BSP User's Guide

Texas Instruments

Jacinto 6 (DRA74x and DRA75x) EVM

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 SDP 7.0 Board Support Package (BSP) for Texas Instruments Jacinto 6 Vayu (DRA74x/DRA75x) EVM and the Sitara AM572x EVM boards.

For convenience, this document refers to the Texas Instruments Jacinto 6 Vayu DRA74x/DRA75x) EVM as “Jacinto 6 EVM” and Texas Instruments Sitara AM572x EVM as the “AM572x EVM.” This BSP also supports the Beaglebone-X15 Development Board, which is based on the processor as the AM57x. If you want to use the Beaglebone-X15 Development Board, use the steps for the “AM572x EVM” from this guide.

This documentation includes:

- an overview of QNX BSPs
- information about how to build QNX embedded systems, which you should read before you begin working with this BSP
- what's new in the BSPs for this release
- structure and contents of a BSP
- how to prepare a bootable microSD card
- how to boot the board

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 operating systems, and boards	*About This BSP*
Installing this BSP	*Installation Notes* It's important to note that based on the board that you choose, that there are different steps.
Using the Screen Graphics Subsystem on the boards	*Configure the Screen Graphics Subsystem on the Jacinto 6 EVM and AM572x EVM*
Using Audio on the Jacinto 6 EVM	*Use the Multichannel Audio Serial Port on the Jacinto 6 EVM*
Using USB device mode on the Jacinto 6 EVM	*Using USB device mode on the Jacinto 6 EVM*
Building this BSP	*Build the BSP* It's important to note that based on the board that you choose, that there are different steps.

To find out about:	See:
Driver commands for the boards	*Driver Commands for the Jacinto 6 EVM* and *Driver Commands for the AM572x*
Using secure boot (Jacinto 6 EVM only)	*Secure Boot Support on the Jacinto 6 EVM*

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 when building QNX embedded systems.

Expansion boards

The Jacinto 6 EVM consists of a Vayu CPU board and up to three optional daughter boards. These daughter boards run as slaves to the Jacinto 6 Vayu board. They can be:

- a JAMR3 application board
- a Vision application board
- a supported board for the LCD display

For Jacinto 6 EVM revisions G onwards, the LCD support is for a 10.1-inch LCD/TS(recommended), as well as for the 7-inch screen

The AM572x EVM platform consists of a CPU board and up to one additional LCD daughter board. The attached LCD board runs as a slave to the AM572x EVM board.

Additional documentation

Before you start to build and install your BSP, you should review the following documentation:

- Information about your board's hardware and firmware, which can be found in the following documents from Texas Instruments:
  - *Vayu EVM CPU Board User's Guide*
  - *DRA7xx Infotainment Applications Processor Silicon Revision 1.0 Technical Reference Manual*. References to this document are to Version G
  - Texas Instruments AM572x EVM hardware and firmware at *http://processors.wiki.ti.com/index.php/AM572X_EVM_Boards*
- *Building Embedded Systems* guide, which contains general information about working with BSPs from QNX Software Systems that's common to all BSPs and is available from QNX Software Development Platform 7.0 documentation

Required software

To work with the BSP for the Texas Instruments Jacinto 6 Vayu (DRA74x/DRA75x) EVM and Sitara AM572x EVM board, you require the following:

- the 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. Both the Jacinto 6 EVM and AM572x boards have an embedded serial-USB conversion chip. To be able to use this chip, you need to install an FTDI driver on your

host system, and connect to the board via the board's mini-USB AB connector (console USB port). This driver must be installed on your 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, Hyperterminal, PuTTy, etc.) to connect to the board for debugging. Alternatively, you can use a `telnet` or `ssh` session from the QNX Momentics IDE.

Additional tools

For the Jacinto 6 EVM, If you want to use USB booting to send an IFS image from your host system directly to the target to boot, you can use the Fastboot utility, which is available as part of the Android SDK. For more information, see *https://wiki.cyanogenmod.org/w/Doc:_fastboot_intro*.

Chapter 2 About This BSP

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

What's in this BSP

This BSP contains the following components:

Driver	Format	Comments
IPL	Source	To boot from microSD card or QSPI NOR Flash
Startup	Source	
Kernel	Binary	
RTC utility	Source	Real time clock utility
Watchdog kick utility	Source	
Serial driver	Source	
SPI master	Source	
QSPI	Source	Driver for S25FL256SAGMFV001 SPI NOR Flash available for Jacinto 6 EVM only
I2C	Source	
Audio (MCASP)	Source	
Audio for JAMR3 (MCASP6)	Source	Available for Jacinto 6 EVM only
HS MMC/SD	Source	
SLC NAND	Source	Available for Jacinto 6 EVM only
SATA	Binary	
USB (Host mode)	Binary only	
USB (Device mode)	Binary only	
Ethernet	Source	
Screen	Source	Delivered with SDP 7.0
Touch screen	Binary only	Available for Jacinto 6 EVM only

Driver	Format	Comments
Input	Binary only	

Supported host operating systems

We recommend that you install and work from one of the following on your host system:

- Linux Red Hat Enterprise Linux 7 64-bit or Ubuntu Workstation LTS 64-bit
- Apple macOS OSX 10.10.x 64-bit
- Microsoft Windows 10 Professional 64-bit, Windows 8 64-bit, Windows 7 64-bit

This BSP supports 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. For the AM572x EVM (and Beaglebone X-15), it's found at **src/hardware/startup/boards/dra74x/am572x-evm/build** in the BSP. For the Jacinto 6 EVM, it's found at **src/hardware/startup/boards/dra74x/vayu-evm/build** in the BSP. These interrupt tables are 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 build, install and start this BSP.

Here's the process to install the BSP:

1. Download the BSP,setup your environment, and extract the BSP. For more information, see "*Download and set up the BSP*."
2. Set up your target board. After you connect the hardware, it may be necessary set DIP switches on your board to configure it. For more information about how to connect to your board, see "*Set up the hardware*."
3. Create a bootable microSD card before your can use the BSP images. After you create a bootable microSD 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 microSD card, see "*Prepare a bootable microSD card*."
4. If this is the first time you are bringing up the BSP, you can transfer the provided IFS image file to the microSD to boot your board U-Boot. To get U-Boot, you must it from the vendor of the Jacinto 6 EVM and AM572x EVM.

   Alternatively, you can use the QNX IPL to load the IFS file. For information about preparing your boot image, see "*Transfer images to the board*." You can copy the image to an eMMC or QSPI NOR Flash device if your board supports it. During the build process, you may need to update configuration files or change the files that are added to your image, depending on the revision of the board you use. See the "*Driver Commands for the Jacinto 6 EVM*" or *Driver Commands for the AM572x*. If you want to configure graphics, see "*Configure the Screen Graphics Subsystem on the Jacinto 6 EVM and AM572x EVM*" for more information.
5. After you put the images on the card, you can boot it. The process to boot the board differs based on whether you chose to use U-Boot or QNX IPL. For more information about booting your board, see "*Boot the board*."

After you boot the board, you can use that image to understand how it works. Depending on the other devices you've 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 generic information about modifying and building an 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/r-car-h3**). 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_ti-j6_dra74x_vayu-evm_beta_WBN***build_ID***.zip**

   where *build_ID* is the BSP build number (e.g., `WBN15`).

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., **ti-j6-dra74x-vayu-evm**). 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:

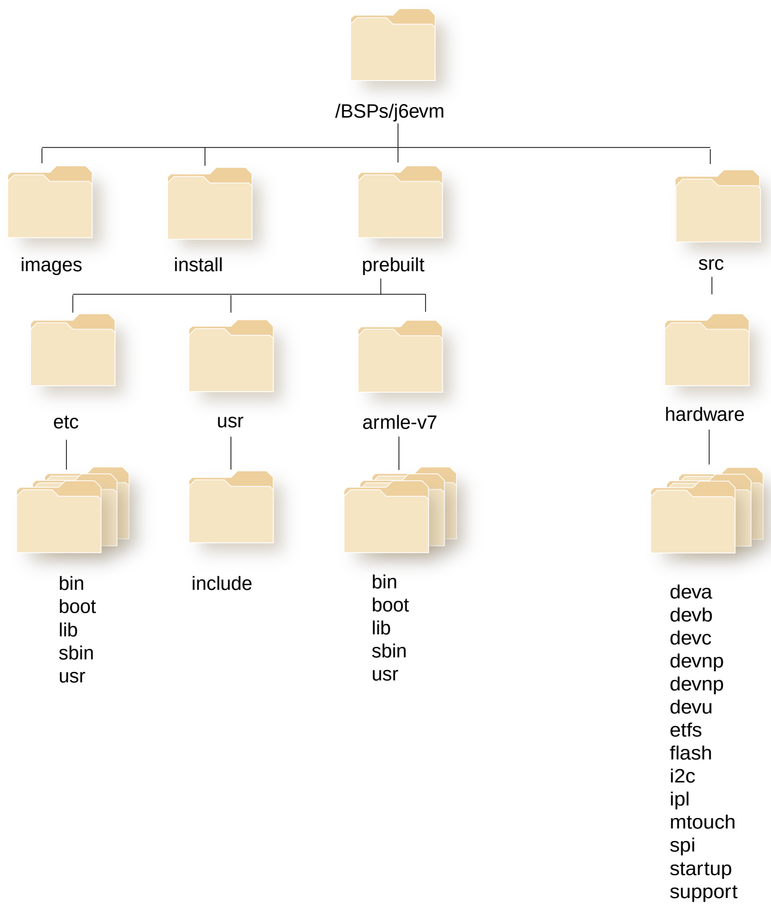


Figure 1: Structure of a BSP

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

Attach the USB and other cables to the appropriate hardware connectors.

Depending on the board you're using and the features on the board, you may require different connectors and different DIP switch settings. For information about the Jacinto 6 EVM, see the "*Set up the Jacinto 6 EVM*" section. For information about the AM572x EVM, see the "*Set up the AM572x EVM*" section.

Set up the Jacinto 6 EVM

These are the connectors that you must connect for the Jacinto 6 EVM.

Connectors for the Jacinto 6 EVM

The figure below identifies the following connectors, ports and slots on the Jacinto 6 EVM, revisions E1 and E2. Locations may differ on other board revisions.

1. 12V DC power supply connector
2. boot microSD card slot
3. console USB port (serial port)
4. USB 2.0 host port
5. USB OTG port

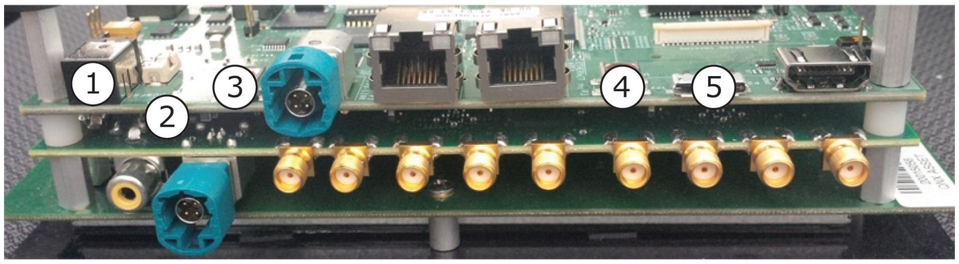


Figure 2: Ports on the Jacinto 6 EVM.

Before you power up your board, you need to connect the 12V DC power supply. Depending on what you'll be doing, you may also want to connect an Ethernet cable connected to your network. Connecting a Ethernet cable is useful to transfer files to your target board.

Connect to the Jacinto 6 EVM

You require a USB serial device driver on your host development machine in order to establish a terminal session with the target over USB.

The Jacinto 6 EVM has an embedded serial-USB conversion chip. To be able to use this chip, you need to install an FTDI driver on your host system, and connect to the Jacinto 6 EVM target board via the board's J1 mini AB connector (port 3).

You can download the appropriate device driver for your host system from the Future Technology Devices International Ltd. (FTDI) website at *http://www.ftdichip.com/FTDrivers.htm*.

