BSP User Guide

Texas Instruments DRA7xx (Jacinto 6 and Jacinto 6 Eco) 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: Friday, October 24, 2014

Table of Contents

About this guide

In this guide

The *SDP 6.6 BSP User Guide: Texas Instruments DRA7xx (Jacinto 6 and Jacinto 6 Eco) EVM* contains installation and start-up instructions for the QNX SDP 6.6 Board Support Packages (BSP) for the Texas Instruments DRA72x (Jacinto 6 Eco) EVM and DRA74x/75x (Jacinto 6) Vayu EVM boards. For convenience, this document may refer to these boards simply as the "Jacinto 6 EVM" boards.

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* (p. 9)
What's included in the BSP, and supported host OSs and boards	*About this BSP* (p. 11)
Building and installing this BSP	*Building the BSP* (p. 20)
Driver commands	*Driver commands* (p. 31)
Using the Jacinto 6 board in USB device mode	*USB device mode* (p. 43)

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.

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

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

Note to Windows users

In our documentation, we 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 platform consists of a CPU board and up to three optional daughter boards. These daughter boards run as slaves to the Jacinto 6 CPU board. They can be:

- a JAMR3 application board
- a Vision application board
- a support board for the LCD display.

LCD display support is as follows:

- DRA74x/75x (Jacinto 6) Vayu EVM revisions prior to revision G support a 7-inch screen only.
- DRA74x/75x (Jacinto 6) Vayu EVM revisions G and more recent support a 7-inch screen or a 10.1-inch LCD/TS (recommended).
- DRA72x (Jacinto 6 Eco) EVM supports a 10.1-inch LCD/TS only.

Additional documentation

Before you begin building and installing 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 Guide*
  - *DRA7xx Infotainment Applications Processor Silicon Revision 1.0 Technical Reference Manual*. References to this document are to version G.
  - *DRA72x Infotainment Applications Processor Silicon Revision 1.0 Technical Reference Manual*. References to this document are to version B.
- General information about QNX BSP and instructions for tasks common to all BSPs, see *Building Embedded Systems* available on the the QNX Infocentre. 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 SD card
- how to modify an older BSP to work with the current release

Technical Support

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

Latest version of this BSP

For the most up-to-date version of this user guide, log in to your myQNX account, and download it from the same location as the BSP.

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:

Component	Format	Comments
IPL	Source	To boot from micro SD card or QSPI NOR Flash
Startup	Source	
Kernel	Binary	
Watchdog	Source	
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
I2C driver	Source	
MCASP	Source	
HS MMC/SD	Source	
SLC NAND	Source	
SATA	Binary	
USB 3.0	Binary only	
USB 2.0	Binary only	
Ethernet	Source	
Screen WFD	Source	Delivered in Screen package
Touch screen driver	Source	
Wilink8 Wi-Fi	Binary only	

Component	Format	Comments
User Guide	PDF	This document

Supported OSs and boards

In order to install and use this BSP, you must have installed the QNX Software Development Platform (SDP) 6.6, on either a Windows or Linux Host PC

This BSP supports the following target OS:

- QNX Neutrino® RTOS 6.6

This BSP supports the following boards:

- Texas Instruments DRA72x (Jacinto 6 Eco) EVM
- Texas Instruments DRA74x/75x (Jacinto 6) Vayu EVM.

Testing was done on Vayu EVM board revisions C1 and E1. Camera support was tested on revision G.

For information about JAMR3 expansion boards, see "*Expansion boards*".

Chapter 3
Installation notes

These installation notes describe how to build, install and start this BSP.

Please refer to the *QNX SDP 6.6.0 BSPs* guide, available as part of the QNX Software Development Platform OS Core Components documentation in the QNX Infocentre for detailed instructions how to extract and build a BSP, and how to prepare a bootable, DOS/FAT32 or FAT16 formatted SD card.

