BSP User's Guide

NXP i.MX6x SABRE for Automotive Infotainment (AI)

BlackBerry | QNX

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: November 01, 2018

Contents

About This Guide

This guide contains installation and start-up instructions for the QNX Board Support Package (BSP) for the NXP i.MX6x SABRE for Automotive Infotainment.

Here's what this guide provides:

- an overview of the BSP for the SABRE AI board
- what's new with working with BSPs for this release
- structure and contents of a BSP
- how to prepare a bootable SD
- how to transfer an image to the SD
- how to boot your board
- how to build the BSP using the command line or the QNX Momentics IDE if you modify the BSP

This BSP is valid for NXP i.MX6x SABRE for Automotive Infotainment boards with Quad, Dual, DualLite, and Solo processors. These boards are sometimes also called "i.MX6x SABRE Automotive Reference Design (ARD)." For convenience, we refer to the NXP i.MX6x SABRE for Automotive Infotainment board as simply the "SABRE AI" board.

NXP Semiconductors (*www.nxp.com*) acquired Freescale Semiconductor in December 2015; the SABRE AI boards were previously known as Freescale i.MX6x SABRE Platform ARD. We have changed the corresponding QNX BSP names.

To find out about:	See:
The resources available to you, and what you should know before starting to work with this BSP	*Before You Begin*
What's included in the BSP, and supported host OSs and boards	*About This BSP*
Installing this BSP. You can install the images provided with this BSP to boot your board	*Installation Notes*
Building this BSP. You can modify the contents of the BSP or modify the buildfile to build a customized image	*Build the BSP*
Using graphics. You can run applications to show content on a display	*Using the Screen Graphics Subsystem*
Using touch	*Using Multitouch*
Using audio	*Using Audio Input and Output*
Driver commands	*Driver Commands*

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`
Constants	NULL
Data types	`unsigned short`
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.

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

DANGER: 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 typically use a forward slash (/) as a delimiter in 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
Before You Begin

Before you begin working with this BSP, you should become familiar with the resources available to you.

Essential information

Before you begin building and installing your BSP, you should review the following documentation:

- Information about your board's hardware and firmware provided by the board vendor. Documentation for the NXP SABRE AI boards can be found at: *http://www.nxp.com/products/software-and-tools/hardware-development-tools/sabre-development-system/sabre-for-automotive-infotainment-based-on-the-i.mx-6-series:RDIMX6SABREAUTO?fpsp=1&tab=Documentation_Tab*
- *Building Embedded Systems*. This guide contains general information about working with BSPs from QNX Software Systems. This information is common to all QNX BSPs and is available from QNX Software Development Platform 7.0 documentation.

Required software

To work with the BSP for the SABRE AI board, you require the following:

- ZIP file for the BSP
- QNX Software Development Platform 7.0 installed on a Linux, macOS, or Windows host system
- Virtual COM Port (VCP) Driver. This driver must be installed on the host system that connects to the board. To get the driver and read more information about it, see the Future Technology Devices International Ltd. (FTDI) website at *http://www.ftdichip.com/FTDrivers.htm*.
- a terminal emulation program (Qtalk, Tera Term, Hyperterminal, PuTTy, etc.) to connect to the board for debugging. Alternatively, you can use a `telnet` or `ssh` session from the QNX Momentics IDE.
- if you are using a Windows host, you require the CFImager utility, which is required to put U-Boot or the QNX IPL on the SD card that's used to boot the board. For more information and to download the tool, see the *i.MX Software Development Tools* page on the NXP website.
- if you want to boot the board to run QNX Neutrino RTOS using U-Boot, you must download the version of U-Boot for your board from the NXP website. This BSP was tested using the U-Boot files from a package called *i.MX 6 Quad, i.MX 6 Dual, i.MX 6 DualLite,i.MX 6 Solo and i.MX 6 Sololite Linux Binary Demo Files*.

Chapter 2
About This BSP

These notes list what's included in the BSP and identify the host OSs and boards it supports.

What's in this BSP

This BSP contains the following components:

Component	Format	Comments
IPL	Source and binary	QNX IPL supports booting the board from serial, SD card, and NAND
Startup	Source and binary	
Watchdog utility	Source and binary	**wdtkick**
Serial driver	Source and binary	
Inter-Integrated Circuit (I2C) driver	Source and binary	
SD/MMCdriver	Source and binary	
SPI NOR Flash driver	Source and binary	
Controller Area Network (CAN) driver	Source and binary	
Real-time Clock	Source and binary	
Audio driver	Source and binary	
PCI driver	Binary only	Included in QNX SDP 7.0
Enhanced SPI driver	Source and binary	
USB OTG host controller (`io-usb-otg` stack)	Binary only	Included on QNX SDP 7.0
USB OTG device controller (`io-usb-otg` stack)	Source and binary	Included in QNX SDP 7.0
Network (Ethernet)	Source and binary	

Component	Format	Comments
Parallel NOR	Source and binary	
SATA driver	Binary only	Included on QNX SDP 7.0
PMIC	Source and binary	
Graphics	Partial source and binary	The **wfdcfg** source is included in the BSP
Touch	Binary only	Included on QNX SDP 7.0
User's Guide	PDF	This document

Supported OSs

To install and use this BSP, you must have installed the QNX SDP 7.0, on a Linux, macOS, or Windows host. In addition, you require the following:

This BSP supports the following target OS:

- QNX Neutrino RTOS 7.0

Supported boards

In the QNX Software Center, see the release notes for this BSP for the list of supported boards using these steps:

1. On the **Available** or **Installed** tab, navigate to Board Support Packages BSP, right-click *Name_of_your_BSP*, and then select **Properties**.
2. In the **Properties for** *BSP_name* window, on the **General Information** pane, click the link beside **Release Notes**.

You should be redirected to the release notes on the QNX website. To see the notes, you must be logged in with your myQNX account.

Interrupt table for this board

The interrupt vector table can be found in the buildfile located at **src/hardware/startup/boards/imx6x/sabreARD/build** in your BSP. This interrupt table is useful when you want to write custom resource managers or customize the BSP.

Known issues

For the list of known issues, see the release notes for this BSP.

Chapter 3
Installation Notes

These installation notes describe how to install and boot a board using this BSP.

Here's the process to install the BSP and run it on your board:

1. Download the BSP, set up your environment, and extract the BSP. For more information, see "*Download and set up the BSP*."
2. Connect your target board. After you connect the hardware, it may be necessary set DIP switches on your board to further configure it. For more information about how to connect to your board, see "*Set up the hardware*."
3. Create a bootable SD card before your can use the BSP images. After you create a bootable SD card, you can either transfer prebuilt images that come with the BSP or you can customize the buildfile and build your own image. For more information about how to create a bootable SD card, see "*Prepare a bootable SD card*."
4. If this is the first time you are bringing up the BSP, you can transfer the provided IFS image file to the SD to boot your board using U-Boot. To get U-Boot, you must get it from the vendor of the SABRE AI. Alternatively, you can use QNX IPL to boot the IFS image. By default, the prebuilt IFS images provided with this BSP are uncompressed.

   If you want to use QNX IPL to boot a compressed IFS, you must modify the buildfile to build a compressed IFS. For more information, see *Build the BSP* in this guide. For information about how to transfer an image to a card, see "*Transfer the image*."
5. After you have your image on the card, you can boot it. For more information about booting your board, see "*Power up the board*."

After your board boots, you can use that image to understand how it works. Depending on the other devices you have connected to the board, you may need to modify configuration settings on the image.

At some point, you may want to customize the image and rebuild it. For specific information about building the image for this board, see the "*Build the BSP*" chapter in this guide. For information about modifying the image and generic information about building the image, see the *Building Embedded Systems* guide, which is available in the QNX SDP 7.0 documentation.

Download and set up the BSP

You can download Board Support Packages (BSPs) using the QNX Software Center.

BSPs are provided in a ZIP archive file. You must use the QNX Software Center to install the BSP, which downloads the BSP archive file for you into the **bsp** directory located in your QNX SDP 7.0 installation. To set up your BSP, you can do one of the following steps:

- Import the BSP archive file into the QNX Momentics IDE (extracting the BSP isn't required)
- Navigate to where the BSP archive file has been downloaded, and then extract it using the `unzip` utility on the command line to a working directory that's outside your QNX SDP 7.0 installation.

  The `unzip` utility is available on all supported host platforms and is included with your installation of QNX SDP 7.0.

  The QNX-packaged BSP is independent of the installation and build paths, if the QNX Software Development Platform 7.0 is installed. To determine the base directory, open a Command Prompt window (Windows) or a terminal (Linux and macOS), and then type `qconfig`.

Extract from the command line

We recommend that you create a directory named after your BSP and extract the archive from there:

1. In a Command Prompt window (Windows) or a terminal (Linux and macOS), run **qnxsdp-env.bat** (Windows) or `source` **qnxsdp-env.sh** (Linux, macOS) from your QNX SDP 7.0 installation to set up your environment. You must run the shell or batch file to use the tools provided with QNX SDP 7.0.