To connect to the board:

1. Connect the USB-A to USB micro-A cable between your host machine and the board's console USB port.
2. Connect the provided power supply to the board.
3. On your host machine, using a terminal software and the appropriate virtual COM port, connect with these settings to t:
    - Baudrate: 115200
    - Data: 8 bit
    - Stop: 1 bit
    - Parity: no
    - Flow control:none

Configure the board switches

Use the board's DIP switches to select boot source and the Jacinto 6 board functionality.

If you have daughter boards connected to your Jacinto 6 EVM board, there are additional DIP switch settings on those daughter boards that aren't discussed here that you typically must set. For example, there are separate DIP switches on the Vision board, in addition to the DIP switches on the Jacinto 6 EVM board you you must set. For more information, see the documentation from your hardware vendor.

Set primary boot device

The SW2 switch bank determines where the board's ROM code looks first for a boot loader. SW2 (1 to 8) maps to the board's `sysboot` interface (0 to 7). The first six switches determine the boot sequence.

To configure the board to boot from the microSD card (MMC1) configure the SW2 DIP switches as shown below. If the board is unable to boot from the microSD card, it attempts to boot from the QSPI NOR device.

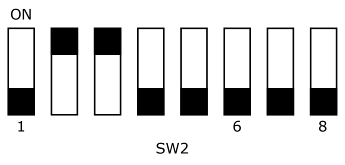


Figure 3: SW2 configuration to boot from the microSD card (MMC1)

To configure the board to boot from the QSPI NOR device, set the SW2 switch bank to one of the configurations shown below.

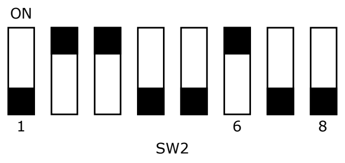


Figure 4: SW2 configuration to boot from QSPI NOR

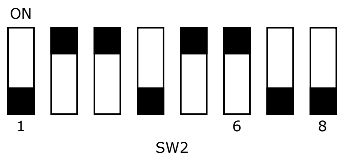


Figure 5: Alternate SW2 configuration to boot from QSPI NOR

For more details, please refer to the “Booting Devices Order” table in the board's *Technical Reference Manual* (Table 32-7 Version G of the manual).

System clock speed

The SW3 switch bank (1 to 8) maps to the board's `sysboot` interface (8 to 15). SW3-2 maps to `sysboot9`, which configures the board's clock speed (SYS_CLK1) to 20 MHz. Don't change this setting when changing the boot sequence configuration. Please refer to the “Sysboot Pads Description” (Table 32-4) in the board's *Technical Reference Manual* for more details.

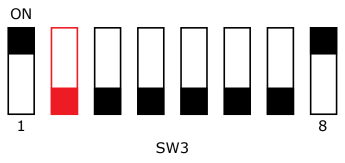


Figure 6: SW3 configuration to set the system clock speed to 20 MHz.

On-board boot routing control

The SW5 switch bank (1 to 10) controls the boot routing. To get the console working at boot time, you must set SW5-3 to `ON`. The SW5-3 switch sets `UART_SEL1_3`, which connects *UART3* to the FT232RQ. Both U-Boot and the QNX IPL assume this connection. For more information, see the *Vayu EVM CPU Board User's Guide*.

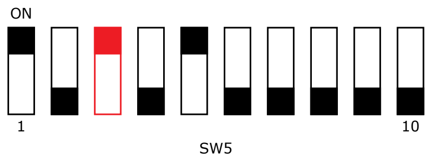


Figure 7: SW5 configuration ensuring the UART3-FT32RQ connection, using parallel NOR

GPMC_nCS0 is used by parallel NOR Flash memory. To ensure that GPMC-nCS0 is connected to parallel NOR, set SW5-1 for parallel NOR (U83, S29GL512S10) to “ON”, as shown below.

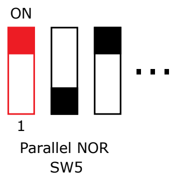


Figure 8: SW5 configuration for parallel NOR

Signaling and operational modes

As the *Texas Instruments Jacinto 6 Vayu EVM CPU Board User's Guide* indicates, leave the SW8 switches in their default positions. For more information, see the Texas Instruments website.

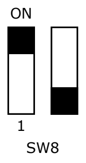


Figure 9: Default SW8 configuration (revisions older than rev. G)

For more information about SW8 settings for different board revisions, see "*Driver Commands for the Jacinto 6 EVM*" and "*Use the Multichannel Audio Serial Port on the Jacinto 6 EVM*."

Set up the AM572x EVM

These are the connectors that you must connect for the AM572x EVM

Connectors for the AM572x EVM

The following ports, connectors and buttons are of most interest when you begin working with your AM572x board:

Front side

- 12V DC board power supply port
- Power ON/OFF button
- Ethernet ports (x2)
- eSata port
- HDMI port

Back side

- Reset button
- 3.5mm Audio out port
- 3.5mm Audio in port
- microSD card slot
- USB 3.0 Host ports (x3)
- Micro USB 2.0 OTG port

Top side

- Serial Debug console connector for to USB cable (FTDI) (You many require adapter cable)

Before you start up your board, you need to connect the 12V DC power supply. Depending on what you're doing, you may also want to connect an Ethernet cable connected to your network.

Configure the board switches on the AM572x

For information about how to configure the board switches, see the board manufacturer's documentation at *http://processors.wiki.ti.com/index.php/AM572x_General_Purpose_EVM_HW_User_Guide*.

Connect to the AM572x EVM

You require a USB serial device driver on your host system in order to establish a terminal session with the target over USB. See the “Required software” section in the “*Before You Begin*” chapter of this guide for more information.

The AM572x EVM has an embedded serial-USB conversion chip. To be able to use this chip, you need to install an FTDI driver on your host system, and connect to the Jacinto 6 EVM target board via the board's serial connector (J37).

To connect to the board:

1. Connect the 6-pin connector to USB cable between your host machine and the board's serial port.
2. Connect the provided power supply to the board.
3. On your host machine, using a terminal software and the appropriate virtual COM port, connect with these settings to t:
   - Baudrate: 115200
   - Data: 8 bit
   - Stop: 1 bit
   - Parity: no
   - Flow control:none

Prepare a bootable microSD card

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

Depending on your host OS, follow the steps in either the “Prepare a bootable microSD card on a Linux or macOS host” or “Partition and format the microSD card on a Windows host” section to prepare your microSD 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 microSD 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 microSD card on a Linux or macOS host

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

1. Run the following command, once with the microSD 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 microSD 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 microSD 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 microSD 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 microSD 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 microSD card on a Windows host

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

You can prepare images to boot from either U-Boot or a QNX IPL on the board

For information about transferring images on to the Jacinto 6 EVM, see the "*Transfer images to the Jacinto 6 EVM*" section in this guide..

For information about transferring images on to the AM572x EVM, see the "*Transfer images to the AM572x EVM*" section in this guide.

Transfer images to the Jacinto 6 EVM

You can prepare images to boot from either U-Boot or a QNX IPL on the Jacinto 6 EVM.

The mechanism you choose is up to you. With QNX IPL images, you can copy IPL and IFS images to QSPI NOR Flash or put the IFS on the eMMC on the board. If you build an optimized boot image, you can also put the entire image on the eMMC board to boot the board. Using the eMMC can help you to optimize the boot time.

For descriptions of the boot processes, see "*Prepare the QNX IPL boot images*" and "*Prepare U-Boot*." After you prepare your boot image and copy the image to your microSD card, see *Boot the Jacinto 6 EVM using U-Boot* or *Boot the Jacinto 6 EVM board using QNX IPL* to understand how to boot your card using U-Boot or QNX IPL, respectively.

Transfer the QNX IPL and IFS images to the Jacinto 6 EVM board

If you need to make changes to the IPL or the IFS, you can make your changes, and then rebuild the images on your host system.

1. The IPL binary: **ipl-dra74x-vayu-evm.bin** comes prebuilt in the *BSP_ROOT_DIR*/**images** directory. If you need to change the IPL source code, you can rebuild the IPL binary without rebuilding the whole BSP.

   To rebuild the IPL:

   ```
   $ cd BSP_ROOT_DIR/src/hardware/ipl/boards/dra74x/arm/vayu-evm.le.v7
   $ make clean
   $ make
   $ cd BSP_ROOT_DIR/images
   $ sh mkflashimage.sh
   ```

   The **mkflashimage.sh** script generates the IPL file **ipl-dra74x-vayu-evm.bin**.

   The QNX IPL assumes that the filename of the OS image is **qnx-ifs**.

2. Rebuild the IFS, and clean up your *BSP_ROOT_DIR*/**images** directory:

   ```
   $ cd BSP_ROOT_DIR/src/hardware/startup/boards/dra74x/vayu-evm
   $ make
   ```

```
$ make install
$ cd BSP_ROOT_DIR/images
$ make clean
$ make
```

3. Copy the new images to the microSD card (if you're using the IFS without Screen, use **ifs-dra74x-vayu-evm.bin** instead of **ifs-dra74x-vayu-evm-graphics.bin**:

```
$ cp BSP_ROOT_DIR/images/ipl-dra74x-vayu-evm.bin $SD_CARD/mlo
$ cp BSP_ROOT_DIR/images/ifs-dra74x-vayu-evm-graphics.bin $SD_CARD/qnx-ifs
```

4. Insert the microSD card into the Jacinto 6 EVM board.
5. Power up the board.

   When the IPL boot menu appears, type **m** to load the IFS from the microSD card.

The startup build file overwrites the **images/vayu-evm.build** and **images/vayu-evm-graphics.build** files.

Copying the IPL and IFS images to QSPI NOR Flash on the Jacinto 6 EVM board

The following instructions partition the 32 MB S25FL256SAGMFV001 QSPI NOR Flash storage into these two parts:

Raw partition

This partition is for the IPL and the IFS. It starts at offset `0`. The IPL uses up to the first 64 KB; the IFS starts at offset `64`, and is of variable length.

Filesystem

This partition is for the other filesystems, as required. It begins after the raw partition. It can be partitioned into as many filesystems as are needed.

Before you erase the QSPI NOR Flash, you should:

- Transfer the IPL (**ipl-dra74x-vayu-evm.bin**) and Image Filesystem (IFS) (**ifs-dra74x-vayu-evm.bin**) images to the Jacinto 6 EVM target.
  - For the images with Screen **ipl-dra74x-vayu-evm.bin** and **ifs-dra74x-vayu-evm-graphics.bin**, respectively
  - For the images without Screen **ipl-dra74x-vayu-evm.bin** and **ifs-dra74x-vayu-evm.bin**, respectively

  If you are performing this task on a remote target, ensure that it has remote access to these images, for example, using the **fs-nfs3** or **fs-cifs** filesystem.

  The rest of these instructions assume that these images have been copied to the **/tmp** directory on the target board. For example, you can enable FTP or use the QNX Momentics IDE to copy the files to your board.
- Check that the QSPI NOR Flash driver (**devf-j6evm-qspi**) is running and that the **/dev/fs0** directory exists.

To copy the IPL and IFS images to QSPI NOR Flash:

1. The IPL must be copied to the beginning of the **/dev/fs0p0** partition. Use one of the following mechanisms to copy the IPL to beginning of the **/dev/fs0p0** partition

   - Erase the QSPI NOR Flash device and copy the IPL to the beginning of the partition:

     ```
     # flashctl -p /dev/fs0p0 -l64k -ev
     Erasing device /dev/fs0p0
     .
     # cp -V /tmp/ipl-dra74x-vayu-evm.bin /dev/fs0p0
     cp: Copying /tmp/ipl-dra74x-vayu-evm.bin to /dev/fs0p0
     100.00% (26/26 kbytes, 838 kb/s)
     ```

   - Use the **dd** file conversion utility to copy the IPL:

     ```
     # dd if=/tmp/ipl-dra74x-vayu-evm.bin of=/dev/fs0p0 seek=0 skip=0
     53+1 records in
     53+1 records out
     ```

2. Erase the storage space need for the IFS, starting from offset 64 kB:

   ```
   # flashctl -p /dev/fs0p0 -o64k -l4M -ev
   Erasing device /dev/fs0p0
   ...............
   # dd if=/tmp/ifs-dra74x-vayu-evm.bin of=/dev/fs0p0 skip=0 seek=64 bs=1024
   2788+0 records in
   2788+0 records out
   ```

3. Restart the board. When the IPL boot menu appears, type **q** to load the IFS from the QSPI NOR Flash filesystem.

Creating additional filesystem partitions on the Jacinto 6 EVM board

You can create additional filesystem partitions on the QSPI NOR Flash device. These partitions can function as root filesystems, or can be used for other purposes.