Connecting the hardware

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

Connectors

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 SD card slot
3. Console USB port
4. USB 2.0 host port
5. USB OTG port
6. Ethernet port
7. VIN1 port (for camera)

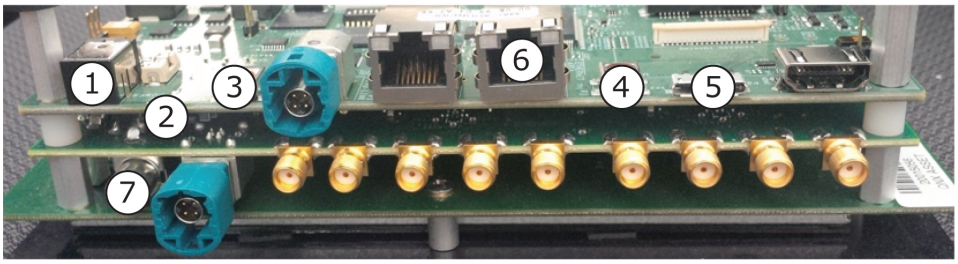

Figure 1: Ports on the Jacinto 6 EVM and the JAMR3 boards.

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

FTDI serial-USB conversion chip

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).

Camera

If you will be using a camera, you also need to connect:

- the camera to the VIN1 port on the JAMR3 extension board (port 7).
- your console to the Jacinto 6 EVM board's J1 mini AB connector (port 3).

Configuring the board switches

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

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

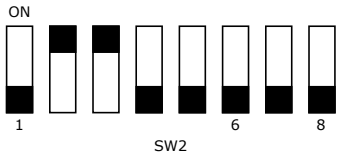

Figure 2: SW2 configuration to boot from the micro SD card (MMC1)

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

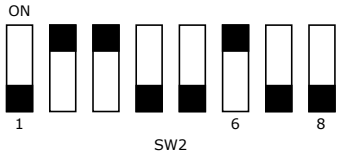

Figure 3: SW2 configuration to boot from QSPI NOR

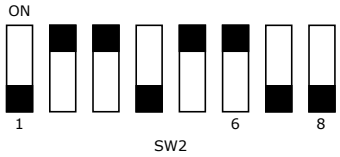

Figure 4: Alternate SW2 configuration to boot from QSPI NOR

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 boards 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 5: 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 Guide*.

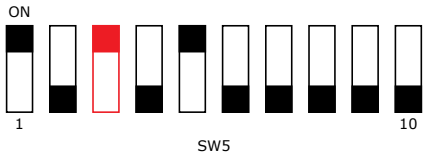

Figure 6: SW5 configuration ensuring the UART3-FT32RQ connection, using parallel NOR

GPMC_nCS0 is used by either the parallel NOR or the NAND Flash memory, depending on which type of memoy is selected. To ensure that GPMC-nCS0 is only ever connected to one type of memory, make sure that only SW5-1 for parallel NOR (U83, S29GL512S10) *or* SW5-2 (NAND (U6,MT29F2G16AADWP) is ever set to “ON”, as shown below.

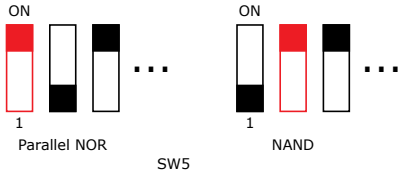

Figure 7: SW5 configurations for either parallel NOR or NAND

Never set both SW5-1 and SW5-2 to “ON” or to “OFF”.

Signaling and operational modes

As the *Vayu EVM CPU Board User Guide* indicates, leave the SW8 switches in their default positions.

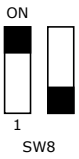

Figure 8: Default SW8 configuration (DRA74x/75x revisions older than rev. G)

The SW8 switch bank applies only to DRA74x/75x (Jacinto 6) Vayu EVM boards. For information about SW8 settings for different board revisions, see "*Driver commands* (p. 31)" and "*Using McASP3* (p. 40)".

