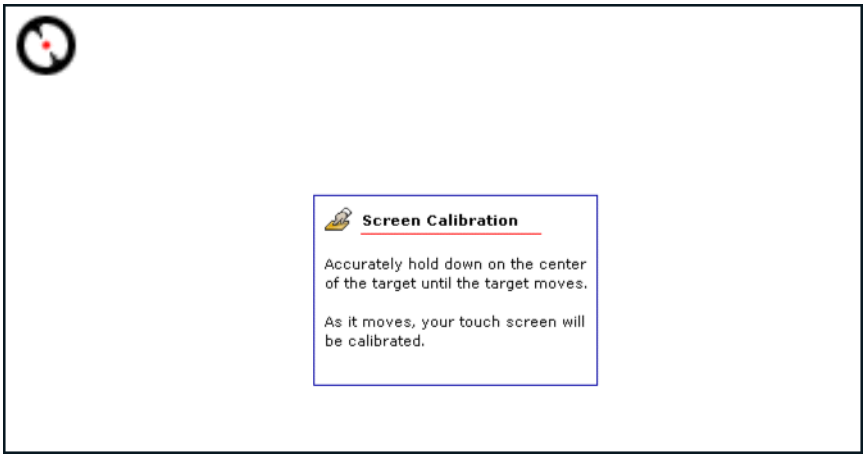
All other types of boards

1. Insert the SD card into your target board and power on the board. The board will automatically boot and prompt you to calibrate your display.
2. Follow the instructions to calibrate the screen.

Chapter 10
Calibrate the Screen and Launch the Browser

When the board boots, you'll be prompted to calibrate the screen:

1. Touch the dot in the upper-left corner and continue to touch the dot where it appears on the display until the calibration process is complete.



2. When the calibration is complete, click **Accept**.

Each time you reboot your board, you'll need to recalibrate the screen.

3. When the target system boots up, the browser frameworks starts in the background. To launch the browser app, run the following command:

```
launch sys.browser
```

The browser app requires both a touchschreen and a USB keyboard on the target.

The target system loads the browser app:

Since RAM disk is used as the filesystem, all changes made directly to the generated `appsmedia-sample` buildfile will be deleted when you reboot your target device.

Chapter 11
Play media

You can play media on your target by inserting a mediastore and then issuing commands with mmrplay.

When you insert a mediastore (i.e., device with media content), the system automatically assigns a mountpoint. For a USB device, its filesystem is mounted to `/fs/usb0`. You must know the full paths of the media files that you want to play.

To play media files:

- Open a QNX Neutrino terminal on the target board and enter a command like this:
  ```
  # mmrplay /fs/usb0/stillness_in_time.mp3
  ```
  This command plays an MP3 audio track but you would use the same syntax to play tracks in different formats or video files. For a list of supported file formats, see the Release Notes for QNX SDK for Apps and Media.

When playing video files, you must use the -v option with a value of `screen:` to tell `mmrplay` to direct the output to the display.

The `mmrplay` utility starts playing the media file at the specified path. The command returns when playback finishes. For more detailed playback examples, see the *Multimedia Test Utilities Guide*.

Chapter 12
Modify your buildfile to support 720p screen resolution

The default buildfiles requires some modifications to support a screen resolution of 720p. Make the following changes to the `src/hardware/support/appsmedia-sample/appsmedia-sample-`*`board`*`.build` file:

1. In the `start-graphics` section, change:

   ```
   #Start devi-hid with a width and height parameter.
   #The width and height must match graphics.conf
   devi-hid -R800,480 touch kbd &
   ```

   to:

   ```
   #Start devi-hid with a width and height parameter.
   #The width and height must match graphics.conf
   devi-hid -R1280,720 touch kbd &
   ```

2. In the `graphics.conf display internal configuration` section, change:

   ```
   video-mode = 800 x 480 @ 60
   ```

   to:

   ```
   video-mode = 1280 x 720 @ 60
   ```

3. In the `graphics.conf mtouch configuration` section, change:

   ```
   options = height=480,width=800,poll=1000
   ```

   to:

   ```
   options = height=720,width=1280,poll=1000
   ```

4. In the `scaling.conf` section, change:

   ```
   800x480:mode=direct
   ```

   to:

   ```
   1280x720:mode=direct
   ```

5. For i.MX6 boards, you must make additional changes:

   - In the `graphics.conf dislay 1 configuration` section (line 117), change:

   ```
   # HDMI display
   wfd-dlls = libwfdcfg-imx6x-hdmi.so libimx6xCSCgamma-generic.so libWFDimx6x.so
   ```

   to:

   ```
   # 720p display
   wfd-dlls = libwfdcfg-imx6x-qnx-car.so libimx6xCSCgamma-generic.so libWFDimx6x.so
   ```

   - In the `Graphics files` section (line 338), add:

   ```
   /usr/lib/graphics/iMX6X/libwfdcfg-imx6x-qnx-car.so=libwfdcfg-imx6x-qnx-car.so
   ```

6. To regenerate the final Apps and Media image, type the following command:
   ```
   make appsmedia-sample
   ```

Index