These instructions show you how to create a 24 MB Flash filesystem that starts at offset 8 M and runs to the end of the QSPI NOR Flash storage:

1. In a serial connection to the target, erase the QSPI NOR Flash device, starting from the 8 MB offset:

   ```
   # flashctl -p /dev/fs0p0 -o8m -ev
   Erasing device /dev/fs0p0
   ...............................................................
   ```

2. Create the raw partition for the IPL, and a Flash filesystem, which starts from offset 8 MB and runs to the end of the storage:

   ```
   # flashctl -p /dev/fs0p0 -o8M -fv
   Formatting device /dev/fs0p0
   # slay devf-j6evm-qspi
   ```

```
# devf-j6evm-qspi edma=8,clk=64000000
# ls /dev/fs0*
/dev/fs0       /dev/fs0p0     /dev/fs0p1
```

When the filesystem creation completes, you should have the following partitions on your QSPI NOR Flash storage device:

/dev/fs0p0

Raw partition, at offset 0-4 MB.

/dev/fs0p1

Flash filesystem partition, at offset 8 MB - 32 MB (the end of the device storage) by default, this partition is mounted as **/fs0p1**.

Copy the IFS image to the eMMC on the Jacinto 6 EVM board

Though we don't boot the IPL from the eMMC, we can load the IFS from the eMMC. The device node for eMMC is determined by the options the **devb-sdmmc-omap_generic** driver passes to the **cam-disk.so** driver. The instructions below assume that the eMMC node is **/dev/emmc0**, which is the default for this BSP.

To copy the IFS image to the eMMC:

1. In a serial connection to the target, check the partition information:

```
# fdisk /dev/emmc0 info
Physical disk characteristics: (/dev/emmc0)
Disk type        : Direct Access (0)
Cylinders        : 7376
Heads            : 64
Sectors/Track    : 32
Total Sectors    : 15106048
Partition table information:
0: (0) beg(h=0,s=0,c=0) end(h=0,s=0,c=0) off=0, size=0
1: (0) beg(h=0,s=0,c=0) end(h=0,s=0,c=0) off=0, size=0
2: (0) beg(h=0,s=0,c=0) end(h=0,s=0,c=0) off=0, size=0
3: (0) beg(h=0,s=0,c=0) end(h=0,s=0,c=0) off=0, size=0
signature1=0x00, signature2=0x00
```

2. Delete all existing partitions:

```
# fdisk /dev/emmc0 delete -a
# fdisk /dev/emmc0 show

_____OS_____      Start      End      ______Number______    Size    Boot
name    type    Cylinder  Cylinder  Cylinders   Blocks

1.   ------   ---   --------   --------   -------  --------  -----
2.   ------   ---   --------   --------   -------  --------  -----
```

```
3.   ------   ---   --------   --------   -------  --------  -----
4.   ------   ---   --------   --------   -------  --------  -----
```

3. Create a 3 GB FAT32 partition from the beginning of the eMMC device:

```
# fdisk /dev/emmc0 add -t 12 -c 0,3000
# fdisk /dev/emmc0 show

_____OS_____      Start      End     ______Number______    Size    Boot
name    type    Cylinder  Cylinder  Cylinders   Blocks

1.   FAT32     12          0       3000      3001    6146016   3000 MB
2.   ------   ---   --------   --------   -------  --------  -----
3.   ------   ---   --------   --------   -------  --------  -----
4.   ------   ---   --------   --------   -------  --------  -----
```

4. Use the `mount` command to enumerate the partitions:

```
# mount -e /dev/emmc0
# ls /dev/emmc0*
/dev/emmc0         /dev/emmc0t12
```

5. Format the FAT32 partition and mount it:

```
# mkdosfs /dev/emmc0t12

Format complete: FAT32 (4096-byte clusters), 3067000 kB available.
# mount -t dos /dev/emmc0t12 /fs/emmc
```

6. Copy the IFS to the eMMC (if you're using the IFS without Screen, use **ifs-dra74x-vayu-evm.bin** instead of **ifs-dra74x-vayu-evm-graphics.bin**):

```
# cp /tmp/ifs-dra74x-vayu-evm-graphics.bin /fs/emmc/qnx-ifs
```

7. When the IPL boot menu appears, type **e** to load the IFS from the eMMC.

Prepare U-Boot

U-Boot requires two images to boot the Jacinto 6 EVM board. To prepare to boot from U-Boot, on your target system, copy both images to your formatted microSD card. Copy:

MLO

the first-stage boot loader; it is loaded by the Texas Instruments ROM boot loader

u-boot.img

the second-stage boot loader

After U-Boot has been launched, it can load the QNX IFS. You can copy *BSP_ROOT_DIR*/**images/ifs-dra74x-vayu-evm-graphics.bin** or *BSP_ROOT_DIR*/**images/ifs-dra74x-vayu-evm.bin** to the microSD card or load it using TFTP.

Transfer images to the AM572x EVM

You can prepare images to boot an IFS file using either U-Boot or a QNX IPL.

For descriptions of the boot processes, see "*Boot the AM572x board using QNX IPL*" and "*Boot the AM572x board using U-Boot*."

Transfer U-Boot to the microSD

U-Boot needs two images to boot the AM572x EVM board. To prepare to boot from U-Boot, on your host system, copy both images to your formatted microSD card. Copy:

MLO	the first-stage boot loader; it is loaded by the Texas Instruments ROM boot loader
u-boot.img	the second-stage boot loader

Once U-Boot has been launched, it can load either a Linux OS or the QNX IFS. For the QNX IFS, you may either copy *BSP_ROOT_DIR*/**images/ifs-dra74x-am572x-evm.bin** to the microSD card, or load it via TFTP.

Transfer the QNX IPL boot images

If you need to make changes to the IPL or the IFS, you can make these changes then rebuild the images:

1. Insert your microSD card into your host system.
2. Copy the new images to the microSD card starting with the ipl-dra74x-am572x-evm.bin first (renamed to **mlo**), and then the ifs-dra74x-am572x-evm.bin (renamed to **qnx-ifs**). For example:

```
$ cp BSP_ROOT_DIR/images/ipl-dra74x-am572x-evm microsd_card_location/mlo
$ cp BSP_ROOT_DIR/images/ifs-dra74x-am572x-evm.bin microsd_card_location/qnx-ifs
```

3. Insert the microSD card into the AM572x EVM board.
4. Power up the board.

   When the IPL boot menu appears, type **m** to load the IFS from the microSD card.

Copying the IFS image to the eMMC

Though we don't boot the IPL from the eMMC, we can load the IFS from the eMMC. The device node for eMMC is determined by the options the **devb-sdmmc-omap_generic** driver passes to the **cam-disk.so** driver. The instructions below assume that the eMMC node is **/dev/emmc0**, which is the default for this BSP.

To copy the IFS image to the eMMC:

1. Check the partition information:

   ```
   # fdisk /dev/emmc0 info
   Physical disk characteristics: (/dev/emmc0)
   ```

```
Disk type        : Direct Access (0)
Cylinders        : 7376
Heads            : 64
Sectors/Track    : 32
Total Sectors    : 15106048
Partition table information:
0: (0) beg(h=0,s=0,c=0) end(h=0,s=0,c=0) off=0, size=0
1: (0) beg(h=0,s=0,c=0) end(h=0,s=0,c=0) off=0, size=0
2: (0) beg(h=0,s=0,c=0) end(h=0,s=0,c=0) off=0, size=0
3: (0) beg(h=0,s=0,c=0) end(h=0,s=0,c=0) off=0, size=0
signature1=0x00, signature2=0x00
```

2. Delete all existing partitions:

```
# fdisk /dev/emmc0 delete -a
# fdisk /dev/emmc0 show

_____OS_____      Start      End     ______Number______    Size    Boot
name    type    Cylinder  Cylinder  Cylinders   Blocks

1.   ------   ---   --------   --------   -------  --------  -----
2.   ------   ---   --------   --------   -------  --------  -----
3.   ------   ---   --------   --------   -------  --------  -----
4.   ------   ---   --------   --------   -------  --------  -----
```

3. Create a 3 GB FAT32 partition from the beginning of the eMMC device:

```
# fdisk /dev/emmc0 add -t 12 -c 0,3000
# fdisk /dev/emmc0 show

_____OS_____      Start      End     ______Number______    Size    Boot
name    type    Cylinder  Cylinder  Cylinders   Blocks

1.   FAT32     12          0       3000      3001    6146016   3000 MB
2.   ------   ---   --------   --------   -------  --------  -----
3.   ------   ---   --------   --------   -------  --------  -----
4.   ------   ---   --------   --------   -------  --------  -----
```

4. Enumerate the partitions:

```
# mount -e /dev/emmc0
# ls /dev/emmc0*
/dev/emmc0         /dev/emmc0t12
```

5. Format the FAT32 partition and mount it:

```
# mkdosfs /dev/emmc0t12
```

```
Format complete: FAT32 (4096-byte clusters), 3067000 kB available.
# mount -t dos /dev/emmc0t12 /fs/emmc
```

6. Copy the IFS to the eMMC:

   ```
   # cp /tmp/ifs-dra74x-am572x-evm.bin /fs/emmc/qnx-ifs
   ```

7. When the IPL boot menu appears, select **e** to load the IFS from the eMMC.

Boot the board

After you have your image ready, you can power and boot your target board.

DANGER:

Use only the 12V DC power supply provided with the board. Use of any other power supply may permanently damage the board. When power cycling the board, turn the power on or off from the AC adapter side, rather than removing and inserting the DC power plug on the board, to prevent damaging the board's power-regulation circuitry.

Follow the relevant instructions depending on whether you chose to use QNX IPL or U-Boot:

- To understand how to boot the Jacinto 6 EVM board using QNX IPL, see "*Boot the Jacinto 6 EVM board using QNX IPL*"
- To understand how to boot the Jacinto 6 EVM board using U-Boot, see "*Boot the Jacinto 6 EVM using U-Boot*"
- To understand how to boot the AM572x GP board using QNX IPL, see "*Boot the AM572x board using QNX IPL*"
- To understand how to boot the AM572x GP board using U-Boot, see "*Boot the AM572x board using U-Boot*"

Boot the Jacinto 6 EVM board using QNX IPL

You can use the QNX IPL to boot your system.

The boot process with the QNX IPL

Booting the QNX IPL involves three stages:

1. The board's on-chip ROM bootloader (RBL) takes control right after the board powers up. When the low-level initializations are complete, the RBL loads the QNX IPL from the boot device.
2. The QNX IPL can be on a microSD card, on a QSPI NOR Flash device. It configures the hardware to create an environment that will allow the startup program and the QNX Neutrino microkernel to run, and to load the IFS.
3. The IFS can be on a microSD card, on a QSPI NOR Flash device or on the eMMC, or it can be downloaded through a serial port. It contains the startup program, OS kernel, build scripts, and any other drivers, applications and binaries that the system requires.

The IPL and the IFS can be loaded from different devices.

If you want to transfer the IFS using a serial connection, use the `sendnto` command or the IDE. For more information, see the "`sendnto`" utility in the *Utilities Reference* or "Using the QNX Send File button" in the *QNX Momentics IDE User's Guide*.

Manual boot mode

Manual boot is the default boot mode for the QNX IPL. When you boot the Jacinto 6 EVM board in manual boot mode, it starts the IPL and shows the following menu:

```
QNX Neutrino Initial Program Loader for DRA74X EVM

Found DRA7xx ES1.1 SoC
Booting from SD card
Command:
Press 'S' for SERIAL download.
Press 'M' for SDMMC download, file QNX-IFS assumed.
Press 'E' for EMMC download, file QNX-IFS assumed.
Press 'Q' for QSPI NOR flash download, IFS assumed to be present at 64KB offset.
Press 'F' for USB booting.
Press 'T' for Memory test.
```

Choose the location that you want to load the IFS. For example, provided **qnx-ifs** has been copied to the microSD card, you can load it by pressing **M**.