Configuring the board switches to use a camera

To use a camera you need to configure switches on both the Jacinto 6 EVM master board and the JAMR3 expansion board.

For other switch settings see "*Configuring the board switches* (p. 15)".

Jacinto 6 EVM switch settings

To use a camera, you need to configure the switch banks on the Jacinto 6 EVM main board, revision G, as shown below:

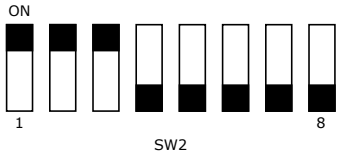

Figure 9: Jacinto 6 EVM SW2 settings for camera

Figure 10: Jacinto 6 EVM SW3 settings for camera

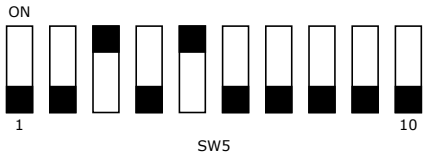

Figure 11: Jacinto 6 EVM SW5 settings for camera

The settings for SW3 and SW5 are the default settings.

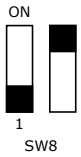

Figure 12: Jacinto 6 Vayu EVM SW8 settings for camera. Note that this setting differs from the setting for boards older than revision G.

The SW8 switch bank applies only to DRA74x/75x (Jacinto 6) Vayu EVM boards.

JAMR3 switch settings

To use a camera, you need to set the following switch banks on the JAMR3 main board, revision B, as shown below:

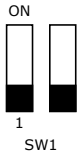

Figure 13: JAMR3 SW1 settings for camera

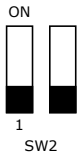

Figure 14: JAMR3 SW2 settings for camera

The setting for SW1 is the default setting.

Building the BSP

You can use the QNX Momentics IDE or the command line to build the BSP.

After you have connected the hardware and configured the board switches, you can use either the QNX Momentics IDE or the command line to build and install the BSP.

Use the IDE to build the BSP

To work with this BSP using the QNX Momentics IDE on a Windows or Linux system, please refer to the instructions for importing and building the BSP available on the *QNX BSP wiki*.

Use the command line to build the BSP

To work with this BSP at the command line on a Windows or Linux machine, simply open a shell, then:

1. Create a new directory for the BSP.
2. Copy the BSP's `.zip` archive into this new directory.

If you plan to work with multiple different BSPs, we recommend creating a top- level BSP directory, then creating subdirectories for each different BSP. The directory you create for each BSP will be that BSP's root directory (***$BSP_ROOT_DIR***).

3. Once the `.zip` archive is in place, extract it using the `unzip` command.

   The `unzip` utility ships with the QNX SDP, for all supported host platforms.
4. When you're ready to build the BSP, from the BSP's root directory, type `make`.

   You may need to source the SDP's environment variables, depending on your tool chain.

The `make` command will build the BSP and create directories, as described above.

Preparing boot images

You can prepare images to boot from either U-Boot or a QNX IPL.

For descriptions of the boot processes, see "*Booting with U-Boot* (p. 27)" and "*Booting with the QNX IPL* (p. 29)."

If you are using a DRA72x (Jacinto 6 Eco) EVM board, when the instructions call for "`dra74x-vayu-evm`", substitute "`dra72x-evm`".

Pre-built image

After you have unzipped the BSP, a pre-built QNX IFS image is available in the BSP's `/images` directory.

This pre-built IFS image is generated by the BSP `make` utility, and is configured for the various BSP device drivers already on you system. When you build the BSP, this image will be overwritten with a new IFS that gets generated by the BSP build process, so you may want to make a copy of it for future reference. If you forget to make backup copy of this image, you can still recover the original prebuilt IFS: simply extract the BSP from the `.zip` archive into a new directory and get the pre-built image form this directory.