   If you type `env`, you should see environment variables that are set such as *QNX_TARGET*.

2. Navigate to the directory where you want to extract the BSP (e.g., `cd` **/BSPs/bsp-nxp-mx6x-sabreARD**). The directory where you extracted the BSP is referred to throughout this document as *bsp_working_dir*. The archive is extracted to the current directory, so you should create a directory specifically for your BSP. For example:

   `mkdir` *bsp_working_dir*

   You should also set the *BSP_ROOT_DIR* environment variable to point to your *bsp_working_dir*. You can use the `export BSP_ROOT_DIR=`*bsp_working_dir* (Linux and macOS) or `set BSP_ROOT_DIR=`*bsp_working_dir* (Windows) command to set the environment variable.

3. In the directory you created in the previous step, use the `unzip` command to extract the BSP. For example:

   `cd /`*bsp_working_dir*

   `unzip -d /`*bsp_working_dir* **BSP_nxp_imx6x-sabreard-700***build_ID***.zip**

   where *build_ID* is the BSP build number (e.g., `WBN14`).

You should now be ready to build your BSP. See the "*Build the BSP (command line)*" chapter for more information.

Import to the QNX Momentics IDE

If you use the QNX Momentics IDE to build your BSP, you don't need to extract the ZIP file. To import the BSP code into the IDE:

1. Open the QNX Momentics IDE.
2. From the C/C++ Perspective, select **File** → **Import**.
3. Expand the **QNX** folder.
4. Select **QNX Source Package and BSP (archive)** from the list and then click **Next**.
5. In the **Select the archive file** dialog, click **Browse...**.
6. In the Open dialog box, navigate to and select the BSP archive file that you downloaded earlier, and then click **Open** and then click **Next**.
7. Confirm the BSP package you want is shown in the **Selected Package** list and then click **Next** to proceed importing the BSP archive file.
8. Optionally, set the **Project Name** and **Location** to import the BSP archive file. Defaults are provided for the project name and location, but you can override them if you choose.
9. Click **Finish**. The project is created and the source brought from the archive.

You should see the project in the **Project Explorer** view (e.g., **bsp-nxp-imx6x-sabreARD**). You can now build your BSP. For more information, see the "*Build the BSP (IDE)*" section in the "*Build the BSP*" chapter of this guide.

Structure of a BSP

After you unzip a BSP archive using the command line, or if you look at the source that's extracted by the IDE in your workspace, the resulting directory structure looks something like this:

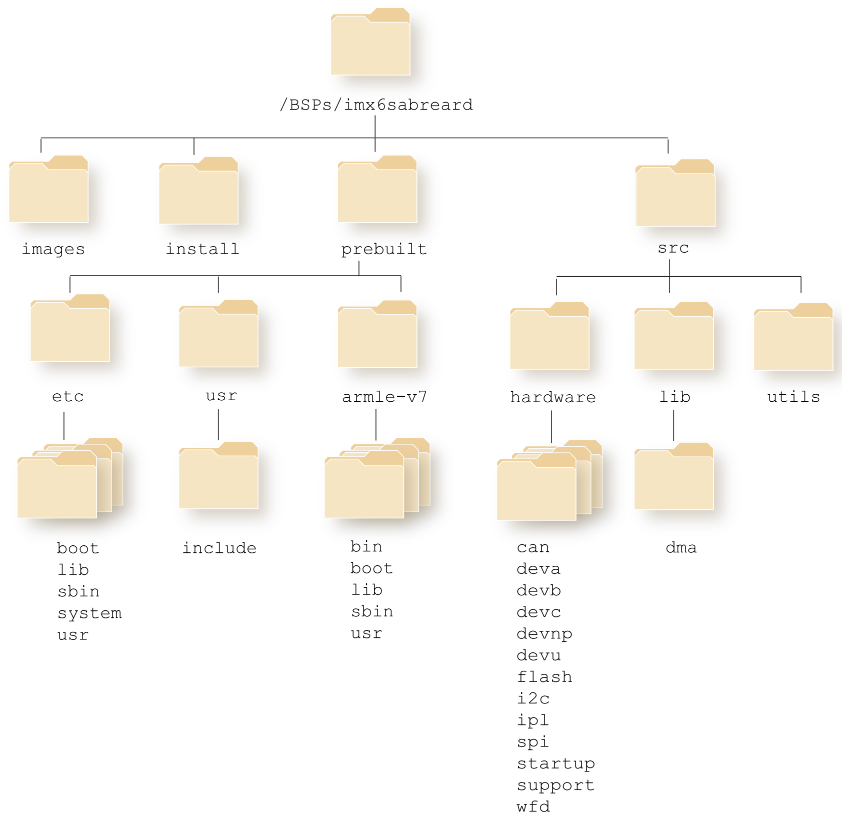


Figure 1: Structure of a BSP

For more information about the contents of the directory structure, see the *Building Embedded Systems* guide in the QNX SDP 7.0 documentation.

Set up the hardware

You must configure switches on your board and set it up for use with the BSP.

To verify that the switches on the board are correct, see the "*Configure the board switches*" section in this chapter. After you configure the switches on the board, connect the hardware.

CAUTION:

Use only the power supplies provided with the board. Use of any other power supply can permanently damage the board.

When cycling the power on the board, turn the power on or off from the AC adapter side, instead of removing and inserting the DC power plug on the board. Cycling the power on the AC adapter side prevents damage to the board's power-regulation circuitry.

To work with this board, you also require the following cables:

- an RS-232 serial port, or a USB-to-serial cable
- a null-modem serial cable
- an Ethernet link (or USB-to-Ethernet adapter)

Board connectors

The NXP i.MX6x SABRE for Automotive Infotainment board includes these power supplies:

5V DC

(Optional) Connect to **J8** on the CPU board.

The 5V DC power supply can be used to power the CPU board when the 12V DC power supply isn't connected to the main board.

12V DC

Connect to **J1** on the *main* board.

The 12V DC power supply is used to power the both the main board and the CPU board. When the main board has power, you don't need to use the 5V DC power supply for the CPU board.

DANGER: **Never** connect the 12V DC power supply to the CPU board. If you do this, you'll *destroy the board*.

Connecting the board

You require a RS-232 serial device to a USB port cable to connect to the UART Debug port on the board.

To connect to your board, after you have *configured the switches*:

1. Connect a straight-through serial cable between the board's serial port and the first available serial port of your host machine (e.g. **/dev/ttyS*** on Linux, COM1 on Windows, etc).
2. Connect the 12V DC power supply to the **main** board's **J1** connector. Alternatively, you can connect the 5V DC power supply to the CPU board's **J8** connector (see "*Power up the board*" above).
3. Identify the host serial port on your host system:

   On a Linux or macOS host, you can check which port is the host serial port by looking at what port appears when the cable is inserted. Do do this, type the command:

   ```
   $ ls /dev/ttyUSB*
   ```

   On a Windows system, open the **Device Manager**, expand the **Ports (COM and LPT)**, and then look for a COM port named **Gadget Serial** or **USB Serial Port**.
4. Connect to the COM port using a terminal or terminal program using these settings:
   - Baud rate: **115200**
   - Data: **8 bit**
   - Parity: **none**
   - Stop: **1 bit**
   - Flow control: **none**

   For more information about the VCP driver and terminal programs, see the "Required software" section of the *Before You Begin* chapter in this guide.

Configure the board switches

You can set the board's DIP switches to select different functionality.

The SABRE AI board DIP switches should be set as shown below to select the desired functionality. Note that the black block represents the physical lever inside the switch, the asterisk (*) means you must check the board's reference manual for the boot configuration options, and the X means you should not modify the switch setting:

SD on CPU board

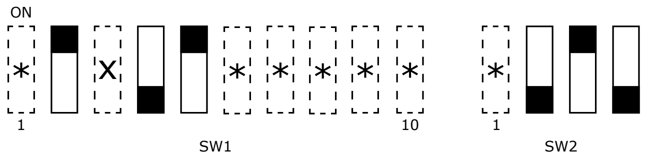


Figure 2: SW1 and SW2 set for SD on CPU board

Serial NOR Flash

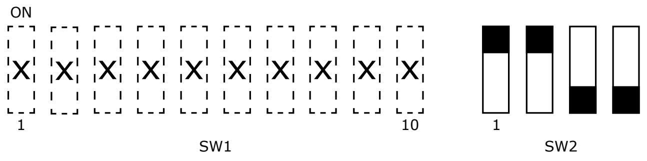


Figure 3: SW1 and SW2 set for serial NOR Flash

Parallel NOR Flash

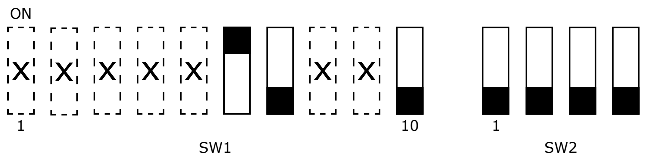


Figure 4: SW1 and SW2 set for parallel NOR Flash

System layout

The table below shows the memory layout for the image that's created from this BSP:

Item	Start	End
OS image location	`0x10800000`	

Prepare a bootable SD card

You prepare a bootable SD card on a Linux, macOS, or Windows host system. We recommend that you use Class 10 (or UHS-1) SD cards.