The QNX IPL booting the Jacinto 6 EVM board will log something like the following on your host system's terminal console:

```
Load QNX image from SDMMC...
Found image        @ 0x81800008
Jumping to startup @ 0x801041D4

VFPv3: fpsid=410430f0
coproc_attach(10): replacing fe062064 with fe0788f0
coproc_attach(11): replacing fe062064 with fe0788f0
Welcome to QNX Neutrino 7.0.0 on the Texas Instruments J6 EVM(ARMv7 Cortex-A15 core)
Starting UART1 (Console)
starting I2C driver...
Starting QSPI NOR Flash Driver...
#
```

USB booting:

To boot your target via USB booting, ensure that you install the *Fastboot utility* on your host machine. USB booting uses the Fastboot protocol to flash a filesystem (QNX-IFS) via a USB connection from your host machine to the board. For information about how to get the Fastboot utility and use it, see *https://wiki.cyanogenmod.org/w/Doc:_fastboot_intro*.

CAUTION: Before you select the USB booting option, ensure that you connect your host machine to the target. You can use a USB 3.0 or micro-USB 2.0 cable. If your cable isn't connected, the board won't boot because it is stuck waiting for an image. There's mechanism available to detect to if it is connected to the host.

To use USB booting:

1. In the terminal session to the target (via serial port), press `F` to boot the target using USB booting.
2. In a Command Prompt window (Windows) or a terminal (Linux, macOS) on your host machine, navigate to the **/images** directory of your BSP.
3. Run the Fastboot utility to send the QNX-IFS to the target:

   For Windows (run command as an Administrator):

   ```
   fastboot boot ifs-dra74x-vayu-evm.bin
   ```

   For Linux:

   ```
   sudo fastboot boot ifs-dra74x-vayu-evm.bin
   ```

After you run the command, you should see the QNX-IFS boot in your terminal session to your target. For example:

```
HIGH SPEED DETECTED
set config = 00000001
download:0072f800
Starting download bytes:0072F800
download finished bytes 0072F800
boot
10616 ms on IPL total
smp_num_cpus: 2 cpus
cpu0: 412fc0f2: Cortex A15 r2p2 1000MHz
cpu0: MIPDIR=80000000
cpu0: CTR=8444c004 CWG=64 ERG=64 Dminline=64 Iminline=64 PIPT
cpu0: CLIDR=a200023 LoUU=1 LoC=2 LoUIS=1
cpu0: L1 Icache: 512x64
cpu0: L1 Dcache: 512x64 WB
cpu0: L2 Dcache: 32768x64 WB
cpu0: VFP-d32 FPSID=410430f0
cpu0: NEON MVFR0=10110222 MVFR1=11111111
JAMR3 detected!
Vayu EVM hardware Rev F or Rev G
```

Autoboot mode

In autoboot mode, the boot process automatically loads the IFS from the same device as the IPL. If the boot process fails to load the IFS or the loaded IFS fails the checksum calculation, the IPL automatically switches to manual boot mode, displays the boot menu and waits for instructions from the user.

To use autoboot, you need to edit the IPL **Makefile**:

1. Go to the *$(IPL)***/arm/vayu-evm.le.v7** directory and open **Makefile** with a text editor.

2. Modify **Makefile** so that `MANBOOT` is set to `0`:

   ```
   include ../../../common.mk

   # MANBOOT OPTION IS USED TO ALLOW USERS TO INTERRUPT AUTO BOOT.
   # BY DEFAULT, IT IS SET TO 0 TO ALLOW AUTO BOOT
   # SET TO 1, IF MANUAL BOOT IS DESIRED (i.e. CCFLAGS += -DMANBOOT=1 )

   # BOOT TYPE
   CCFLAGS += -DMANBOOT=0
   ```

3. Save the **Makefile** file, rebuild the IPL, and regenerate the IPL image.

In autoboot mode, the `NO_DISPLAY` macro is defined automatically to not print any debugging information and keep the boot time to the minimum possible.

To use manual boot, you must set `MANBOOT` to `1`(one).

Boot the Jacinto 6 EVM using U-Boot

You can use U-Boot to boot your system.

To boot your Jacinto 6 EVM board from U-Boot:

1. Connect your Jacinto 6 EVM target board to your host system.
2. On your host machine, start your favorite terminal program with these settings:

   ```
   Baud: 115200
   Bits: 8
   Stop bits: 1
   Parity: none
   ```

3. Connect the Jacinto 6 EVM board to the power supply, then push the **PWR RESET** button.

   You should see the output from U-Boot appear on your host system console.

4. When U-Boot prompts with “Hit any key to stop autoboot”, press a key on your host’s keyboard to interrupt the boot process.
5. Use the command-line instructions below to set the U-Boot environment variables so that the IFS can be launched via TFTP or from either SD card:

   With Screen:

   ```
   DRA752 EVM # setenv loadaddr '0x80100000'
   DRA752 EVM # setenv bootcmd_microsd 'mmcinfo; fatload mmc 0 ${loadaddr}
   ifs-dra74x-vayu-evm-graphics.bin; go ${loadaddr}'
   DRA752 EVM # >setenv bootcmd 'run bootcmd_microsd'
   DRA752 EVM # >saveenv
   DRA752 EVM # >boot
   ```

Without Screen:

```
DRA752 EVM # setenv loadaddr '0x80100000'
DRA752 EVM # setenv bootcmd_microsd 'mmcinfo; fatload mmc 0 ${loadaddr} ifs-
DRA752 EVM # >setenv bootcmd 'run bootcmd_microsd'
DRA752 EVM # >saveenv
DRA752 EVM # >boot
```

6. To re-boot with the new environment variables, type `boot` in the command line.

In your terminal connection, you should see output that shows U-Boot booting the Jacinto 6 EVM board:

```
U-Boot SPL 2013.04-rc1-g00344b1-dirty (March 03 2016 - 21:45:53)
DRA752 ES1.0
OMAP SD/MMC: 0
reading u-boot.img
reading u-boot.img

U-Boot 2013.04-rc1-g00344b1-dirty (March 03 2016 - 21:45:53)

CPU  : DRA752 ES1.0
Board: DRA7xx
I2C:   ready
DRAM:  1 GiB
WARNING: Caches not enabled
MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1
In:    serial
Out:   serial
Err:   serial
Hit any key to stop autoboot:  0
Device: OMAP SD/MMC
Manufacturer ID: 3
OEM: 5344
Name: SU02G
Tran Speed: 50000000
Rd Block Len: 512
SD version 2.0
High Capacity: No
Capacity: 1.8 GiB
Bus Width: 4-bit
Reading ifs-dra74x-vayu-evm-graphics.bin
6132520 bytes read in 722 ms (8.1 MiB/s)
□ÿVFPv3: fpsid=410430f0 at 0x80100000 ...
coproc_attach(10): replacing fe062064 with fe0788f0
coproc_attach(11): replacing fe062064 with fe0788f0
```

```
Welcome to QNX Neutrino 7.0.0 on the Texas Instruments J6 EVM(ARMv7 Cortex-A1
Starting UART1 (Console)
starting I2C driver...
Starting QSPI NOR Flash Driver...
#
```

If you loaded the IFS that includes graphics (Screen Graphics Subsystem):

```
U-Boot SPL 2013.04-rc1-g00344b1-dirty (March 02 2016 - 22:00:01)
DRA752 ES1.0
OMAP SD/MMC: 0
reading u-boot.img
reading u-boot.img

U-Boot 2013.04-rc1-g00344b1-dirty (March 02 2016 - 22:00:01)

CPU  : DRA752 ES1.0
Board: DRA7xx
I2C:   ready
DRAM:  1 GiB
WARNING: Caches not enabled
MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1
In:    serial
Out:   serial
Err:   serial
Hit any key to stop autoboot:  0
Device: OMAP SD/MMC
Manufacturer ID: 3
OEM: 5344
Name: SU02G
Tran Speed: 50000000
Rd Block Len: 512
SD version 2.0
High Capacity: No
Capacity: 1.8 GiB
Bus Width: 4-bit
Reading ifs-dra74x-vayu-evm.bin
6132520 bytes read in 722 ms (8.1 MiB/s)
□ÿVFPv3: fpsid=410430f0 at 0x80100000 ...
coproc_attach(10): replacing fe062064 with fe0788f0
coproc_attach(11): replacing fe062064 with fe0788f0
Welcome to QNX Neutrino 7.0.0 on the Texas Instruments J6 EVM(ARMv7 Cortex-A1
Starting UART1 (Console)
starting I2C driver...
```

```
Starting QSPI NOR Flash Driver...
#

#
```

You should now be able to test your image by executing any shell command, or any command residing within the OS image, such as `ls` or `uname`.

Boot the AM572x board using QNX IPL

You can use the QNX IPL to bootthe AM572x board.

The boot process with the QNX IPL

Booting the QNX IPL involves three stages:

1. The board's on-chip ROM bootloader (RBL) immediately takes control after the board powers up. When the low-level initializations are done, the RBL loads the QNX IPL from the boot device.
2. The QNX IPL can be on an microSD card or downloaded to the board using TFTP. It configures the hardware to create an environment that allows the startup program and the microkernel to run, and to load the IFS.
3. The IFS can be on an microSD card or on the eMMC. You can also download the image from your host to the board through a serial port. It contains the startup program, OS kernel, build scripts, and any other drivers, applications and binaries that the system requires.

The IPL and the IFS can be loaded from different devices.

Manual boot mode

Manual boot is the default boot mode for the QNX IPL. When you boot the AM572x EVM board in manual boot mode, after it starts the IPL shows the menu:

```
QNX Neutrino Initial Program Loader for AM572x EVM

Found AM572x ES1.1 SoC
Booting from SD card
Command:
Press 'S' for SERIAL download.
Press 'M' for SDMMC download, file QNX-IFS assumed.
Press 'E' for EMMC download, file QNX-IFS assumed.
```

Choose the location from which you want to load the IFS. For example, provided **qnx-ifs** has been copied to the microSD card, you can load it by pressing **M**.

The QNX IPL booting the AM572x EVM board will log something like the following on your host system's terminal console:

```
Load QNX image from SDMMC...
Found image        @ 0x81800008
Jumping to startup @ 0x801041D4

VFPv3: fpsid=410430f0
coproc_attach(10): replacing fe062064 with fe0788f0
coproc_attach(11): replacing fe062064 with fe0788f0
Welcome to QNX Neutrino 7.0.0 on the Texas Instruments AM572X EVM
Starting UART1 (Console)
starting I2C driver...
#
```

Autoboot mode

In autoboot mode, the boot process will automatically load the IFS from the same device as the IPL. If the boot process fails to load the IFS or the loaded IFS fails the checksum calculation, the IPL will automatically switch to manual boot mode, display the boot menu and wait for instructions from the user.

To use autoboot, you need to edit the IPL Makefile:

1. Go to the *$(IPL)***arm/am572x-evm** directory and open **Makefile** with a text editor.
2. Modify **Makefile** so that `MANBOOT` is set to `0`:

   ```
   include ../../../common.mk

   # MANBOOT OPTION IS USED TO ALLOW USERS TO INTERRUPT AUTO BOOT.
   # BY DEFAULT, IT IS SET TO 0 TO ALLOW AUTO BOOT
   # SET TO 1, IF MANUAL BOOT IS DESIRED (i.e. CCFLAGS += -DMANBOOT=1 )

   # BOOT TYPE
   CCFLAGS += -DMANBOOT=0
   ```
3. Save the Makefile, rebuild the IPL and regenerate the IPL image.

In autoboot mode, the `NO_DISPLAY` macro is defined automatically to not print any debugging information and keep the boot time to the minimum possible.

To use manual boot, just set `MANBOOT` back to `1` (one).