Prepare a bootable micro SD card

To enable booting the system from the micro SD card, you need create a DOS FAT32 partition (type 12) on the card. The following is a quick, step-by-step method for formatting the SD card from an Ubuntu Linux terminal:

1. Show the SD card's mountpoint. We'll assume for this example that it is `/dev/sda`:
   ```
   $ mount
   ```
2. Unmount the card:
   ```
   $ umount /dev/sda
   ```
3. Create the partition, entering the default start and end cylinder numbers, and the partition type: 12 or `c`, FAT32:
   ```
   $ sudo fdisk /dev/sda
   > u
   > o
   > n
   > p
   > 1 start cylinder end cylinder
   > a
   > 1
   > t
   > c
   > w
   ```

```
$ sudo mkfs.vfat -F 32 /dev/sda1
$ sync
```

When you have finished preparing your micro SD card, all the boot images should be in the root directory in the FAT partition.

The Texas Instruments ROM Bootloader (RBL) doesn't work for all cylinder/head/sector (CHS) numbers. Though its exact limitations aren't documented, our tests suggest that the classic 63 sectors per track, with 255 heads and less than 1024 cylinders should work.

Preparing U-Boot

We have tested U-Boot only on DRA74x/75x (Jacinto 6) Vayu EVM boards.

U-Boot needs two images to boot the Jacinto 6 EVM board. To prepare to boot from U-Boot, on your host system, copy both images to your formatted micro SD 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 lauched, 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-vayu-evm.bin` or to the micro SD card, or load it from TFTP.

Preparing the QNX boot images

If you need to make changes to the IPL or the IFS, you can make the changes described below, then rebuild the images.

1. The IPL binary: `ipl-dra74x-vayu-evm.bin` comes pre-built in the ***`$BSP`***`/images` directory. If you need to change the 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.7
   $ make clean
   $ make
   $ cd BSP_ROOT_DIR/images
   $ sh mkflashimage.sh
   ```

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

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 micro SD card:

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

4. Insert the micro SD 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 micro SD card.

The startup build file overwrites the `images/vayu-evm.build` file.

Copying the IPL and IFS images to QSPI NOR Flash

The instructions below will partition the 32 MB S25FL256SAGMFV001 QSPI NOR Flash storage into two parts, as follows:

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.

  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 assumes that these images have been copied to the `/tmp` directory on the target.

- Check that the QSPI NOR Flash driver (DRA72x: `devf-ti_qspi`, or DRA74x/75x: `devf-j6evm-qspi`) is running and that the `/dev/fs0` directory exists.

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

1. The IPL must be copied to the 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)
   ```

   Or, use the `dd` file conversion utilty:

   ```
   # 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, select **q** to load the IFS from the QSPI NOR Flash filesystem.

Creating additional filesystem partitions.

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. 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 running to the end of the storage:

   DRA72x

   ```
   # flashctl -p /dev/fs0p0 –o8M -fv
   Formatting device /dev/fs0p0
   # slay devf-ti_qspi
   # devf-ti_qspi
   ```

```
# ls /dev/fs0*
/dev/fs0       /dev/fs0p0     /dev/fs0p1
```

DRA74x/75x

```
# 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`.

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`, whihc 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-vayu-evm.bin /fs/emmc/qnx-ifs
```

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

Booting with U-Boot

You can use U-Boot to boot your system.

We have tested U-Boot only on DRA74x/75x (Jacinto 6) Vayu EVM boards.

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:

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

U-Boot booting the Jacinto 6 EVM board will log something like the following (from a DRA7xx board) on your host system's terminal console:

```
U-Boot SPL 2013.04-rc1-g00344b1-dirty (May 06 2013 - 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 (May 06 2013 - 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.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 6.6.0 on the Texas Instruments J6 EVM(ARMv7 Cortex-A15 core)
Starting UART1 (Console)
starting I2C driver...
Starting QSPI NOR Flash Driver...
#