Depending on your host OS, follow the steps in either the "Prepare a bootable SD card on a Linux or macOS host" or "Partition and format the SD card on a Windows host" section to prepare your SD card.giyu After you've prepared the card, you can copy the image to it so that you can boot the board.

CAUTION: Ensure that your SD card isn't write-protected.

Typically there's a lock switch on the side of an SD card or SD card case (for microSD cards). When this switch is in the lock position, you can't modify or delete the contents of the card. Make sure that this switch is in the unlock position so that you can format and partition the card.

Prepare a bootable SD card on a Linux or macOS host

The following is a quick, step-by-step method for formatting the SD card from a terminal:

1. Run the following command, once with the SD card out of the reader on your host system:

   ```
   $ mount
   . . .
   ```

   Note the mounted devices listed. Now insert the card and run the same command a second time.

   ```
   $ mount
   . . .
   /dev/sda1
   ```

   When the command is run with the card inserted, an additional device should appear (for example, **sda1** or **mmcblk0p1**; what you see may differ on your computer). This additional device is the target device.

   For the remainder of these instructions, we'll use **sda1** for the device used. Substitute your own device that you noted when you ran the `mount` command for the second time.

2. If the device is mounted, unmount it. For example:

   ```
   $ umount /dev/sda1
   ```

   After your run the command, your SD card is now ready to be partitioned.

3. Using administrator privileges, run the following commands, substituting your device for **sda1**, and responding to the prompts as indicated. For example:

   ```
   $ sudo fdisk /dev/sda
   ```

   Since you are formatting the whole SD card and not just a partition, you may need to strip the identifier at the end of the mountpoint when you run the `fdisk` command. For example, for **sda1**, the identifier is `1` or for **mmcblk0p1**, the identifer is `p1`. For more information about the identifier, see the user guide for the variant of Linux or macOS that you're running.

4. Remove the existing partitions. Keep typing the `d` command until no partitions are left. For example:

```
Command (m for help): d
No partition is defined yet!
```

5. Type `u` and then `o` to Create the FAT32 partition (partition type: 12):

```
...

Command (m for help): u
Changing display/entry units to cylinders

Command (m for help): o

Building a new DOS disklabel with disk identifier 0xcdd1b702.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by write.

WARNING: cylinders as display units are deprecated. Use command 'u' to change units to sectors.
```

6. Type `n` followed by `p` to create a new primary partition. For example:

```
Command (m for help): n

Partition type:
p   primary (0 primary, 0 extended, 4 free)
e   extended
Select (default p): p
```

7. Type `1` to set the partition as the primary partition and specify the defaults for your card. For example, enter the specified defaults for *start cylinder* and *end cylinder*, which in this example are 1 and 240. Your defaults may be different depending on the size of your SD card.

```
Partition number (1-4, default 1): 1

First cylinder (1-240, default 1): start cylinder

Last cylinder, (100-240, default 240): end cylinder

Using default value 240
```

8. Type `a` to set the active partition. Generally, this should be partition 1.

```
Command (m for help): a Partition number (1-4): 1
```

9. Type `t` and then `c` to set the partition type (1 is default).

```
Command (m for help): t
Selected partition 1

Hex code (type L to list codes): c
Changed system type of partition 1 to c (W95 FAT32 (LBA))
```

10. Type `w` to write the changes.

```
Command (m for help): w

The partition table has been altered!
```

```
Calling ioctl() to re-read partition table.
Syncing disks.
```

11. Format the SD card. Note that the identifier, such as "1" or "p1" at the end of the mountpoint must be included:

```
$ sudo mkfs.vfat /dev/sda1
mkfs.msdos 3.0.12 (16 Jan 2017)
```

Partition and format the SD card on a Windows host

The default Windows formatting tool that appears when you insert a blank (or unrecognized) SD card into a Windows host isn't sufficient to format the SD card with a bootable partition. In addition, some SD cards come with pre-created partitions that aren't suitable to boot the board. Instead, use the diskpart command-line tool to format your SD card.

1. Start a Command Prompt window (e.g., on Windows 10, run **Start Menu** → **All Programs** → **Accessories** → **Command Prompt**). Ensure that you start the Command Prompt window using Administrator privileges.
2. Run the `diskpart` utility from your Command Prompt window:

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\windows\system32>diskpart

Microsoft DiskPart version 6.3.9600

Copyright (C) 1999-2013 Microsoft Corporation.
On computer: CI0700000001064

DISKPART>
```

3. Run the `list disk` command to get a list of available drives, then identify your SD card on this list (in this example, Disk 3):

```
DISKPART> list disk

Disk ###     Status       Size       Free      Gyn      Gpt
--------     ------       ----       ----      ---      ---
Disk 0       Online       40 GB      20 GB
Disk 1       No Media      0  B       0  B
Disk 2       No Media      0  B       0  B
Disk 3       Online       14 GB       0  B
```

4. Run the `select disk` command to perform operations on the SD card:

```
DISKPART > select disk 3

Disk 3 is now the selected disk.

DISKPART >
```

5. Run the `clean` command to erase any existing partitions:

```
DISKPART> clean
```

```
DiskPart succeeded in cleaning the disk.

DISKPART>
```

6. Create a 256 MB partition with a 1024 byte offset, by running the following command:

```
DISKPART> create partition primary size=256 offset=1024

DiskPart succeeded in creating the specified partition.

DISKPART>
```

You can choose a larger size for the partition. The offset you use may vary based on the hardware platform. For more information regarding the offset to use, see the documentation for your hardware platform.

7. Run the `list partition` command, to see the partition you created:

```
DISKPART> list partition

  Partition ###  Type              Size     Offset
  -------------  ----------------  -------  -------
* Partition 1    Primary            256 MB  1024 KB
```

8. Run the `select partition` command to perform operations on the partition that you created in the previous step:

```
DISKPART> select partition 1

Partition 1 is now the selected partition.

DISKPART>
```

9. Run the `active` command to make the partition active:

```
DISKPART> active

DiskPart marked the current partition as active.
```

10. Run the `format` command to format the partition:

```
DISKPART> format fs=fat32 label="CMD" QUICK

  100 percent completed

DiskPart successfully formatted the volume.

DISKPART>
```

11. Run the `list partition` command to verify that the partition is active and that you have the correct partition size and offset:

```
DISKPART> list partition

  Partition ###  Type              Size     Offset
  -------------  ----------------  -------  -------
* Partition 1    Primary            256 MB  1024 KB
```

```
DISKPART>
```

The “*” (asterisk) beside the partition name indicates that the partition is active. You can now copy the QNX IPL or U-Boot file and IFS file to the partition.

12. Exit the `diskpart` utility:

```
DISKPART> exit

Leaving DiskPart...