Boot the AM572x board using U-Boot

You can use U-Boot to boot your system.

To boot your AM572x EVM board from U-Boot:

1. If you haven't already, connect to your AM572x EVM target board using a console.
2. Insert the microSD card into your board.
3. Connect the AM572x EVM board to the power supply, then push the **PWR RESET** button.

   You should see the output from U-Boot appear on your host in your terminal connection to the board.
4. When you see a prompt that says: `Hit any key to stop autoboot` in your terminal connection. Press any key on your keyboard to interrupt the boot process.
5. Enter the following commands to set the U-Boot environment variables so that the IFS can be launched from a microSD card:

   ```
   # setenv loadaddr '0x80100000'
   # setenv bootcmd_microsd `mmcinfo; fatload mmc 0 ${loadaddr} ifs-dra74x-am572
   # setenv bootcmd `run bootcmd_microsd`
   # saveenv
   EVM # boot
   ```
6. To re-boot with the new environment variables, type `boot` in the command line.

After the board boots using U-Boot, you see that QNX Neutrino boot as shown here:

```
U-Boot SPL 2013.04-rc1-g00344b1-dirty (May 06 2013 - 21:45:53)
AM572 ES1.0
OMAP SD/MMC: 0
reading u-boot.img
reading u-boot.img

U-Boot 2013.04-rc1-g00344b1-dirty (May 06 2013 - 21:45:53)

CPU  : AM572 ES1.0
Board: AM572X
I2C:   ready
DRAM:  1 GiB
WARNING: Caches not enabled
MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1
In:    serial
Out:   serial
Err:   serial
Hit any key to stop autoboot:  0
Device: OMAP SD/MMC
Manufacturer ID: 3
OEM: 5344
Name: SU02G
Tran Speed: 50000000
Rd Block Len: 512
SD version 2.0
```

```
High Capacity: No
Capacity: 1.8 GiB
Bus Width: 4-bit
Reading ifs-dra74x-am572x-evm.bin
6132520 bytes read in 722 ms (8.1 MiB/s)
ÿVFPv3: fpsid=410430f0 at 0x80100000 ...
coproc_attach(10): replacing fe062064 with fe0788f0
coproc_attach(11): replacing fe062064 with fe0788f0
Welcome to QNX Neutrino 7.0.0
Starting UART1 (Console)
starting I2C driver...
#
```

You can now test your QNX Neutrino image by executing any shell command or any command residing within the image, such as `ls` or `uname`.

Chapter 4
Configure the Screen Graphics Subsystem on the Jacinto 6 EVM and AM572x EVM

The provided default buildfile (**vayu-evm.build**) for the AM572X EVM starts the Screen Graphics Subsystem (Screen) driver. However, thethe **vayu-evm.build** buildfile that's provided for Jacinto 6 EVM doesn't start the Screen Graphics Subsystem (Screen) driver by default. Another buildfile that's included with your BSP, **vayu-evm-graphics.build** starts Screen.

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 **graphics.conf** file, which is found on the target in the **/usr/lib/graphics/jacinto6** directory. You can modify this file at runtime, but you'll need to restart Screen.

There's different graphics configuration file in the same directory for Jacinto 6 EVM boards Revision H and later.

For example:

```
# slay screen
# screen -c /usr/lib/graphics/jacinto6/graphics.conf
```

Configure the display for Screen

For the **graphics.conf** file, you must ensure that the *video-mode* settings to match the native display resolution on the display. Both the Jacinto 6 EVM revision E and earlier and the AM572x EVM boards support similar resolutions, though some resolutions are supported only by the Jacinto 6 EVM G and later as described below.

When you use Screen, you can configure the display settings in the **graphics.conf** file for Screen. If you're using a Jacinto 6 EVM, Revision G boards with an attached 10.1 inch LCD display, the *video-mode* parameter in the default **graphics.conf** file might be set to 1280x800 resolution at 57Hz as shown here:

```
...
 ...
begin display 2
    #Jacinto 6, revision E and earlier
    #video-mode = 800 x 480 @ 60
    #Jacinto 6, revision G (default)
    video-mode = 1280 x 800 @ 57
    #Jacinto 6, revision H
    #video-mode = 1920 x 1200 @ 60
end display
...
...
```

If you're using a Jacinto 6 EVM, Revision H board with an attached 10.1 inch LCD display, it must be set to 1920x1200 at 60Hz. To do this, in the **graphics.conf** file, comment out the *video-mode* entry for the Jacinto 6 EVM, revision G (default) and then uncomment the *video-mode* entry for the Jacinto 6 EVM, revision H as shown here:

```
 ...
 ...
 begin display 2
    video-mode = 1920x1200@60begin display 2
    #Jacinto 6, revision F and older
    #video-mode = 800 x 480 @ 60
    #Jacinto 6, revision G (default)
    #video-mode = 1280 x 800 @ 57
    #Jacinto 6, revision H
    video-mode = 1920 x 1200 @ 60
end display
...
...
```

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

Display driver

The video display configuration library is located on the target at **/usr/lib/graphics/jacinto6/libwfdcfg-generic.so** and **/usr/lib/graphics/jacinto6/libWFDjacinto6.so**.

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 verify that your display works with basic Screen functionality.

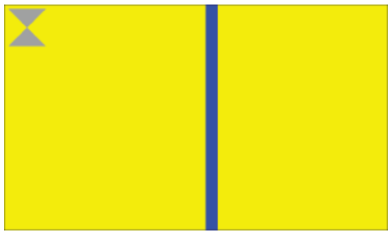


Figure 10: sw-vsync

`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.

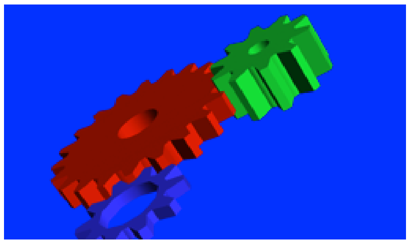


Figure 11: gles2-gears

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 with the Jacinto 6 EVM

If you have a multitouch display, you can use it with the Jacinto-based board.

Multitouch requires that you have Screen Graphics Subsystem running. For more information about using Screen, see *Configure the Screen Graphics Subsystem on the Jacinto 6 EVM and AM572x EVM*. To use touch, your target must have:

- the necessary drivers based on the touch controller for your hardware. The version of the buildfile that starts graphics (**vayu-evm-graphics.build**) is configured to use a specific-based touch controller. In the buildfile, it includes the following:
  - **libmtouch-calib.so** library (driver)
  - **libmtouch-lg-tsc101.so** library (driver)
  - **libmtouch-focal-tech.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 default touch configuration file is 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 `poll=1,verbose=3,max_touchpoints=10`, which are specific to the touch controller hardware used on the display for Jacinto 6 EVM, Revision G boards. If you have a Revision H board, see "*Configuration changes for Jacinto 6 EVM, Revision H boards*." Here's what the touch configuration file for the touch controller looks like:

```
begin mtouch
  driver = lg-tsc101
  options = poll=1,verbose=3,max_touchpoints=10
  display = 2
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 how to map different resolution types. In this case, the first valid one is used:

```
# J6EVM
800x480:mode=direct
1280x720:mode=direct
1280x800:mode=direct
1920x1200:mode=direct
```

Configuration changes to use Jacinto 6 EVM, Revision H boards

The default configuration is configured for Jacinto 6 EVM, Revision G boards. Revision H boards use a different driver for touch. Provided with this BSP is the configuration file **mtouch-j6-revh.conf**, which you specify as the config file using the `-c` option when you start the `mtouch` command. The config file specifies to use the **focaltech** driver instead of the **lg-tsc101** driver as follows:

```
begin mtouch
  driver = focaltech
  options = generic_j6=1,interrupt=1002,i2c_slave_addr=0x38,i2c_devname=/dev/i2c0,protocol_ver=1
end mtouch
```

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(`io-hid` and `libmtouch-hid`).

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
Use the Multichannel Audio Serial Port on the Jacinto 6 EVM

The Jacinto 6 EVM board supports an audio Multichannel Audio Serial Port (McASP).

Choosing the address for the codec

Jacinto 6 EVM board revisions F and later, you can connect to either a 7-inch or a 10-inch display. However, the I2C address for the 10-inch display (`0x18`) conflicts with the default address for the AIC3106 codec I2C slave. Therefore, in order to accommodate both 7-inch and and 10-inch displays, when you start **io-audio** you need to set the SW8-2 DIP switch to configure the AIC3106 codec's I2C address:

Display	Address	SW8-2
7-inch	`0x18`	"OFF"
10-inch	`0x19`	"ON"

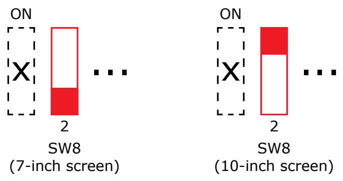


Figure 12: SW8-2 configured for a 7-inch and for a 10-inch screen.

For Jacinto 6 EVM board revisions F and later, to use a 10-inch display, start `io-audio` as follows, provided you've set the *i2c_addr*:

```
# io-audio -vv -d mcasp-j6-aic3106 i2c_addr=0x19
```

If have a 7" display and you specified the `startup` command with the `-a` option, specify the command as follows (the *i2c_addr* isn't required):

```
# io-audio -d mcasp-j6-aic3106 sample_rate=48000,ahxclk=output:12288000,mclk=12288000
```

If have a 10.1" display and you specified the `startup` command with the `-a` option, but need to specify the *i2c_addr* option, then use the following command:

```
# io-audio -d mcasp-j6-aic3106 sample_rate=48000,ahxclk=output:12288000,mclk=12288000,i2c_addr=0x19
```

Audio routing

The *Mic In* and *Line In* groups determine which input is sent to the digital converter (ADC). Their *Volume Range* parameters are pre-set to 0 (zero), as they don't affect volume, which is controlled by the "Input Gain" group.

Signals from the MIC IN (P10) and the LINE IN (P11) jacks can be routed to the ADC. To route the audio signal from MIC IN to the ADC (**waverec**), start **mix_ctrl** as follows:

```
# mix_ctl group "Mic In" capture=on
```

To route the audio signal from LINE IN to the ADC, start **mix_ctrl** as follows:

```
# mix_ctl group "Line In" capture=on
```

The *Line Out* and *Headphones* groups aren't independent data paths, so you can choose to send the audio to either the headphones or the line out to the speakers, but *not to both*. Use the **mix_ctl** utility to set the codec's analog output routing to either *Line Out* or *Headphone*.

For example:

- List all mixer switches:

  ```
  # mix_ctl  switches
  "Headphone Select"                    BOOLEAN  on
  ```

- Enable Headphone, so Line Out is disabled simultaneously:

  ```
  # mix_ctl switch " Headphone Select" on
  ```

- Disable Headphone, so Line Out is enabled simultaneously:

  ```
  # mix_ctl switch " Headphone Select" off
  ```

Chapter 7
Secure Boot Support on the Jacinto 6 EVM

You can add secure boot support to your BSP.

The secure boot functionality is supported for only the Jacinto 6 EVM. With secure boot, the QNX IPL that you build checks the signature of the IFS before loading it. As a result, during the boot process, you should see the `Verifying signature` message in your terminal when you boot the board.

To use secure boot, perform the following steps:

1. Download and set up your BSP as you would normally. See "*Download and set up the BSP*" in the "*Installation Notes*" chapter of this guide for more information.
2. Navigate to the *BSP_ROOT_DIR* and run the demo `genkeys.sh` command. Only run this command once. If you're on a Windows host, run the `bash` command from Command Prompt window first.

   The `genkeys.sh` is for development purposes. You must use your own mechanism to generate the key for production.

   ```
   > cd $BSP_ROOT_DIR/images
   > ./genkeys.sh
   ```

3. Run the following command to build a secure image:

   ```
   > cd $BSP_ROOT_DIR
   > QNX_SECURE_BOOT=1 make
   ```

4. Transfer the QNX IPL and IFS images to your bootable microSD card. For information about creating a bootable microSD and transferring your images to the card, see the "*Prepare a bootable microSD card*" and "*Transfer images to the Jacinto 6 EVM*" in the Installation Notes chapter of this guide.
5. Boot your board. For information about booting the board, see "*Boot the board*" section in the Installation Notes chapter of this guide.

Chapter 8
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-dra74x-vayu-evm.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.

Using graphics (Screen Graphics Subsystem) in the image

The default buildfile for the AM572 EVM board includes Screen Graphics Subsystem (Screen), which is required to use graphics; however the default buildfile for the Jacinto 6 EVM doesn't include Screen.

To include Screen in your image, use the buildfile located at ***$BSP_ROOT_DIR*/images/vayu-evm-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 to build IPL and IFS images for the Jacinto 6 EVM and AM572x EVM.

`j6`

Invokes all targets to build IPL and IFS images for the Jacinto 6 EVM.

`am572x`

Invokes all targets to build IPL and IFS images for the AM572x EVM.

`ifs-dra74x-vayu-evm.bin`

Builds and IFS image that excludes the Screen Graphics Subsystem. You won't be able to run graphics applications by default using this image.

`ifs-dra74x-vayu-evm-graphics.bin`

Builds and IFS image that includes the Screen Graphics Subsystem, which allows you to run various graphics applications.

`ifs-dra74x-vayu-evm-rootless.bin`

Builds and IFS image that excludes the Screen Graphics Subsystem and runs services without requiring root execution.

`ipl-dra74x-vayu-evm.bin`

Builds the IPL image file for the Jacinto 6 EVM. This make target runs the `mkflashimage.sh`.

`ifs-dra74x-am572x-evm.bin`

Builds the IFS file for the target.

`ipl-dra74x-am572x-evm.bin`

Builds the IPL image file for the AM572x EVM. This make target runs the `mkflashimage_am572x.sh`.

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-dra74x-vayu-evm.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 **ti-j6-dra74x-vayu-evm**) 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.

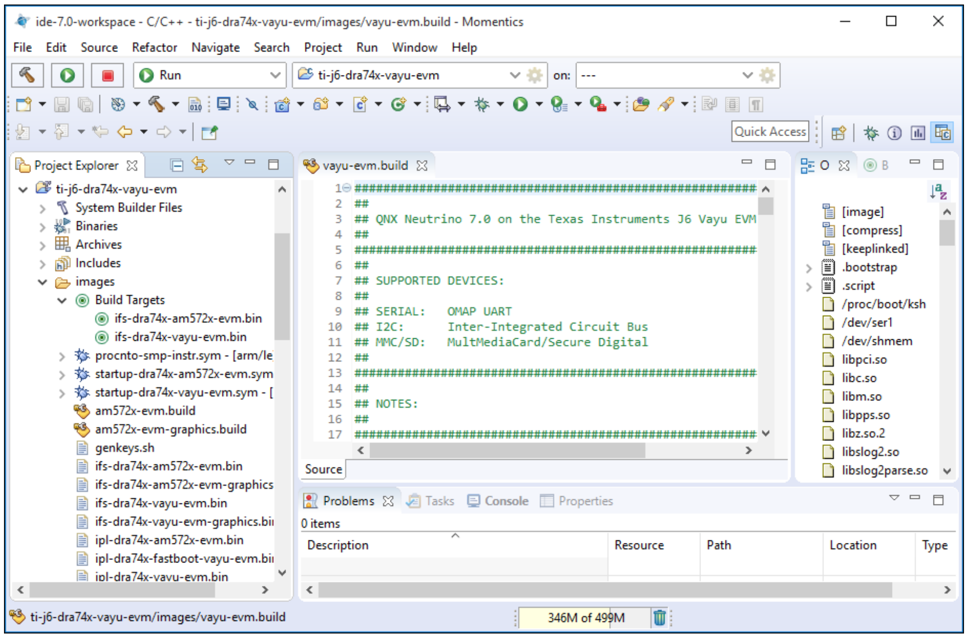
1. In the Project Explorer view, expand the **images** folder, right-click the IFS file you want recreated, and select **Delete**. The IFS must be deleted or it won't be rebuilt.
2. In the example view, right-click the **images** folder and select **Build Targets** → **Create...**.
3. In the **Create 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-dra74x-vayu-evm.bin**, it maps to the ifs-dra74x-vayu-evm.bin target in the Makefile, you would type `ifs-dra74x-vayu-evm.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:



```
###############################################################
##
## QNX Neutrino 7.0 on the Texas Instruments J6 Vayu EVM
##
###############################################################
##
## SUPPORTED DEVICES:
##
## SERIAL:   OMAP UART
## I2C:      Inter-Integrated Circuit Bus
## MMC/SD:   MultMediaCard/Secure Digital
##
###############################################################
##
## NOTES:
##
###############################################################
```

Figure 13: Build image in the IDE

Chapter 9
Driver Commands for the Jacinto 6 EVM

The tables below provide a summary of driver commands.

Some of the drivers are commented out in the buildfile. To use these drivers on the target hardware, you may need to uncomment them in the build file, 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)*
- *Real-time Clock (RTC)*
- *USB host controller*
- *USB device controller*
- *SPI*
- *QSPI NOR Flash*
- *Ethernet*
- *Input*
- *SD/MMC*
- *NAND*
- *SATA*
- *HCI*
- *Audio for MCASP3*
- *Audio for JAMR3 (McASP6)*

- *Audio for Bluetooth*
- *Graphics*
- *Touch*

If you have wireless hardware (Wilink8), you can add the *wireless (Wilink8)* driver.

Audio (MCASP3)

For detailed information about how to use Audio MCASP3, see “Using McASp3.”

Device	MCASP3
Command	For board revisions F and later which use a 10.1" display, use `io-audio -vv -d mcasp-j6-aic3106 i2c_addr=0x19`
Command	For board revisions E and earlier, use `io-audio -vv -d mcasp-j6-aic3106`
Required binaries	**io-audio**, **mix_ctl**, **wave**, **waverec**
Required libraries	**deva-ctrl-mcasp-j6-aic3106.so**, **libasound.so**
Source location	**src/hardware/deva/ctrl/mcasp**

Audio for JAMR3 (MCASP6)

Device	MCASP6
Command	`io-audio -d mcasp-j6_jamr3-aic3106`
Command (if the `startup` command is used with the `-a` option)	`io-audio -d mcasp-j6_jamr3-aic3106 sample_rate=48000,ahxclk=output:12288000,mclk=12288000`
Required binaries	**io-audio**, **mix_ctl**, **wave**, **waverec**
Required libraries	**deva-ctrl-mcasp-j6_jamr3-aic3106.so**, **libasound.so**
Source location	**src/hardware/deva/ctrl/mcasp**

Audio for Bluetooth (MCASP7)

To support audio for Bluetooth, you need to set the SW5-5 and SW5-6 DIP switches as follows:

- SW5-5: “ON.” This configures UART_SEL1_3 high to select UART3, which is needed by the HCI driver.
- SW5-6: “OFF.” This configures MCA_ENn low to enable the COM8 signals. It is also needed for Wi-Fi support.

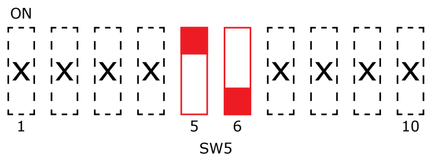


Figure 14: SW5-5 and SW-6 configuration to support Bluetooth

See "*HCI*" for details about starting **devc-seromap_hci**.

Device	MCASP7
Command	`io-audio -vv -d mcasp-j6 mcasp=7,tx_serializer=1,rx_serializer=0`
Command (if the `startup` command is used with the `-a` option)	`io-audio -d mcasp-j6 mcasp=7,clk_mode=master,tx_serializer=1,rx_serializer=0,sample_rate=8000,ahxclk=output:12288000,mclk=12288000`
Required binaries	**io-audio**, **mix_ctl**, **wave**, **waverec**
Required libraries	**deva-ctrl-mcasp-j6.so**, **libasound.so**
Source location	**src/hardware/deva/ctrl/mcasp**

McASP7 works in slave mode, which means that the Wi-Fi module should provide the clock back to McASP. Make sure you download and use the proper Bluetooth script to enable the clock in the Wi-Fi module.

Graphics

Device	GRAPHICS
Command	`screen`
Required binaries	**screen**, **gles1-gears**, **gles2-gears**, **sw-vsync**
Required libraries	For more information, see the *Screen Graphics Subsystem Developer's Guide*.
Source location	**N/A**

HCI

Device	SERIAL shared transport
Command	`devc-seromap_hci -E -f -S -n /etc/system/config/wl1285.bts -b 115200 -g 0x4805b000,132 0x48020000,106`

Required binaries	**devc-seromap_hci**
Required libraries	N/A
Source location	**src/hardware/devc/seromap_hci**

- The **wl1285.bts** file is a sample Bluetooth script; ensure that you specify the relevant script path in the command line option.
- BT_EN is connected to GPIO5[4], which is GPIO132; hence the `132` parameter for the `-g` option.
- The `106` parameter for the `-g` is for the UART3 interrupt vector.

I2C

The Jacinto 6 EVM board supports I2C devices on buses 1, 2, 3 and 4. You need to launch an I2C driver instance for each device.

Devices	I2C1, I2C2, I2C3, I2C4
Command	`i2c-omap35xx-omap4 -p 0x48070000 -i 88 --u0`
Command	`i2c-omap35xx-omap4 -p 0x48072000 -i 89 --u1`
Command	`i2c-omap35xx-omap4 -p 0x48060000 -i 93 --u2`
Command	`i2c-omap35xx-omap4 -p 0x4807a000 -i 94 --u3`
Required binaries	**i2c-omap35xx-omap4**
Required libraries	N/A
Source location	**src/hardware/i2c/omap35xx**

Input

Device	INPUT
Command	`io-hid -d usb /dev/usb/io-usb-otg`
Required binaries	**io-hid**
Required libraries	**libinputevents.so**
Source location	**Prebuilt only**

NAND

NOR and NAND share GPMC_nCS0. To use NAND, you need to set the SW5-1 and SW5-2 DIP switches to “ON” and “OFF”, respectively, as show in the figure below:

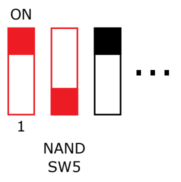


Figure 15: SW5-2 configured for NAND.

Normally, to start the ETFS driver and mount it to **/etfs**, you would use:

```
fs-etfs-jacinto5_micron -D cfg=1,elm=0x48078000,edma=0x43300000^3^0x2003,cache,irq=52 -m/etfs
```

However, if you also want to initialize the NAND partition, you can use a single command to start the ETFS driver and mount it to **/etfs**:

```
# fs-etfs-jacinto5_micron -D cfg=1,elm=0x48078000,edma=0x43300000^3^0x2003,cache,irq=52 -e -m/etfs
```

Notice the `-e` option to erase the device and create an empty filesystem that is ready to use.

Alternately, you can start the ETFS driver, then look after the filesystem:

```
# fs-etfs-jacinto5_micron -D cfg=1,elm=0x48078000,edma=0x43300000^3^0x2003,cache,irq=52
# etfsctl -d /dev/etfs2  -ef
# slay fs-etfs-jacinto5_micron
# fs-etfs-jacinto5_micron -D cfg=1,elm=0x48078000,edma=0x43300000^3^0x2003,cache,irq=52 -m/etfs
```

For more information about routing control, see “On-Board Boot Routing Control.”

Device	NAND
Command	`fs-etfs-jacinto5_micron -D cfg=1,elm=0x48078000,edma=0x43300000^3^0x2003,cache,irq=52 -m/etfs`
Required binaries	**fs-etfs-jacinto5_micron**, **etfsctl**
Required libraries	N/A
Source location	**src/hardware/etfs/nand2048**

This driver is only supported on Jacinto 6 EVM, revisions E1 and more recent.

Network

This BSP supports both the P5 and P6 Ethernet interfaces. By default the dm0 interface is associated with the P5 socket, while the dm1 interface is associated with the P6 socket:

```
io-pkt-v6-hc -d dm814x-j6 -p tcpip random
if_up -p dm0
if_up dm0 up
dhclient -nw dm0
```

If for some reason the MAC address on the Jacinto 6 EVM board is not valid, you can use the command line to assign a valid MAC address to the board so that the Ethernet interfaces will work. For example:

```
# io-pkt-v6-hc -d dm814x-j6 p0mac=001122334455,p1mac=00112233445 -p tcpip
```

Device	P5 and P6 Ethernet interfaces
Command	`io-pkt-v6-hc -d dm814x-j6 -p tcpip random`
Required binaries	**io-pkt-v6-hc**, **ifconfig**, **dhclient**, **nicinfo**, **ping**, **fs-nfs3**, **fs-cifs**, **ifconfig**
Required libraries	**devn-dm814x-j6.so**, **devnp-shim.so** , **libsocket.so**, **libnbutil.so**, **libcrypto.so**, **libssl.so**
Source location	Prebuilt only

QSPI NOR Flash

This BSP supports the S25FL256SAGMFV001 SPI NOR Flash on the Jacinto 6 EVM board.

The **spi-master** resource manager does *not* need to be running for you to be able to use the SPI Flash driver.

Device	QSPI
Command	`devf-j6evm-qspi edma=8,clk=64000000`
Required binaries	**devf-j6evm-qspi**
Required libraries	**devf-j6evm-qspi**
Source location	**src/hardware/flash/boards/j6evm-qspi**

Real-time clock

To program the real-time clock (RTC):

1. Set the system time to the correct value. For example:

   ```
   # date 10 Dec 2013 4 55 pm
   ```

2. Program the RTC hardware with the current OS date and time:

   ```
   # rtc -s hw
   ```

3. Reboot the board (type `shutdown` in the command line).

   When the board boots up, the `rtc hw` instruction in the build file will cause the system to load the time from the RTC.

Device	RTC
Command	`rtc hw`
Required binaries	**rtc**
Required libraries	N/A
Source location	**src/utils/r/rtc**

SATA

Device	SATA
Command	`devb-ahci-omap5 ahci ioport=0x4a140000,irq=86 blk cache=2M cam cache`
Required binaries	**devb-ahci-omap5**
Required libraries	**libcam.so**, **cam-disk.so**, **io-blk.so**, **fs-qnx6.so**
Source location	Prebuilt only

SD/MMC

The Jacinto 6 EVM BSP supports all of the following:

- the eMMC memory (U4 :MTFC4GMVEA-4M WT)
- the micro SD_CARD (P14, MMC1) port on the CPU board
- the MMC3 that is connected to the daughter board through the EXP_P3 expansion connector

A single **devb-sdmmc** instance can manage all three SD/MMC devices and create device nodes **/dev/hd0**, **/dev/hd1** and **/dev/hd/2**. However,the device node's sequence number is assigned based on the relative order in which a device is initialized. This means that, for example, **/dev/hd0** could randomly represents the microSD card, the MMC3 connected to the daughter board, or even the eMMC memory, if you start the driver without explicitly assigning a device node to a device.

Thus, when you start **devb-sdmmc**, you need to specify the name of the device node for each MMC/SD device. See the "Command" entries in the table below.

Devices	eMMC, MMC3, microSD card

Command (generic)	`devb-sdmmc-omap_generic cam pnp blk cache=2M`
Command (MMC1: microSD on the CPU board)	`devb-sdmmc-omap_generic sdio clk=192000000,addr=0x4809C000,irq=115,hc=omap,bs=cd_irq=1187:cd_base=0x4805d000:cd_pin=27 cam pnp blk cache=2M disk name=sd1`
Command (MMC2 for eMMC)	`devb-sdmmc-omap_generic sdio clk=192000000,addr=0x480B4000,irq=118,hc=omap,bs=emmc:bw=8 blk cache=2M disk name=emmc`
Command (MMC3: microSD on the daughter board)	`devb-sdmmc-omap_generic sdio clk=192000000,addr=0x480AD000,irq=126,dma=78,dma=77,hc=omap,bs=nocd cam pnp blk cache=2M cmd disk name=sd2`
Required binaries	**devb-sdmmc-omap_generic**
Required libraries	N/A
Source location	**src/hardware/devb/sdmmc**

Serial

Device	SERIAL (UART1)
Command	`devc-seromap -e -F -b115200 -c48000000/16 0x4806A000^2,104 -u1`
Required binaries	**devc-seromap**
Required libraries	N/A
Source location	**src/hardware/devc/seromap**

SPI

One driver instance is needed for each SPI bus.

Device	SPI1
Command	`spi-master -u 1 -d omap4430 base=0x48098100,irq=97,sdma=1,channel=1`
Required binaries	**spi-master**
Required libraries	**spi-omap4430.so**
Source location	**src/hardware/spi/omap4430**, **src/hardware/spi/master**

SPI2 pads are multiplexed with the UART3 module, which is selected through SEL_UART3_SPI2. Currently we only support SPI1.

Startup

Device	STARTUP
Command	`startup-dra74x-vayu-evm -v -n852,668 -W -G`
Required binaries	**startup-dra74x-vayu-evm**
Required libraries	N/A
Source location	**src/hardware/startup/boards/dra74x**

-a

Set the ABE DPLL at 196 Mhz. When this option is set, you must change the command line option for `io-audio` accordingly.

-b

Enable the Wi-Fi interface.

-G

Enable the NAND interface.

-n

Use a non-spin value that's specified as two values delimited with a comma. For example `-n1234,1234`.

-W

Enable watchdog timer support. Ensure that you start the *watchdog* driver after when you use this option.

Touch screen

For Jacinto 6 EVM, Revision H boards, you must use a different configuration file that specifies to use the `focaltech` driver instead of the `lg-tsc101` driver, that's used for Revision G boards. For more information, see the "*Configuration for Jacinto 6 EVM, Revision H boards*" section in the *Using Multitouch with the Jacinto 6 EVM* chapter of this guide.

Device	10.1-inch LCD with a touch screen controller
Command	`mtouch` (Revision G boards) or `mtouch -c /etc/system/config/mtouch-j6-revh.conf` (Revision H boards)

Required binaries	`mtouch`
Required libraries	**libmtouch-lg-tsc101.so**, **libmtouch-calib.so**, **libinputevents.so**, **libkalman.so**, **libmtouch-focaltech.so**
Source location	**Prebuilt only**

USB Device controller

Device	USB OTG (Device mode)
Command	`io-usb-otg -d dcd-usbumass-omap543x-dwc3 ioport=0x488d0000,irq=110 -n /dev/usb-dcd/io-usb-dcd`
Required binaries	**io-usb-otg**, **devb-umass**, **ulink_ctrl**, **devu-umass_client-block devc-serusb_dcd**
Required libraries	**devu-dcd-usbumass-omap543x-dwc3.so**, **dcd-usbser-omap543x-dwc3.so**, **devu-dcd-usbncm-omap543x-dwc3.so**, **libusbdi.so**
Source location	Prebuilt only

CAUTION: If you want a specific USB port (specified using the `ioport` option) to be started as your USB device controller, ensure that you don't start that port as a USB host controller. In other words, if you wanted to start `ioport=0x488d0000, irq=110` as a device controller, don't specify the same USB port when you start the USB host controller driver.

Below are some examples of how you can use USB OTG.

CDC-ACM (serial) device

1. Start USB device stack:

```
io-usb-otg -d dcd-usbser-omap543x-dwc3 ioport=0x488d0000,irq=110 -n /dev/usb-dcd/io-usb-otg
waitfor /dev/usb-dcd/io-usb-otg 4
waitfor /dev/usb-dcd/devu-dcd-usbser-omap543x-dwc3.so 4
```

2. Start the USB CDC-ACM function driver, and enable USB soft connect:

```
devc-serusb_dcd -e -v -F -s -d iface_list=0,path=/dev/usb-dcd/io-usb-otg
waitfor /dev/serusb1
ulink_ctrl -s /dev/usb-dcd/io-usb-otg -l1
```

Mass storage device

1. 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
```

2. Start device stack:

```
io-usb-otg -d dcd-usbumass-omap543x-dwc3 ioport=0x488d0000,irq=110 -n /dev/usb-dcd/io-usb-otg
waitfor /dev/usb-dcd/io-usb-otg 4
waitfor /dev/usb-dcd/devu-dcd-usbumass-omap543x-dwc3.so 4
```

3. 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 -s /dev/usb-dcd/io-usb-otg
ulink_ctrl -s /dev/usb-dcd/io-usb-otg -l1
```

Example of USB audio device

1. Start USB device stack:

```
io-usb-otg -d dcd-usbaudio-omap543x-dwc3 ioport=0x488d0000,irq=110 -n /dev/usb-dcd/io-usb-otg
waitfor /dev/usb-dcd/io-usb-otg 4
waitfor /dev/usb-dcd/devu-dcd-usbaudio-omap543x-dwc3.so 4
```

2. Start Audio service:

```
io-audio -d usb_dcd path=/dev/usb-dcd/io-usb-otg,sample_rate=48000
ulink_ctrl -s /dev/usb-dcd/io-usb-otg -l1
```

Example of NCM device (Ethernet over USB)

1. Start USB device stack:

```
io-usb-otg -d dcd-usbncm-omap543x-dwc3 ioport=0x488d0000,irq=110 -n /dev/usb-dcd/io-usb-otg
waitfor /dev/usb-dcd/io-usb-otg 4
waitfor /dev/usb-dcd/devu-dcd-usbncm-omap543x-dwc3.so 4
```

2. Start USB NCM function driver and enable USB soft connections:

```
io-pkt-v6-hc -d usbdnet mac=123456789abc,protocol=ncm,path=/dev/usb-dcd/io-usb-otg -ptcpip
ulink_ctrl -s /dev/usb-dcd/io-usb-otg -l 1
if_up -p ncm0
ifconfig ncm0 192.168.10.100
# Alternatively, you can use this command instead of the ifconfig command:
# dhclient -nw ncm0
```

USB (Host mode)

Device	USB OTG (Host mode)
Command	`io-usb-otg -d hcd-omap5-xhci ioport=0x48890000,irq=108,ioport=0x488d0000,irq=110`
Required binaries	**io-usb-otg**, **usb**, **devb-umass**, **devu-dcd-usbumass-omap543x-dwc3.so**
Required libraries	**devu-omap5-xhci.so**, **libusbdi.so**
Source location	Prebuilt only

Watchdog

To enable the watchdog:

1. Modify the build file so that `startup` launches with the `-W` option:

   ```
   startup-dra74x-vayu-evm -W
   ```

2. Launch the watchdog timer utility early on in the boot script:

   ```
   dm814x-wdtkick -e
   ```

Device	WATCHDOG
Command	`dm814x-wdtkick -e -v`
Required binaries	**dm814x-wdtkick**
Required libraries	
Source location	**src/hardware/support/dm814x-wdtkick**

-a

The watchdog timer register's physical base address.

-e

Terminate the watchdog.

-l

The watchdog timer register's size (in bits). For example, 64 indicates a 64-bit size register.

-p

The priority of the watchdog timer. The default is 10 when this option isn't specified.

-t

The kick-time interval in milliseconds. The default is 15000 milliseconds when this option isn't specified.

-v

Enable verbose output.

Wireless (Wilink8)

For Wi-Fi support, you need to:

- Set the SW5-6 DIP switch to “OFF”
- Enable the Wi-Fi interface by specifying the **startup** `-b` option in the build file: `startup-dra74x-vayu-evm -b`.

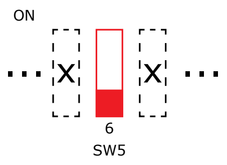


Figure 16: SW5-6 configured to enble Wi-Fi.

Chapter 10
Driver Commands for the AM572x

The tables below provide a summary of driver commands.

The tables list any required additional libraries are binaries that are required. Default required libraries, such as **libc.so** aren't listed, but should be included in your buildfile.

Some of the drivers are commented out in the buildfile. To use these drivers on the target hardware, you may need to uncomment them in the build file, 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)*
- *Real-time Clock (RTC)*
- *USB device controller*
- *USB host controller*
- *SPI*
- *Ethernet*
- *Input*
- *SD/MMC*
- *SATA*
- *Audio*
- *Graphics*

Audio

The AM572x EVM board supports an audio Multichannel Audio Serial Port (McASP). The AM572x EVM board has only one Audio in and one Audio out port.

Audio in input is sent to the digital converter (ADC). Volume range parameters are pre-set to 0 (zero), as they don't affect volume, which is controlled by the Input Gain group.

Device	McASP3
Command	`io-audio -vv -d mcasp-am572x_aic3104`
Required binaries	**io-audio**, **mix_ctl**, **wave**, **waverec**
Required libraries	**deva-ctrl-mcasp-am572x_aic3104.so**, **libasound.so** **libaudio_manager.so.1**, **libcsm.so.1**
Source location	**src/hardware/deva/ctrl/mcasp**

Graphics

Device	GRAPHICS
Command	`screen`
Required binaries	**screen**, **gles1-gears**, **gles2-gears**, **sw-vsync**
Required libraries	For more information, see the *Screen Graphics Subsystem Developer's Guide*.
Source location	**Prebuilt only**

Input

Device	Input
Command	`io-hid -d /dev/io-usb/io-usb`
Required binaries	**io-hid**
Required libraries	**libinputevents.so**
Source location	Prebuilt only

Inter-integrated Circuit (I2C

The AM572x EVM board supports I2C devices on buses 0, 1, 2, 3, and 4. You need to launch an I2C driver instance for each device.

Device	I2C0, I2C2, I2C3, I2C4, I2C
Command	`i2c-omap35xx-omap4 -p 0x48070000 -i 88 -c3 --u0`

Command	`i2c-omap35xx-omap4 -p 0x48072000 -i 89 -c3 --u1`
Command	`i2c-omap35xx-omap4 -p 0x48060000 -i 93 -c3 --u2`
Command	`i2c-omap35xx-omap4 -p 0x4807a000 -i 94 -c3 --u3`
Command	`i2c-omap35xx-omap4 -p 0x4807a000 -i 92 -c3 --u4`
Required binaries	**i2c-omap35xx-omap4**
Required libraries	N/A
Source location	**src/hardware/i2c**

Network

This BSP supports both the P5 Ethernet interface. By default the dm0 interface is associated with the P5 top socket, while the dm1 interface is associated with the P6 bottom socket:

```
io-pkt-v6-hc -d dm814x-j6 -p tcpip random
```

If for some reason the MAC address on the AM572x EVM board isn't valid, you can use the command line to assign a valid MAC address to the board so that the Ethernet interfaces will work. For example:

```
io-pkt-v6-hc -d dm814x-j6 -p tcpip random
waitfor /dev/socket
if_up -p dm0
ifconfig dm0 up
dhclient -nw dm0
```

Device	P5 Ethernet interface
Command	`io-pkt-v6-hc -d dm814x-j6 -p tcpip random`
Required binaries	**io-pkt-v6-hc**, **ifconfig**, **dhclient**
Required libraries	**devn-dm814x-j6.so**, **devnp-asix.so**, **libsocket.so**, **libnbutil.so**, **libcrypto.so**, **libssl.so**, **libncursesw.so**
Source location	src/hardware/devnp

Real-time clock

To program the real-time clock (RTC):

1. Set the system time to the correct value. For example:

   ```
   # date 10 May 2016 4 55 pm
   ```

2. Program the RTC hardware with the current OS date and time:

   ```
   # rtc -s hw
   ```

3. Reboot the board (type `shutdown` in the command line).

When the board boots up, the `rtc hw` instruction in the build file will cause the system to load the time from the RTC.

Device	RTC
Command	`rtc hw`
Required binaries	**rtc**
Required libraries	N/A
Source location	**src/utils/r/rtc**

SATA

The AM572x has an eSata port on the CPU board, and an mSata port on the LCD board. Only one of these ports can be active at any given time.

The J1 jumper on the LCD board controls which port is active:

- If the jumper is connected, the mSATA port is active.
- If the jumper is open, the eSATA port is active.

Device	SATA
Command	`devb-ahci-omap5 ahci ioport=0x4a140000,irq=86 blk cache=2M cam cache`
Required binaries	**ddevb-ahci-omap5**
Required libraries	**libcam.so**, **cam-disk.so**, **io-blk.so**, **fs-qnx6.so**
Source location	Prebuilt only

SD/MMC

The AM572x EVM BSP supports all of the following:

- the eMMC memory (U4 :MTFC4GMVEA-4M WT)
- the SD_CARD (P14, MMC1) port on the CPU board

A single **devb-sdmmc** instance can manage all three SD/MMC devices and create device nodes **/dev/hd0**, **/dev/hd1** and **/dev/hd/2**. However,the device node's sequence number is assigned based on the relative order in which a device is initialized. This means that, for example, **/dev/hd0** could randomly represents the microSD card, or even the eMMC memory, if you start the driver without explicitly assigning a device node to a device.

Thus, when you start **devb-sdmmc**, you need to specify the name of the device node for each MMC/SD device. See the "Command" entries in the table below.

Device	eMMC, microSD card
Command (generic)	`devb-sdmmc-omap_generic cam pnp blk cache=2M`
Command (MMC1: microSD on the CPU board)	`devb-sdmmc-omap_generic sdio clk=192000000,addr=0x4809C000,irq=115,hc=omap,bs=cd_irq=1187:cd_base=0x4805d000:cd_pin=27 cam pnp blk cache=2M disk name=sd1`
Command (MMC2 for eMMC)	`devb-sdmmc-omap_generic sdio clk=192000000,addr=0x480B4000,irq=118,hc=omap,bs=emmc:bw=8 blk cache=2M disk name=emmc`
Required binaries	**devb-sdmmc-omap_generic**
Required libraries	N/A
Source location	**src/hardware/devb/sdmmc**

Serial

Device	SERIAL
Command	`(UART3)devc-seromap -e -F -b115200 -c48000000/16 0x48020000^2,106 –u3` Debug `(UART1)devc-seromap -e -F -b115200 -c48000000/16 0x4806A000^2,104 -u1`
Required binaries	**devc-seromap**, **devc-seromap_hci**
Required libraries	Only Serial-USB: **libusbdi.so**, **libhiddi.so**, **devu-ehci.so**, **devu-ohci.so**, **devu-uhci.so**,**devu-xhci.so**
Source location	**src/hardware/devc/seromap**

SPI

One driver instance is needed for each SPI bus.

Device	SPI1
Command	`spi-master -u 1 -d omap4430 base=0x48098100,irq=97,sdma=1,channel=1`
Required binaries	**spi-master**
Required libraries	**spi-omap4430.so**
Source location	**src/hardware/spi/omap4430**, **src/hardware/spi/master**

Startup

Device	STARTUP
Command	`startup-dra74x-am572x-evm -n852, 668 -W`
Required binaries	**startup-dra74x-am572x-evm**
Required libraries	**libstartup.a**
Source location	**src/hardware/startup/boards/dra74x**

Here are the additional options you can use with this command:

-b

Enable the Wi-Fi interface.

-n

Use a non-spin value that's specified as two values delimited with a comma. For example `-n1234,1234`.

-W

Enable watchdog timer support. Ensure that you start the *watchdog* driver after when you use this option.

USB Device mode

Device	USB OTG (Device mode)
Command	`io-usb-otg -d dcd-usbumass-omap543x-dwc3 ioport=0x488d0000,irq=110 -n /dev/usb-dcd/io-usb-dcd`
Required binaries	**io-usb-otg**, **devb-umass**, **ulink_ctrl**, **devu-umass_client-block**,**devc-serusb_dcd**, **dhclient**, **if_up**
Required libraries	**devu-dcd-usbumass-omap543x-dwc3.so**, **dcd-usbser-omap543x-dwc3.so**, **devu-dcd-usbncm-omap543x-dwc3.so**, **libusbdi.so**
Source location	Prebuilt only

CAUTION: If you want a specific USB port (specified using the `ioport` option) to be started as your USB device controller, ensure that you don't start that port as a USB host controller. In other words, if you wanted to start `ioport=0x488d0000, irq=110` as a device controller, don't specify the same USB port when you start the USB host controller driver.

Below are some examples of how you can use USB OTG.

CDC-ACM (serial) device

1. Start USB device stack:

```
io-usb-otg -d dcd-usbser-omap543x-dwc3 ioport=0x488d0000,irq=110 -n /dev/usb-dcd/io-usb-otg
waitfor /dev/usb-dcd/io-usb-otg 4
waitfor /dev/usb-dcd/devu-dcd-usbser-omap543x-dwc3.so 4
```

2. Start the USB CDC-ACM function driver, and enable USB soft connect:

```
devc-serusb_dcd -e -v -F -s -d iface_list=0,path=/dev/usb-dcd/io-usb-otg
waitfor /dev/serusb1
ulink_ctrl -s /dev/usb-dcd/io-usb-otg -l1
```

Mass storage device

1. 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
```

2. Start device stack:

```
io-usb-otg -d dcd-usbumass-omap543x-dwc3 ioport=0x488d0000,irq=110 -n /dev/usb-dcd/io-usb-otg
waitfor /dev/usb-dcd/io-usb-otg 4
waitfor /dev/usb-dcd/devu-dcd-usbumass-omap543x-dwc3.so 4
```

3. 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 -s /dev/usb-dcd/io-usb-otg
ulink_ctrl -s /dev/usb-dcd/io-usb-otg -l1
```

Example of USB audio device

1. Start USB device stack:

```
io-usb-otg -d dcd-usbaudio-omap543x-dwc3 ioport=0x488d0000,irq=110 -n /dev/usb-dcd/io-usb-otg
waitfor /dev/usb-dcd/io-usb-otg 4
waitfor /dev/usb-dcd/devu-dcd-usbaudio-omap543x-dwc3.so 4
```

2. Start Audio service:

```
io-audio -d usb_dcd path=/dev/usb-dcd/io-usb-otg,sample_rate=48000
ulink_ctrl -s /dev/usb-dcd/io-usb-otg -l1
```

Example of NCM device (Ethernet over USB)

1. Start USB device stack:

```
io-usb-otg -d dcd-usbncm-omap543x-dwc3 ioport=0x488d0000,irq=110 -n /dev/usb-dcd/io-usb-otg
waitfor /dev/usb-dcd/io-usb-otg 4
waitfor /dev/usb-dcd/devu-dcd-usbncm-omap543x-dwc3.so 4
```

2. Start USB NCM function driver and enable USB soft connections:

```
io-pkt-v6-hc -d usbdnet mac=123456789abc,protocol=ncm,path=/dev/usb-dcd/io-usb-otg -ptcpip
ulink_ctrl -s /dev/usb-dcd/io-usb-otg -l 1
```

```
if_up -p ncm0
ifconfig ncm0 192.168.10.100
# Alternatively, you can use this command instead of the ifconfig command:
# dhclient -nw ncm0
```

USB Host mode

Device	USB OTG (Host mode)
Command	`io-usb-otg -d hcd-omap5-xhci ioport=0x48890000,irq=108`
Required binaries	**io-usb-otg**, **usb**
Required libraries	**devu-omap5-xhci.so**, **devu-hcd-omap5-xhci.so**, **libusbdi.so**
Source location	Prebuilt only

Watchdog

To enable the watchdog:

1. Modify the build file so that `startup` launches with the `-W` option:

   ```
   startup-dra74x-am572x-evm -W
   ```

2. Launch the watchdog timer utility early on in the boot script:

   ```
   dm814x-wdtkick -v
   ```

Device	WATCHDOG
Command	`dm814x-wdtkick -e -v`
Required binaries	**dm814x-wdtkick**
Required libraries	N/A
Source location	**src/hardware/support/dm814x-wdtkick**

-a

The watchdog timer register's physical base address.

-e

Terminate the watchdog.

-l

The watchdog timer register's size (in bits). For example, 64 indicates a 64-bit size register.

-p

The priority of the watchdog timer. The default is 10 when this option isn't specified.

-t

The kick-time interval in milliseconds. The default is 15000 milliseconds when this option isn't specified.

-v

Enable verbose output.