#
```

You should now be able to test the OS by executing any shell command, or any command residing within the OS image. For example: `ls`.

Booting with the 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 an SD card, or 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 an SD 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.

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, when it starts the IPL will present 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.
```

Choose the location from which you want to load the IFS. For example, provided `qnx-ifs` has been copied to the SD 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 6.6.0 on the Texas Instruments J6 EVM(ARMv7 Cortex-A15 core)
Starting UART1 (Console)
starting I2C driver...
Starting QSPI NOR Flash 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 directory with the IPL `Makefile`:

   DRA72x

   $(IPL)`/boards/dra72x/arm/evm.le.v7`

   DRA74x/75x

   $(IPL)`/boards/dra74x/arm/vayu-evm.le.v7`

2. Open the `Makefile` with a text editor.
3. 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
   ```

4. 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).

Chapter 4
Driver commands

The tables below provide a summary of driver commands.

Some of the drivers are commented out in the default build file in the startup directory. To use the drivers in the target hardware, you'll need to uncomment them in your build file, rebuild the image, and load the image onto the board.

For more information about these and other commands, see the *Neutrino Utilities Reference*.

Startup

Device	STARTUP
DRA72x	
Command	`startup-dra72x-evm`
Required binaries	`startup-dra72x-evm`
Required libraries	
Source location	`src/hardware/startup/boards/dra72x`
DRA72x/75x	
Command	`startup-dra74x-vayu-evm`
Required binaries	`startup-dra74x-vayu-evm`
Required libraries	
Source location	`src/hardware/startup/boards/dra74x`

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 with cause the system to load the time from the RTC.

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

Watchdog

To enable the watchdog:

1. Modify the build file so that `startup` launches with the `-W` option. For example, for a DRA74x/75x board:
   ```
   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`

Serial

Device	SERIAL (UART1)
Command	`devc-seromap -e -F -b115200 -c48000000/16 0x4806A000^2,104 -u1`
Required binaries	`devc-seromap`
Required libraries	
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.

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
DRA72x	
Command	`devf-ti_qspi`
Required binaries	`devf-ti_qspi`
Required libraries	`devf-ti_qspi`
Source location	`src/hardware/flash/boards/ti_qspi`
DRA74x/75x	
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`

I2C

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

Device	I2C1, I2C2, I2C3, I2C4, and I2C5 (DRA72x only)
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 (DRA72x only)	`i2c-omap35xx-omap4 -p 0x4807a000 -i 94 -c3 --u4`
Required binaries	`i2c-omap35xx-omap4`
Required libraries	
Source location	`src/hardware/i2c/omap35xx`

Audio for Bluetooth

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 MCASP1_ENn low to enable the COM8 signals. It is also needed for Wi-Fi support.

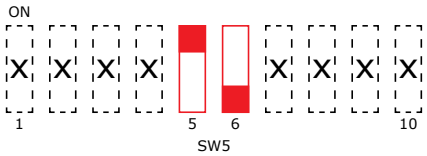

Figure 15: SW5-5 and SW-6 configuration to support Bluetooth

See “*HCI* (p. 38)” for details about starting `devc-seromap_hci`.

Device	McASP
Command	`io-audio -vv -d mcasp-j6 mcasp=6`
Required binaries	`io-audio`, `mix_ctl`, `wave`, `waverec`
Required libraries	`deva-ctrl-mcasp-j6.so`, `libasound.so` , `libasound.so` `libaudio_manager.so.1`, `libcsm.so.1`, `libabe.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.

MMC/SD

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 the 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 micro SD 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 specificy the name of the device node for each MMC/SD device. See the "Command" entries in the table below.