C:\windows\system32>
```

Transfer the image

After you format and make your SD card bootable, you are ready to transfer the image to it.

The SABRE AI board has its reset vector offset by `1024` bytes, so when you copy your IPL to the SD card, you must copy it to this offset. Linux and macOS hosts allow you copy the the image to a specific offset using the `dd` command from a terminal. On a Windows host, you must use the CFImager tool. For information about how to get the CFImager tool, see "Required software" in the "*Before You Begin*" chapter of this guide.

Provided with your BSP are prebuilt files that you can use. Alternatively, if you make modifications to the buildfile, you can rebuild the IFS and IPL. For information about building your BSP, see the "*Build the BSP*" chapter.

In addition, you can use the QNX IPL or U-Boot (provided by the board vendor) to load the QNX IFS.

- To boot using U-Boot, you must get the applicable U-Boot software for your board. For more information about how to transfer to your SD card and boot using U-Boot, see "*Use U-Boot to boot the board*" section in this chapter.
- Instead of using U-Boot, you can boot your board using QNX IPL. For information about how to transfer the QNX IPL and IFS files to your SD card, see the "*Use QNX IPL to boot board*" section in this chapter.

Use U-Boot to boot the board

Instead of using QNX IPL to load the QNX IFS, you can use U-Boot.

Depending on the type (Solo, Dualite, Dual, Quad) and revision of the board, you may need to use different U-Boot files, which you can get for your board from the NXP website; *http://www.nxp.com*.

Put U-Boot and the QNX IFS image on the card (Linux and macOS)

1. From your host system, copy the U-Boot file as the first file to your formatted SD card. U-Boot must start on the 1024th byte of your SD card. Depending on your board, you may require a different U-Boot file from the hardware vendor. For example, if you're using the Quad variation of the board, you would use the following command:

   ```
   $ sudo dd if=u-boot-imx6qsabreard_sd.imx of=/dev/sdX bs=1k seek=1 conv=fsync
   ```

2. Synchronize the SD card using this command:

   ```
   $ sync
   ```

   After you run the command, remove and re-insert the SD card.

3. Navigate to location of the QNX IFS file (**ifs-mx6x-sabreARD.bin**) and then copy it to the SD card.

   For example, on a Linux host system (Ubuntu), use these commands on the default FAT mountpoint, **/media/$LOGNAME/boot**:

   ```
   $ cd $BSP_ROOT_DIR/images
   $ cp ifs-mx6x-sabreARD.bin /media/$LOGNAME/QNX-IFS
   ```

4. Unmount the FAT partition:

   ```
   $ umount /media/$LOGNAME/boot
   ```

Put U-Boot and the QNX IFS image on the card (Windows)

The Windows host doesn't support the `dd` utility used on Linux and macOS hosts to convert files while copying them. NXP provides the `CFImager` utility, which you can download from the NXP website. For information about how to get the CFImager tool, see “Required software” in the *Before You Begin* chapter of this guide. These steps presume that you have installed `CFImager` utility on your host.

1. In a Command Prompt window(you should run this as an Administrator), navigate to the location where you installed `CFImager` and run it to copy the U-Boot file to the offset of 1024 B on the SD card. For example, where `G` is the location of your SD card:

   ```
   > cfimager.exe -raw -offset 0x400  -skip0x400 -f u-boot-imx6qsabreard_sd.imx -d G
   ```

   `CFImager` should output something like this:

   ```
   drive = G
   removable = no
   device block size = 512 bytes
   device block count = 0xed49c8

   firmware size = 0x72c00 bytes (0x396 blocks)
   extra blocks = 1128
   Writing firmware...
   done!
   ```

2. In a Command Prompt window, run `bash`. You might need to run the **qnxsdp-env.bat** from your QNX SDP 7.0 installation to set up your environment to use `bash`.

   ```
    $ bash
   ```

3. Navigate to the ***$BSP_ROOT_DIR*/images** directory and copy the QNX IFS (**ifs-mx6x-sabreARD.bin** image) to the card.

   ```
   $ cd $BSP_ROOT_DIR/images
   $ cp ifs-mx6x-sabreARD.bin SD_location_of_card/QNX-IFS
   ```

Use QNX IPL to boot board

You can use the QNX IPL (Initial Program Loader) to load your QNX IFS image.

You can use the QNX IPL to boot your board and load the QNX IFS to run the QNX Neutrino RTOS on the target. Depending on the host you use, you must use the steps to copy the QNX IPL and IFS to your SD card using the steps in “*Copy the QNX IPL and IFS (Linux and macOS hosts)*” or “*Copy the IPL and IFS (Windows)*.”

Copy the QNX IPL and IFS (Linux and macOS hosts)

To copy the images to an SD card, perform the following steps:

1. Insert the SD card into your host system and mount it if necessary.

   The SD card should appear in the list of mounted devices. If it doesn't appear, remove and re-insert the card into your host system.

2. In a terminal on your host system, navigate to the ***$BSP_ROOT_DIR*/images** directory and copy the **ipl-mx6x-sabreARD.bin** to the SD card at offset of `1024` bytes using the `dd` command.

   The number 1 at the end of the specified mountpoint must be removed, so for the mountpoint **/dev/sdd1**, you would specify **/dev/sdd**.

   For example:

   ```
   cd $BSP_ROOT_DIR/images
   dd if=ipl-mx6x-sabreARD.bin of=/dev/sdd bs=512 seek=2 skip=2
   ```

3. Copy the IFS image file to the SD card. To do this, you copy the file to the local mountpoint. If necessary, run the `mount` command again to determine your mountpoint. For example:

   ```
   $ mount
   ...
   /dev/sdd1 on /media/074B-DAC7

   $ cp ifs-mx6x-sabreARD.bin /media/074B-DAC7/qnx-ifs
   ```

The SD card should now be ready to insert into your SABRE AI board.

Copy the QNX IPL and IFS (Windows)

The Windows host doesn't support the `dd` utility used on Linux and macOS hosts to write files to a specific offset. Instead, use the `CFImager` utility, which you can download from the NXP website. For information about how to get the CFImager tool, see “Required software” in the *Before You Begin* chapter of this guide. These steps presume that you have installed `CFImager` utility on your host.

1. In a Command Prompt window (you should run this as an Administrator), navigate to the location where you installed the `CFImager` utility and run it to copy the IPL to offset of 1024 bytes onto the SD card. For example, where `G` is the location of your SD card, the command looks like this:

   ```
   > cfimager.exe -raw -offset 0x400 -skip 0x400 -f ipl-mx6x-sabreARD.bin -d G
   ```

   The output from `CFImager` should look like this:

   ```
   drive = G
   removable = no
   device block size = 512 bytes
   device block count = 0x1dd071a

   firmware size = 0x44000 bytes (0x220 blocks)
   extra blocks = 1502
   Writing firmware...
   done!
   ```

2. In a Command Prompt window, run `bash`. You might need to run the **qnxsdp-env.bat** from your QNX SDP 7.0 installation to set up your environment to use `bash`.

   ```
   $ bash
   ```

3. Navigate to the ***$BSP_ROOT_DIR*/images** directory and copy the QNX IFS (U-Boot **ifs-mx6x-sabreARD.bin** image) to the card.

   ```
   $ cd $BSP_ROOT_DIR/images
   $ cp ifs-mx6x-sabreARD.bin SD_location_of_card/qnx-ifs
   ```

   If you use File Explorer to move files, rename the **ifs-mx6x-sabreARD.bin** file to **qnx-ifs** before you drag it to the SD card.

Power up the board

After you've transferred your images to the SD card, you're ready to boot the board.

To boot your board, make sure you have the hardware connected. If it's the first time you're booting your board with the image, it's a good idea to connect a console to see that the board boots correctly. For more information about connecting your board, see the "*Set up the hardware*" section in this guide.

DANGER: Never connect the 12V DC power supply to the board with the CPU. If you do, this, the board will be damaged. For more information, see the "*Set up the hardware*" section in this chapter.

Boot the board using U-Boot

After you've loaded the applicable U-Boot for your type and revision of the board, you can use U-Boot to load the QNX IFS using. To do this, do the following steps:

1. Insert the SD card into the board's SD slot.
2. Connect the 12V DC power supply to the **main** board's **J1** connector. Alternatively, you can connect the 5V DC power supply to the CPU board's **J8** connector (see "*Power up the board*").
3. Power up the board.
4. After U-Boot loads, type these commands to load the IFS image:

```
U-Boot> mmc rescan
=> mmc list
FSL_SDHC: 0
FSL_SDHC: 1 (SD)
```

5. Switch to the FSL_SDHC that's identified as the SD slot, and then load your IFS image (in this case, it is 1):

```
=> mmc dev 1
switch to partitions #0, OK
mmc1 is current device
=> fatload mmc 1:1 0x10800000 QNX-IFS
reading QNX-IFS
27874868 bytes read in 1311 ms (20.3 MiB/s)
```

6. Load the IFS:

```
=> go 0x10800000
```

When booting using U-Boot, on the console, you should see the following output:

```
=> go 0x10800000
Welcome to QNX Neutrino 7.0 on the NXP i.mx6 Quad/Dual/DualLite/Solo/QuadPlus Sabre-ARD RevB (AR
Starting watchdog...
Starting I2C 2,3 driver (/dev/i2c2,3)...
Starting SD3 memory card driver...
Starting SD1 (microSD connector) memory card driver on the base board...
Starting RTC utility...
Starting ESAI audio driver...
Starting USB OTG driver
Launching devb-umass...
Starting Ethernet driver
Starting Screen
#
```

The QNX Neutrino RTOS should now be running on your target. If you type `uname` at the prompt, you should see `QNX` returned. You can test the OS by executing any shell command, or any command residing within the OS image (`ls`, `pidin`, etc.).

Boot the board using QNX IPL

After you've loaded the QNX IPL, it automatically loads the QNX IFS. Use the following steps to boot your board using QNX IPL:

1. Insert the SD card into the board's SD slot.
2. Connect the 12V DC power supply to the **main** board's **J1** connector. Alternatively, you can connect the 5V DC power supply to the CPU board's **J8** connector (see "*Power up the board*").
3. Power up the board.

When booting using QNX IPL, you should see the following output in your console:

```
Command:
Press 'D' for serial download, using the 'sendnto' utility
Press 'M' for SDMMC download, IFS filename MUST be 'QNX-IFS'.
SDMMC download...
sdmmc_highspeed:  Switch to high speed mode successful
Load QNX image from SDMMC...
load image done.
Found image               @ 0x18800008
Jumping to startup        @ 0x10807D60
MMFLAGS=1
Welcome to QNX Neutrino 7.0 on the NXP i.mx6 Quad/Dual/DualLite/Solo/QuadPlus Sabre-ARD RevB (ARI
Starting watchdog...