Device	eMMC, MMC3, micro SD card
Command (generic)	`devb-sdmmc-omap_generic cam pnp blk cache=2M`
Command (MMC1: micro SD 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 eMCC)	`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: SD on the daughter board)	`devb-sdmmc-omap_generic sdio clk=192000000,addr=0x480AD000,irq=126,hc=omap,bs=nocd:vdd1_8 cam pnp blk cache=2M cmd disk name=sd2`
Required binaries	`devb-sdmmc-omap_generic`
Required libraries	
Source location	`src/hardware/devb/sdmmc`

NAND

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

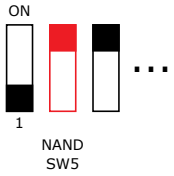

Figure 16: SW5-2 configured for NAND.

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	
Source location	`src/hardware/etfs/nand2048`

This driver is not supported on DRA74x/75x (Jacinto 6) Vayu EVM board revisions prior to E1.

SATA

Device	SATA
Command	`devb-ahci-omap5 ahci ioport=0x4a140000,irq=86 blk cache=2M cam cache`
Required binaries	`devb-ahci-omap5`

Device	SATA
Required libraries	`libcam.so`, `cam-disk.so`, `io-blk.so`, `fs-qnx6.so`
Source location	Prebuilt only

USB

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

Audio McASP3

For detailed information about how to use Audio McASP3, see “Using McASp3”.

Device	McASP3
DRA72x	
Command	`audio -vv -d mcasp-j6_aic3106 mcasp=2,i2c_addr=25,i2c_dev=0`
DRA74x/75x	
Command	`io-audio -vv -d mcasp-j6_aic3106 mcasp=2`
All	
Required binaries	`io-audio`, `mix_ctl`, `wave`, `waverec`
Required libraries	`deva-ctrl-mcasp-j6_aic3106.so`, `libasound.so` `libaudio_manager.so.1`, `libcsm.so.1`, `libabe.so`
Source location	`src/hardware/deva/ctrl/mcasp`

Ethernet

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

```
# io-pkt-v4 -d dm814x-j6 -p tcpip
# dhcp.client –i dm0 &
# dhcp.client –i dm1 &
```

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-v4 -d dm814x-j6 p0mac=001122334455,p1mac=00112233445 -p tcpip
```

Device	P5 and P6 Ethernet interfaces
DRA72x (One Ethernet interface)	
Command	`io-pkt-v4 -d dm814x-j6 deviceindex=0 -p tcpip`
DRA74x/75x (Two Ethernet interfaces)	
Command	`io-pkt-v4 -d dm814x-j6 p0mac=001122334455,p1mac=00112233445 -p tcpip`
Required binaries	`io-pkt-v4`, `ifconfig`, `dhcp.client`
Required libraries	`devn-dm814x-j6.so`, `devnp-shim.so`
Source location	Prebuilt only

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	
Source location	`src/hardware/devc/seromap_hci`

Please note the following about the command-line options:

- The `wl1285.bts` file is a sample Bluetooth script; please 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.

Touch screen

Please note the following before you start the driver for the touch screen:

- The default interrupt vector in the `devi-mxt224` driver for the 7-inch WVGA LCD touch screen is not the one you need for your screen (`100`). Therefore, when you start the driver, use the `-i` option to specify the correct interrupt vector:

  ```
  # devi-mxt224 -PrR800,480 -v touch -i1002 abs
  ```

- To work with the QNX Screen framework, you need to make sure that the following section appears in the `graphics.conf` graphics configuration file. For example, for a DRA74x/75x board:

  ```
  begin mtouch
     driver=devi
     options=height=480,width=800
  end mtouch
  ```

  and that the `devi-mxt224` driver is running *before* Screen is started.

Device	10.1-inch touch screen controller
Command	`devi-lg-tsc-101 -vvvv -PrR1280,800 -v touch abs`
Required binaries	`devi-lg-tsc-101`
Required libraries	
Source location	`src/hardware/devi/lg-tsc-101`

Device	7-inch touch screen controller
Command	`devi-mxt224 -PrR800,480 -v touch -i1002 abs`
Required binaries	`devi-mxt224`
Required libraries	
Source location	`src/hardware/devi/mxt224`

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`.

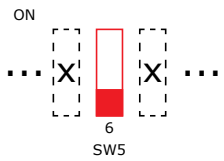

Figure 17: SW5-6 configured to enble Wi-Fi.

Using McASP3

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

Choosing the address for the codec

DRA74x/75x (Jacinto 6) Vayu EVM board revisions F and more recent 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, on boards that support both 7-inch and 10-inch displays, before 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"

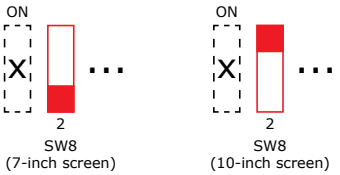

Figure 18: SW8-2 configured for a 7-inch and for a 10-inch screen.

Thus, to use a 10-inch display, start `io-audio` as follows:

```
# io-audio -vv -d mcasp-j6_aic3106 mcasp=2,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 route 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 5
USB device mode

Both the USB 2.0 (mini) and USB 3.0 (micro) ports can work in either host or device mode.

This chapter presents some examples of how to use the Jacinto 6 EVM board's USB ports to use the board as:

- a USB over Ethernet (NCM) device
- an iPod Accessory Protocol 2 (IAP2) device

NCM driver

To use Ethernet over USB (NCM) you need to start drivers on both the Jacinto 6 EVM target board and the host system (or the other device).

Jacinto 6 EVM side

When you use the Jacinto 6 EVM target board as an NCM device, you need to perform the following tasks on the board:

1. Start a driver for the port you will use to connect to the board.

   To start the driver for the USB 2.0 mini port:

   ```
   # io-usb-dcd -dusbncm-omap543x-dwcotg3 ioport=0x488d0000,irq=110
   ```

   To start the driver for the USB 3.0 micro port:

   ```
   # io-usb-dcd -dusbncm-omap543x-dwcotg3 ioport=0x48890000,irq=108
   ```

   The USB driver should create and load the device node: `/dev/io-usb-dcd/devu-usbncm-omap543x-dwcotg3.so`.

2. Start the NCM class driver for USBDNET. You can start it when you mount `io-pkt`:

   ```
   # mount -T io-pkt -o mac=000102030405,protocol=ncm
   devnp-usbdnet.so
   # ifconfig ncm0 192.168.1.1
   ```

   Or in a separate `io-pkt` instance:

   ```
   # io-pkt-v4-hc prefix=/alt -d usbdnet
   mac=000102030405,protocol=ncm -ptcpip
   ```

3. Set the board's USB link state to “connected” so that it is visible as a USB device to the host side:

   ```
   # ulink_ctrl -l1
   ```

   Or disconnect it:

   ```
   # ulink_ctrl –l0
   ```

Host side (or other target)

After you have run `ulink_ctrl -l1` on the Jacinto 6 EVM board, the host side system should be able to see it as an NCM device. For example, on the host system:

```
# usb
Device Address              : 3
Vendor                      : 0x1234 (QNX Software Systems)
Product                     : 0xfff1 (QNX NCM Network Device)
Class                       : 0x02 (Communication)
Subclass                    : 0x0d
Protocol                    : 0x00
```

You should then perform the following tasks on your host system:

1. Start the NCM class driver on the host system from the command line when you start `io-pkt`:

   ```
   # io-pkt-v4-hc -d ncm
   ```

   Or the NCM class driver to the currently running instance of `io-pkt`:

   ```
   # mount -T io-pkt devnp-ncm.so
   ```

2. Use either `dhcp.client` or `ifconfig` to assign an IP address to the NCM interface.

For more information about `dhcp.client` or `ifconfig`, see the QNX Neutrino DSP *Utilities Reference*

IAP2 driver

You can configure your system to support IAP2 or IAP2 NCM so that your Jacinto 6 EVM assumes the role of an iPod device.

Enable PPS

The IAP2 stack needs PPS, so *before* you start IAP2:

1. Edit the BSP's build file to uncomment `pps` and `libpps.so` so that they will be included in the BSP build.
2. Rebuild the BSP.

Customize the `usblauncher` configuration file

To support your Jacinto 6 EVM target board, you need to customize the `usblauncher` configuration file (`etc/usblauncher/rules.lua`):

1. Uncomment the `Host_stack` and `Device_Stack` sections by removing the surrounding `--[[`, `--[[` comment markers.
2. Change to the command line option to invoke the USB driver:

   For the USB 2.0 mini port:

   ```
   Host_Stack = {
   cmd  = 'io-usb -c -d omap5-xhci ioport=0x488d0000,irq=110';
   path = '/dev/io-usb/io-usb';
   }

   Device_Stack = {
   cmd  = 'io-usb-dcd -diap2-omap543x-dwcotg3
   ioport=0x488d0000,irq=110';
   path = '/dev/io-usb-dcd/io-usb';
   }
   ```

   For the USB 3.0 micro port:

   ```
   Host_Stack = {
   cmd  = 'io-usb -c -d omap5-xhci ioport=0x48890000,irq=108';
   path = '/dev/io-usb/io-usb';
   }

   Device_Stack = {
   cmd  = 'io-usb-dcd -diap2-omap543x-dwcotg3
   ioport=0x48890000,irq=108';
   path = '/dev/io-usb-dcd/io-usb';
   }
   ```

3. Specify the correct path to the iPod authentication chip that is needed by `mm-ipod`. This chip may be attached directly to the target, or it may be somewhere on the accessible network. In this example, it is accessed via a QNET node: `path=/net/someone.ott.qnx.com/dev/i2c99`. Make sure you specify the

VID (`0x1234`) and DID (`0xfff1`) so that they match the actual descriptors for the Jacinto 6 EVM board you will use as a peripheral IAP2 device:

```
device(0x1234, 0xfff1) {
class(0xFF, 0xF0, 0x0) {
    driver"mm-ipod -d iap2,config=/etc/mm/iap2.cfg,
    probe,ppsdir=/pps/services/multimedia/iap2
    -a i2c,addr=0x11,path=/net/someone.ott.qnx.com/dev/i2c99
    -t usbdevice,path=/dev/io-usb-dcd/io-usb
    -l /dev/shmem/iap2.log -vvvvvv";
    }
}
```

In the example above, the `driver` command is divided up for clarity. You should enter it on a single line, however.

4. Enable the Vayu EVN board to switch roles with an Apple device. To do this, in the product section of the configuration file, uncomment `RoleSwap_AppleDevice` by removing the leading dashes ("--"):

```
product(0x05AC, 0x1200, 0x12FF) {
 --  RoleSwap_DigitaliPodOut;
RoleSwap_AppleDevice;
--  Probe_iAP2;
}
```

Start PPS and `usblauncher`

To get the services you need started:

1. Start the PPS service by entering the following command on the Jacinto 6 EVM target board, making sure that the PPS path you specify matches the path specified in the `rules.lua` configuration file:

```
# pps
# mkdir -p /pps/qnx
# mkdir -p /pps/services/multimedia
```

2. Plug an iPod into the USB port you are using, and start `usblauncher`:

```
# usblauncher
```

If everything is in order, `usblauncher` should launch `io-usb-dcd` and `mm-ipod`. If you need to debug, you can check the following two files for information:

- `/dev/shmem/iap2.log`
- `/pps/services/multimedia/.all?delta,wait`

3. Use the `iap2cli` command-line utility to send commands to the Apple device. For example:

```
# iap2cli play
Play (rc=0)
# iap2cli next
next (rc=0)
```

Index

J

L

M

N

O

P

Q

R

S

T

U