Starting I2C 2,3 driver (/dev/i2c2,3)...
Starting SD3 memory card driver...
Starting SD1 (microSD connector) memory card driver on the base board...
Starting RTC utility...
Starting ESAI audio driver...
Starting USB OTG driver
Launching devb-umass...
Starting Ethernet driver
Starting Screen
#
```

The QNX Neutrino RTOS should now be running on your target. If you type `uname` at the prompt, you should see `QNX` returned. You can test the OS by executing any shell command, or any command residing within the OS image (`ls`, `pidin`, etc.).

Chapter 4
Using the Screen Graphics Subsystem

The default buildfile (**sabreARD.build**) that's provided with this BSP doesn't build an image that starts the Screen Graphics Subsystem (Screen). Instead, you must use the image that's created from the **sabreARD-graphics.build** file, which starts Screen automatically.

The Screen Graphics Subsystem (Screen) includes the libraries, services and tools required to run graphics applications on the board with supported graphics processors, and to manage graphics shown on the display. To use Screen, you often must configure the graphics and display. To confirm whether Screen is running properly on your target, we provide demo applications with this BSP that you can run on the target..

Graphics configuration

The specifics of a board's graphics configuration, such as the target output display port or the output resolution, are defined in the file **graphics.conf**, found on the target in the **/usr/lib/graphics/*board-variant*** directory, where *board-variant* represents the board variant (Solo, DualLite, Dual, Quad) supported by this BSP. You can modify this file at runtime, but you'll need to restart Screen. For example:

```
# slay screen
# screen -c /usr/lib/graphics/iMX6X/graphics.conf
```

Configure the display for Screen

In the **graphics.conf** file, the *video-mode* setting must match the screen resolution and frame rate supported by the display that's connected to your board. Screen reads these parameters from the **graphics.conf** file to initialize the display drivers. These are the resolution and frame rates that the board supports, and these can be found in the **graphics.conf** file for the iMX6X variants of the boards:

- 1024x768 @ 60

For example, for a 1024x768 @60 Hz display, set the `video-mode` in the **graphics.conf** file as shown here:

```
  ...
  ...
begin display internal
    video-mode = 1024x768 @60
end display
...
...
```

For more information about configuring the `display subsection` in the **graphics.conf**, see the "Configure display subsection" section in the *Screen Developer's Guide*.

Display drivers

The video display configuration library that's used is available on the target based on the variant of the board that you have:

Board variant	Location of drivers
iMX6Q and iMX6D (Quad and Dual)	***$QNX_TARGET*/armle-v7/usr/lib/graphics/iMX6X**
iMX6S and iMX6DL (Solo and DualLite)	***$QNX_TARGET*/armle-v7/usr/lib/graphics/iMX6DLS**
iMX6DP and iMX6QP (QuadPlus and DualPlus)	***$QNX_TARGET*/armle-v7/usr/lib/graphics/iMX6DQP**

You only require one libwfdcfg-*.so file at any one time; however, it must match the hardware you are using. These are the display drivers available:

- **libwfdcfg-imx6x-hdmi** (works with most HDMI monitors and it supports multiple resolutions, which can be configured using the `video-mode` parameter;however EDID isn't supported so you can't use this to autoselect the preferred resolution from the display)
- **libwfdcfg-imx6x-hsd100pxn1.so** (works LVDS panels that ship with the NXP Evaluation Kits and with most HDMI monitors; however this driver supports only 1024x768 @ 60 Hz resolution)
- **libwfdcfg-imx6x-innolux.so**
- **libwfdcfg-imx6x-okaya.so**
- **libwfdcfg-lilliput7.so** (for Lilliput 7" displays)
- **libwfdcfg-lilliput10.so** (for Lilliput 10" displays)

Run Screen applications

This BSP archive comes with built-in demo applications that show how to use Screen. It's a good idea to run these demos on your target to confirm that Screen is configured properly and all the necessary drivers are available for Screen to run. If the following demo applications run successfully, it's a good indication that Screen is properly configured on your target.

`sw-vsync`

Shows `vsync` that uses software rendering, but doesn't use GLES. This application is useful to test your display works with basic Screen functionality.

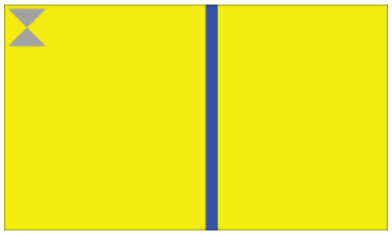


Figure 5: sw-vsync

`gles2-gears`

Shows gears that use OpenGL ES 2.X for the rendering API; if `gles2-gears` runs, then it confirms that `screen` and drivers necessary for OpenGL ES have started successfully.

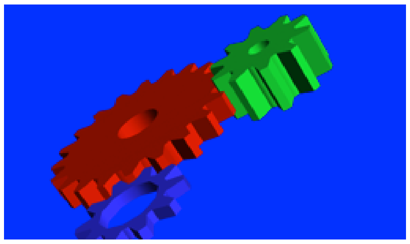


Figure 6: gles2-gears

For more information about using the demo applications or debugging them, see the "Utilities and Binaries" and the "Debugging" chapters in the *Screen Developer's Guide*, respectively.

Create Screen applications

When you create an application that supports Screen, it must link to the Screen library (**libscreen.so**). This library provides the interfaces for applications to use the Screen API. Depending on the graphical requirements of your application, you may need to link in additional libraries. Below are some of the libraries commonly used when building applications that use Screen:

libEGL

The interface to access vendor-specific EGL functionality. Refer to the EGL standard from *www.khronos.org*.

libGLESv2

The interface to access vendor-specific OpenGL ES functionality. Refer to the OpenGL ES standard from *www.khronos.org*.

libimg

The interface to load and decode images. For more information, see the *Image Library Reference* guide in the QNX Software Development Platform 7.0 documentation.

For more information about application development using Screen, see the *Screen Developer's Guide*. For information about linking libraries, see "QNX C/C++ Project properties" in the "Reference" chapter of the *QNX Momentics IDE User's Guide* or the "Compiling and Debugging" chapter of the *QNX Neutrino Programmer's Guide*.

Chapter 5
Using Multitouch

If you have a multitouch display, you can use it with this board.

Multitouch requires that you have Screen running. For more information about using Screen, see *Using the Screen Graphics Subsystem*. To use touch, your target must have:

- the necessary drivers based on the touch controller for your hardware. The default buildfile provided for this board is configured to use touch displays that have an Egalax I2C-based touch controller. In the buildfile, you must include the following:
  - **libmtouch-calib.so.1** library (driver)
  - **libmtouch-egalax.so** library (driver)
  - **libinputevents.so** library
  - **libkalman.so** library
  - `mtouch` binary

  If you want to use a USB-based touch controller for the display (e.g., Lilliput), see "*Configuration changes for USB-based touch controllers*." For information about other supported touch controllers, see the "`mtouch`" section in the "Utilities and Binaries" chapter of the *QNX Developer's Guide*.

- the touch and scaling configuration files. For information about these configuration files, see "*Modifying the touch configuration file*" and "*Modifying the scaling configuration file*." in this chapter.

Modifying the touch configuration file

The touch configuration file is typically located at **/etc/system/config/mtouch.conf** on the target.

You can modify the file and build it into your image. For this board, your options should be set to `lvds=1,protocol=0`, which are specific to the touch controller hardware used on the display. Here's what the touch configuration file for an Egalax controller looks like:

```
begin mtouch
    driver = egalax
    options = lvds=1,protocol=0
end mtouch
```

For more information about the parameters used for the `mtouch` configuration file, see the "`mtouch`" section in the "Utilities and Binaries" chapter of the *QNX Developer's Guide*.

In the above configuration, because the `width` and `height` settings of the touch matrix aren't specified, they default to 32768 and 32768, respectively. Don't confuse this with the dimensions that you set for the display's resolution.

Modifications to the scaling configuration file

The touch configuration file is typically located at **/etc/system/config/scaling.conf** on the target.

You can modify this board-specific version of the **scaling.conf** file. The file is used to map touch matrix coordinates.For more information about other scaling modes you can use, see the “Scaling configuration file” section in the “Utilities and Binaries” chapter of the *QNX Developer's Guide*. For example, here's an example of using direct mapping:

```
 # i.MX6 SabreARD
1024x768:mode=direct
```

Configuration changes for USB-based touch controllers

If you have a touch display that's connected using USB, you may want to use the Human Interface Devices (HID) driver instead.

To use the HID driver, attach a display with a USB-enabled touch controller, such as a Lilliput 10" display unit. To enable touch, ensure that the USB drivers are started before you run the `mtouch` command.

To change your configuration to use HID driver, you set the `driver` parameter to `hid` in the touch configuration file (**mtouch.conf**). Your file should be modified to look like this:

```
begin mtouch
    driver = hid
    options = vid=0x0eef,did=0x0001,verbose=6,width=4096,height=4096,max_touchpoints=1
end mtouch
```

For more information about using HID, see the “`mtouch`” section in the *QNX Developer's Guide*.

Chapter 6
Using Audio Input and Output

You can change the audio input and output of the board.

By default, audio will play (output) on "LINE OUT", and record (input) from MIC. The following commands show how to see the status of the headphone output and how to switch the output to the headphone:

```
# mix_ctl switches
"Headphone Select"                    BOOLEAN   off
# mix_ctl switch "Headphone Select" ON
"Headphone Select"  BOOLEAN  on
```

The following command shows how to switch the input to "LINE IN":

```
# mix_ctl switches
"Headphone Select"                    BOOLEAN   on
# mix_ctl group "Line In" capture=on
```

Chapter 7
Build the BSP

You can use the QNX Momentics IDE or the command line on Linux, macOS, or Windows to build an image.

Generic instructions to modify your BSP (add contents and modify the buildfile) can be found in the *Building Embedded Systems* guide, which is available as part of the QNX SDP 7.0 documentation.

For instructions on how to build this BSP using the IDE, see "*Build the BSP (IDE)*" section in this chapter. For detailed information on how to build using the IDE, see the *IDE User's Guide*, which is available as part of the QNX SDP 7.0 documentation.

If you plan to work with multiple BSPs, we recommend that you create a top-level BSP directory and then subdirectories for each different BSP. The directory you create for a specific BSP will be that BSP's root directory (*BSP_ROOT_DIR*). This allows you to conveniently switch between different BSPs by simply setting the *BSP_ROOT_DIR* environment variable.

This BSP includes prebuilt IFS images that are provided as a convenience to quickly get QNX Neutrino running on your board, however these prebuilt images might not have the same components as your development environment. For this reason, we recommend that you rebuild the IFS image on your host system to ensure that you pick up the same components from your own environment.

Prebuilt image

After you've unzipped the BSP, a prebuilt QNX IFS image is available in the BSP's **/images** directory. This prebuilt IFS image (**ifs-mx6x-sabreARD.bin**) can be regenerated with the BSP `make` file, and is configured for the BSP device drivers already available for your board.

If you modify and build the IFS, the original IFS files are overwritten. If you need to revert to this prebuilt image, simply run the `make` command from the BSP's root directory using the original buildfiles. To determine the location of the buildfile to modify, open the ***$BSP_ROOT_DIR*/source.xml** file.

If you forget to make a backup copy of the IFS file, you can still recover the original prebuilt IFS; simply extract the BSP from the **.zip** archive into a new directory and get the prebuilt image from that directory.

Considerations before building the BSP

By default, the buildfile creates an uncompressed IFS that can be loaded with U-Boot or QNX IPL. If want to use a compressed IFS, you can boot using only QNX IPL. To build a compressed IFS, you must modify the buildfile before you build your BSP. To do this, add `+compress` after the `raw` option to your buildfile. For example:

```
[image=0x10800000]
[virtual=armle-v7,raw] .bootstrap = {
```

Change them to:

```
[image=0x10800000]
[virtual=armle-v7,raw +compress] .bootstrap = {
```

Using graphics (Screen Graphics Subsystem) in the image

To build the image with graphics (Screen Graphics Subsystem or simply Screen), use the `ifs-mx6x-sabreARD-graphics.bin` target. The corresponding buildfile to include graphics in the image you build is located at ***$BSP_ROOT_DIR*/images/sabreARD-graphics.build**.

Build the BSP (command line)

You can use the command line on Linux, macOS, or Windows to work with this BSP.

Makefile targets in the BSP

The **Makefile** is located under the ***$BSP_ROOT_DIR*/images** directory. It defines more than one target:

`all`

Invokes all targets.

`ifs-mx6x-sabreARD.bin`

Builds the IFS file for the target.

`ifs-mx6x-sabreARD-graphics.bin`

Builds the IFS file with Screen Graphics Subsystem for the target.

`ipl-mx6x-sabreARD.bin`

Builds the QNX IPL file for the target.

If you don't specify a target, `make` invokes the `all` target.

Build from the command line

To build the BSP on your host system, in a Command Prompt window (Windows) or a terminal (Linux, macOS), you must first build at the root directory of the BSP (*BSP_ROOT_DIR*), then you can build using the Makefile targets described previously in the ***$BSP_ROOT_DIR*/images** directory. To build your BSP, perform the following steps:

1. Download the BSP archive, unzip it, and set up your environment to build. For more information, see "*Download and set up the BSP*."
2. In the BSP's root directory (*BSP_ROOT_DIR*), type `make clean` to remove the default image files from the ***$BSP_ROOT_DIR*/images** directory.

   ```
   cd $BSP_ROOT_DIR
   cd images
   make clean
   ```

   This action removes all existing image files.
3. Navigate back to the *BSP_ROOT_DIR* and type `make`. Running `make` does the following:
   - builds the BSP and creates its directories
   - generates the default buildfiles, which are copied from **prebuilt** to the **install** directory. A copy of the buildfile is placed in the **images** directory. As a convenience, you can modify the buildfile located in the **images** directory and run the Make targets listed previously.
   - places any images required for the board into the ***$BSP_ROOT_DIR*/images** directory

```
cd $BSP_ROOT_DIR
make
```

You should now have an IFS file called **ifs-mx6x-sabreARD.bin**.

After you build, there might be multiple IFS files in the ***$BSP_ROOT_DIR*/images** directory if your **Makefile** builds multiple IFS files.

We recommend that you use the `make` command to build your IFS image. If you use the `mkifs` utility directly, ensure that you use the `-r` option to specify the location of the binaries.

Build the BSP (IDE)

You can use the QNX Momentics IDE on Linux, macOS, or Windows, to work with this BSP.

Build in the IDE

You can build the entire BSP in the QNX Momentics IDE, which includes building the source and the IPL and IFS images.

1. If you haven't done so, launch the IDE, then import the BSP archive (which was downloaded for you using the QNX Software Center) into the IDE (see "*Import to the QNX Momentics IDE*" for details).
2. If you haven't already, switch to the C/C++ Perspective, then in the Project Explorer view, right-click the BSP source project (such as **bsp-nxp-imx6x-sabreARD**) and select **Build Project**.

   This step builds the entire BSP, which includes the images and the binaries required for the images. You can check the progress and outcome of your build in the Console view.
3. If you want to modify the buildfile, open the **.build** file under the **System Builder Files** folder. After you've made your changes, right-click the BSP source project and select **Build Project**.

Build changes to the buildfile

You can make changes to the buildfile(s) in the **images** folder.

CAUTION: Changes you make to the buildfile in the **images** folder are overwritten if you build the entire BSP Project as described in the previous section. Ensure that you save a backup copy of the buildfile elsewhere.

To build the changes you've made to a buildfile, create a *make target* in the **images** folder and then map it to an existing make target in the **Makefile** located in the **images** folder, which is equivalent to running the `make` command from the **images** directory.

For more information about creating make targets, see **Tasks** → **Building Projects** → **Creating a make target** in the *C/C++ Development User Guide* in the IDE Help.

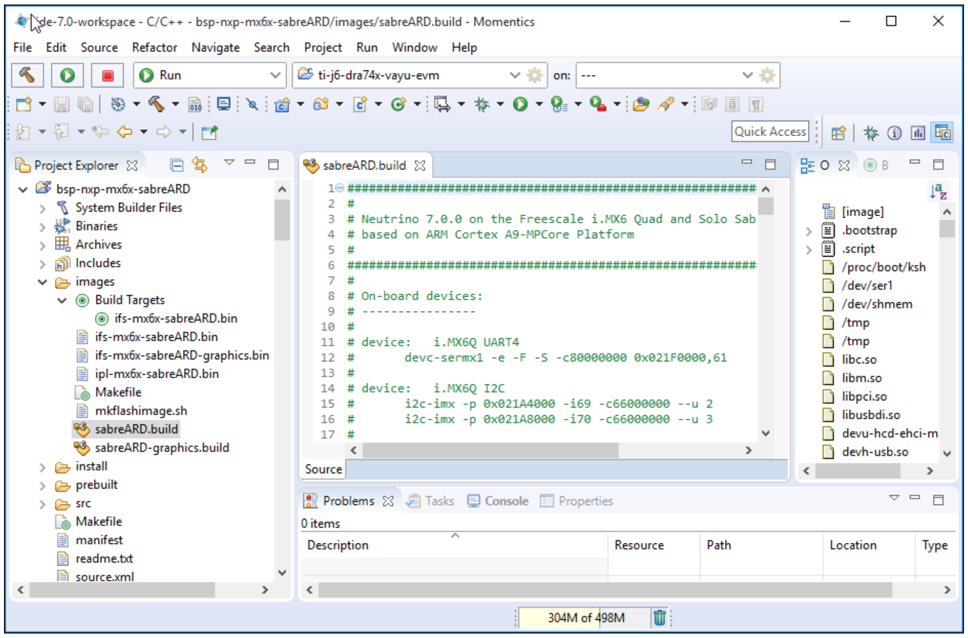
1. In the Project Explorer view, expand the **images** folder, right-click the IFS file you want recreated, and select **Delete**. The IFS must be deleted or it won't be rebuilt.
2. In the example view, right-click the **images** folder and select **Build Targets** → **Create...**.
3. In the **Create Build Target** dialog box, type the name of an existing target in the **Makefile**, and then click **OK**.

   For example, if you wanted to build the default IFS file, **ifs-mx6x-sabreARD.bin**, it maps to the ifs-mx6x-sabreARD.bin target in the Makefile, you would type `ifs-mx6x-sabreARD.bin` as the target. For more information on the targets available, see the "*Build the BSP (command line)*" section.

   It's a good idea to create `clean` and `all` build targets, which run the `make clean` (to remove all created IFS and IPL files in the images folder) and `make all` (to rebuild all the IFS and IPL files) commands, respectively.

4. In the Project Explorer view, under the **images** folder, you should see **Build Targets** created. Right-click **Build Targets**, select the build target that you created in the previous step, and click **Build**.

After your image builds, you should see it appear under the **images** folder in the IDE as shown in the figure below:



```
###############################################################
#
# Neutrino 7.0.0 on the Freescale i.MX6 Quad and Solo Sab
# based on ARM Cortex A9-MPCore Platform
#
###############################################################
#
# On-board devices:
# ----------------
#
# device:   i.MX6Q UART4
#        devc-sermx1 -e -F -S -c80000000 0x021F0000,61
#
# device:   i.MX6Q I2C
#        i2c-imx -p 0x021A4000 -i69 -c66000000 --u 2
#        i2c-imx -p 0x021A8000 -i70 -c66000000 --u 3
#
```

Figure 7: Build image in the IDE

Build the QNX IPL image

The QNX Initial Program Loader (IPL) can be built separately using a script.

The IPL file is provided as a prebuilt file and is rebuilt when you build the entire BSP.

To build the IPL file separately, you can run a script included with your BSP called `mkflashimage.sh`. The `mkflashimage.sh` script generates the **ipl-mx6x-sabreARD.bin** used to boot your board.

1. On your host, in a Command Prompt window (Windows) or a terminal (Linux, macOS), navigate to the ***$BSP_ROOT_DIR*/src/hardware/ipl/boards/mx6x-sabreARD/arm/le.v7/** directory and run `make clean` and then `make`.

```
$ cd $BSP_ROOT_DIR/src/hardware/ipl/boards/mx6x-sabreARD/arm/le.v7/
$ make clean
$ make
```

2. Navigate to the ***$BSP_ROOT_DIR*/images** directory and then run the `mkflashimage.sh` command. This command is necessary to create the IPL files:

```
$ cd $BSP_ROOT_DIR/images
$ sh mkflashimage.sh
```

After you complete the steps successfully, the **ipl-mx6x-sabreARD.bin** file (for booting using a SD card) is found in the ***$BSP_ROOT_DIR*/images** directory.

Chapter 8
Driver Commands

The tables below provide a summary of driver commands.

Some of the drivers are commented out in the buildfile provided in the **startup** directory of this BSP. To use these drivers on the target hardware, you may need to uncomment them in the buildfile, rebuild the image, and load the image onto the board.

The order that the drivers are started in the provided buildfile is one way to start them. The order that you start the drivers is completely customizable and is often specific to the requirements of your system. For example, if you need an audible sound to play immediately, you must start the audio driver earlier than other drivers. It's important to recognize that the order you choose impacts the boot time.

Some drivers depend on other drivers, so it's sometimes necessary to start them in a specific order because of these dependencies. For example, in the provided buildfile, the USB driver is started before the mass storage drivers because they rely on it.

- For more information on best practices for building an embedded system and optimizing boot times for your system, see the *Building Embedded Systems* and the *Boot Optimization* guides in the QNX Software Development Platform 7.0 documentation.
- For more information about the drivers and commands listed here, see the QNX Neutrino *Utilities Reference*. This chapter provides information about BSP-specific options for any of the commands and drivers that aren't described in the *Utilities Reference*.

Here's the list of drivers available for this board and the order they appear in the provided buildfile:

- *Startup*
- *Watchdog*
- *Serial*
- *Inter-integrated Circuit (I2C)*
- *SD/MMC*
- *SPI NOR flash*
- *Controller Area Network (CAN)*
- *Real-time Clock (RTC)*
- *Audio*
- *PCI server*
- *Enhanced SPI*
- *USB host controller (`io-usb_otg` stack)*
- *USB device controller (`io-usb_otg` stack)*
- *Network*
- *Parallel*
- *SATA*
- *PMIC*

- *Graphics*
- *Touch*

Audio

Device	AUDIO
Command	`io-audio -vv -d mxesai-mx6sabreARD`
Required binaries	`io-audio`, `mix_ctl`, `wave`, `waverec`
Required libraries	**deva-ctrl-mxesai-mx6sabreARD.so**, **libasound.so**
Source location	**src/hardware/deva/ctrl/mxesai**. The codecs are available at **src/hardware/deva/ctrl/codecs**.

Controller Area Network (CAN)

Device	CAN (i.MX6Q FlexCAN)
Command	`dev-can-mx6x -n 500 can0`
Required binaries	`dev-can-mx6x`, `canctl`, `canutil`
Required libraries	N/A
Source location	**src/hardware/can/mx6x**

Enhanced SPI

Device	Enhanced SPI driver (i.MX6Q SPI 1,2,3,4,5)
Command	`spi-master -u1 -d mx51ecspi base=0x02008000,irq=63,loopback=1` (starting **/dev/spi1**) `spi-master -u2 -d mx51ecspi base=0x0200C000,irq=64,loopback=1` (starting **/dev/spi2** `spi-master -u3 -d mx51ecspi base=0x02010000,irq=65,loopback=1` (starting **/dev/spi3** `spi-master -u4 -d mx51ecspi base=0x02014000,irq=66,loopback=1` (starting **/dev/spi4** `spi-master -u5 -d mx51ecspi base=0x02018000,irq=67,loopback=1` (starting **/dev/spi5**
Required binaries	`spi-master`
Required libraries	**spi-mx51ecspi.so**
Source location	**src/hardware/spi/mx51ecspi**

The parameters to the `spi-master` command must be adjusted to the target hardware.

The SPI NOR Flash (`devf-norapi-mx6-sabreARD-ecspi`) driver must not be running at the same time as the ECSPI (`spi-master`) driver.

Options:

`-u` ***unit***

Set the SPI unit number. The default is 0.

`-d` ***modulename***

The driver module name.

`-P` ***permissions***

Set the permissions for the device. The default permissions are 0666 (non-directory user: read/write, group: read/write, and other: read/write.

`base=`***address***

the base address of SPI controller. The default 0x70010000.

`burst=`***mode***

Indicate which mode to use for SPI transfers.

Set *mode* to 1 (default mode) to support 8, 16, 32-bit words; otherwise set to 0 to set to word mode, which supports any word length (from 1- to 32-bits.

`clock=`***num***

Enhanced SPI clock. The default is 66500000 Hz.

`errata=`***support***

Support ERRATA Engcm09397. Set to 1 to enable ERRATA support (default); otherwise set to 10 to disable ERRATA support.

`gpiocsbase=`***address***

GPIO base address of the GPIO Salve Select pin.

`gpiocs0=`***pinnum***

Pin number for the SS0 pin.

`gpiocs1=`***pinnum***

Pin number for the SS1 pin.

gpiocs2=*pinnum*

GPIO pin number for the SS2 pin.

gpiocs3=*pinnum*

GPIO pin number for the SS3 pin.

`irq=`*num*

IRQ of the interface. The default is 14.

`loopback=`*enable*

Set the internal loopback for testing. The default is 0 (loopback disabled). Set to 1 to enable loopback testing.

`verbose=`*num*

Verbosity of logging. Default is 0 for no logs.

`waitstate=`*num*

Set the number of waitstates between transfers. The default is 0.

Graphics

Device	GRAPHICS
Command	`screen`
Required binaries	`screen`, `gles1-gears`, `gles2-gears`, `sw-vsync`
Required libraries	Refer to the *Screen Developer's Guide* in the QNX SDP 7.0 documentation.
Source location	**src/hardware/wfd/imx6x/wfdcfg/imx6x-hdmi**

Inter-integrated Circuit (I2C) (2 interfaces)

Device	I2C
Command	`i2c-imx -p 0x021A4000 -i69 -c66000000 --u 2`
Command	`i2c-imx -p 0x021A8000 -i70 -c66000000 --u 3`
Required binaries	`i2c-imx`
Required libraries	N/A
Source location	**src/hardware/i2c/imx**

Network

Device	NETWORK

Command	`io-pkt-v6-hc -dmx6x -p tcpip random`
	`if_up -p fec0`
	`ifconfig fec0 up`

	`dhclient -nw fec0`
Required binaries	`io-pkt-v6-hc`, `if_up`
Required libraries	**devnp-mx6x.so**, **libsocket.so**, **libz.so**, **libssl.so**, **libcrypto.so** **libnbutil.so**
Source location	**src/hardware/devnp/mx6x**

Parallel (linear) NOR Flash

Device	Linear NOR Flash
Command	`devf-generic -s0x08000000,32M -m /fs/nor`
Required binaries	`devf-generic`
Required libraries	
Source location	**src/hardware/flash/boards/generic**

To use the linear (parallel) NOR Flash driver, make sure that:

- SPI NOR is *not* running.
- `screen` is *not* running.
- You specify the `-n1` in the `startup` command line.

PCI Server

Device	PCI
Command	`pci-server -c -v`
Required binaries	`pci-server`, `pci-tool`
Required libraries	**pci_hw-iMX6-sabreard.so**, **pci_bkwd_compat.so**, **pci_debug.so**, **pci_slog.so**
Source location	Binary only

PMIC

Device	PMIC
Command	`pmic-pmpf0100 -b2 -i304 -v5 -a 0x08`
Required binaries	`pmic-pmpf0100`
Required libraries	N/A

Source location	**src/hardware/support/pmic_pmpf0100**

Real-time clock (RTC)

Device	RTC
Command	`rtc hw`
Required binaries	`rtc`, **date**
Required libraries	
Source location	**N/A**

SATA

Device	SATA (only i.MX6QP support)
Command	`devb-ahci ahci ioport=0x02200000,irq=71`
Required binaries	`devb-ahci`
Required libraries	N/A
Source location	**Binary only**

Serial

Device	SERIAL (UART)
Command	`devc-sermx1 -e -F –S -c80000000 0x021F0000,61`
Required binaries	`devc-sermx1`
Required libraries	N/A
Source location	**src/hardware/devc/sermx1**

SD/MMC

Device	MMCSD

Command	`devb-sdmmc-mx6 cam pnp,verbose blk rw,cache=2M sdio addr=0x02198000,irq=56,bs=cd=0x020b0000^15^335:wp=0x0209c000^13^173 disk name=sd3` `devb-sdmmc-mx6 cam pnp,verbose blk rw,cache=2M sdio addr=0x02190000,irq=54,bs=cd=0x0209c000^1^161:wp=0x020ac000^20^309 disk name=sd1` (microSD connector on the base board)
Required binaries	`devb-sdmmc-mx6`
Required libraries	**libcam.so**, **cam-disk.so**, **io-blk.so**, **fs-qnx6.so**
Source location	**src/hardware/devb/sdmmc**

SPI NOR Flash

Device	SPI NOR Flash (SST 25VF016B SPI Flash Chip)
Command	`devf-norspi-mx6_sabreARD-ecspi`
Required binaries	`devf-norspi-mx6_sabreARD-ecspi`, `flashctl`
Required libraries	N/A
Source location	**src/hardware/flash/boards/norspi**

To use the SPI NOR Flash driver, make sure that:

- The J3 jumper is set for an SPI NOR boot (pins 2-3). If no bootloader is found in the SPI NOR Flash, the board defaults back to boot from the SD card.
- The **spi-master** resource manager is *not* running.
- You specify the `-n2` option in the `startup` command line.
- To format the SPI Flash chip, run the following command:

  ```
  flashctl -p /dev/fs0p0 -e -f -m
  ```

  After you format a partition, it appears in **/fs0p0**.

Startup

Device	STARTUP
Command	`startup-imx6x-sabreARD -n1 -m -W -s`
Required binaries	`startup-imx6x-sabreARD`
Required libraries	N/A
Source location	**src/hardware/startup/boards/imx6x**

Here are other parameters that you can use. In some cases, some of these parameters may conflict with other features on the board.

-b

Enable Bluetooth on the board. Using Bluetooth conflicts with the SPI NOR and PCIe.

-c

Enable CAN, however this conflicts with Ethernet.

-n0

Don't use NOR Flash. This enables I2C3.

-n1

Use Parallel NOR Flash. This disables I2C3.

-n2

Use SPI NOR Flash. This disables I2C3.

-r

Reserve the top 256 MB of RMA (`0x80000000 - 0x8FFFFFFF`) so that the Screen Graphics Subsystem works properly. This is required for graphics processor on the boards that use the Quad Plus because code won't run unless it's allocated within this range of memory. This is a hardware-specific error that must be resolved by the hardware vendor.

For example: `startup-imx6x-sabreARD -n0 -m -r 0x80000000,256M`

-s

Use SDMA to load the IFS image.

Touch

Device	Lilliput FA1012-NP/C/T 10.1” LCD

Command	`mtouch`
	`io-hid -dusb devi-hid -R1024,768 touch`
	You must start the USB driver before you start this driver.

Required binaries	`mtouch`
Required configuration files	**/etc/system/config/mtouch.conf**, **/etc/system/config/scaling.conf**
Required libraries	**libmtouch-calib.so.1**, **libinputevents.so**, **libkalman.so**, **libmtouch-hid.so** (HID and Egalax USB-based touch displays), **libmtouch-egalax.so**(Egalax I2C-based touch displays)
Source location	Binary only

USB OTG host controller (`io-usb-otg` stack)

USB host controller (J30) and OTG controller (J10); host mode uses `io-usb-otg` stack.

Device	USB host

Command	`io-usb-otg -d hcd-ehci-mx28 ioport=0x02184100,irq=75,phy=0x020c9000,no_stream, verbose=5,ioport=0x02184300,irq=72,phy=0x020ca000, no_stream,verbose=5`
	 `waitfor /dev/usb/io-usb-otg 4`

Required binaries	`io-usb-otg`, `usb`
Required libraries	**devu-hcd-ehci-mx28.so**
Source location	Binary only

USB OTG device controller (`io-usb-otg` stack)

USB OTG controller

Device	USB device

Command (Mass storage device)	`io-usb-otg -d dcd-usbumass-mx6sabrelite-ci ioport=0x02184000,irq=75 -n /dev/io-usb-dcd/io-usb`
	 `waitfor /dev/io-usb-dcd/devu-usbumass-omap543x-dwcotg3.so 4`

Command (CDC-ACM serial device)	`io-usb-otg -d dcd-usbser-mx6sabrelite-ci ioport=0x02184000,irq=75 -n /dev/io-usb-dcd/io-usb`
Required binaries	`io-usb-otg`, `usb`
Required libraries	**libusbdci.so**, **devu-dcd-usbumass-mx6sabrelite-ci.so**, **devu-dcd-usbser-mx6sabrelite-ci.so**, **devu-dcd-usbncm-mx6sabrelite-ci.so**
Source location	Binary only

Example for a mass storage device

```
# Create a RAM disk
devb-ram ram capacity=16384,nodinit,cache=512k disk name=hd@10
waitfor /dev/hd10
fdisk /dev/hd10 add -t 6
mount -e /dev/hd10
waitfor /dev/hd10t6
mkdosfs /dev/hd10t6
mount -tdos /dev/hd10t6 /dos

# Start the device stack
```

```
io-usb-otg -d dcd-usbumass-mx6sabrelite-ci ioport=0x02184000,irq=75 -n /dev/io-usb-dcd/io-usb
waitfor /dev/io-usb-dcd/io-usb 4
waitfor /dev/io-usb-dcd/devu-usbumass-mx6sabrelite-ci  4

#Start the mass storage function driver and enable USB soft connect
devu-umass_client-block -l lun=0,devno=1,iface=0,fname=/dev/hd10
ulink_ctrl -l 1
```

Example for creating a CDC-ACM (serial device)

```
# Start the USB device stack
io-usb-otg -d dcd-usbser-mx6sabrelite-ci ioport=0x02184000,irq=75 -n /dev/io-usb-dcd/io-usb
waitfor /dev/io-usb-dcd/io-usb 4
waitfor /dev/io-usb-dcd/devu-usbser-mx6sabrelite-ci.so 4

# Start the USB CDC-ACM function driver and enable USB soft-connect
devc-serusb_dcd -e -v -F -s -d iface_list=0,unit=1
waitfor /dev/serusb1
ulink_ctrl -l 1
```

Watchdog utility

Device	WATCHDOG
Command	`wdtkick`
Required binaries	**wdtkick**
Required libraries	
Source location	**src/hardware/support/wdtkick**