Release Notes for the QNX Neutrino 6.4.0 BSP for Texas Instruments OMAP5912 OSK Trunk#

System requirements#

Target system#

- QNX Neutrino RTOS 6.4.0
- Board version: Texas Instruments OMAP5912 OSK
- ROM Monitor version UBoot 1.0.0
- 32MB Micron flash

Host development system#

- QNX Momentics 6.4.0
- Terminal emulation program (Qtalk, Momentics IDE Terminal, tip, HyperTerminal, etc.)
- RS-232 serial port
- NULL-modem serial cable
- Ethernet link

System Layout#

The tables below depict the memory layout for the image and for the flash.

Item	Address
OS image loaded at:	0x10010000
OS image begins execution at:	0x10800000
Flash base address	0x00004000

The interrupt vector table can be found in the buildfile located at `src/hardware/startup/boards/osk/build`

Getting Started#

Step 1: Connect your hardware#

Connect the serial cable to the serial port of the OMAP 5912 board and to the first serial port on the host machine (e.g. ser1 on a Neutrino host).

Note:If you have a Neutrino host with a serial mouse, you may have to move the mouse to the second serial port on your host, because some terminal programs require the first serial port.

Step 2: Build the BSP#

You can build a BSP OS image from the source code or the binary components contained in a BSP package.

For instructions about building a BSP OS image, please refer to the chapter Working with a BSP in the Building Embedded Systems manual.

Step 3: Transfer the OS image to the target using the ROM monitor#

1. On your host machine, start your favorite terminal program with these settings:

- Baud: 115200
- Bits: 8
- Stop bits: 1
- Parity: none
- Flow control: none

Note: The OSK board from the factory doesn't have a Boot Loader or ROM Monitor installed. You'll need to initially install the U-Boot boot loader onto the board, using Texas Instruments' Code Composer Studio software, and a suitable JTAG emulator, such as the Spectrum Digital XDS510PP.

The bootloader image is provided with the software that comes with the OSK, and the instructions to install it are described in the OSK_quickstart.doc MS Word document, also provided with the OSK software.

Once you've installed the bootloader, you need to configure it with the TFTP server, etc., which will provide the boot image. At this point, the OSK board is ready to receive a QNX boot image, in raw format.

A raw-format image is simply a binary image, with a header on the beginning, that allows the bootloader to jump to the very beginning of the image (the raw header), where it executes another jump to the first instruction of the image. Later, when the QNX boot image is combined with the native QNX IPL, the boot image will remain in raw format, and the IPL will simply scan past the raw boot header and locate the QNX boot image signature.

2. Start your target. You should see output similar to the following:

```
U-Boot 1.0.0 (Jan 30 2004 - 07:49:02)

U-Boot code: 11000000 -> 11015970  BSS: -> 11019674
DRAM Configuration:
Bank #0: 10000000 32 MB
Micron StrataFlash MT28F128J3 device initialized
Flash: 32 MB
In:    serial
Out:   serial
Err:   serial
```

Use the **`printenv`** command to show the current settings for ipaddr, ethaddr, serverip, etc., use the **`setenv`** command to configure the following parameters:

- serverip
- gatewayip
- netmask
- bootfile
- ipaddr
- ethaddr

3. Once these parameters are configured, use the saveenv command to store your changes. Refer to the U-Boot documentation for more information: www.sourceforge.net/projects/u-boot

After U-boot is configured, boot the ifs-osk.raw image as follows (we'll assume it's in a directory called /xfer/):

```
tftpboot 10010000 /xfer/ifs-osk59xx.raw
```

You should see output similar to the following:

```
OMAP5912 OSK # tftpboot 10010000
TFTP from server 10.0.0.5; our IP address is 10.1.0.28; sending through
gateway 10.0.0.5
Filename '/xfer/ifs-osk59xx.raw'.
Load address: 0x10010000
Loading: #################################################################
         #################################################################
         ############################
done
Bytes transferred = 799464 (c32e8 hex)
OMAP5912 OSK #
```

Now, to run the image, enter:

`go 10010000`

You should see output similar to the following:

```
## Starting application at 0x10010000 ...
Dcache: 256x32 WB
Icache: 512x32
arm926 rev 3 192MHz

System page at phys:101bd000 user:fc404000 kern:fc404000
Starting next program at vfe01c72c
Welcome to QNX Neutrino on the Texas Instruments OMAP Starter Kit (OSK)
Board
#
#
```

4. Once you've established the U-Boot commands to download the image and boot the board, you can also configure U-Boot to save these commands; type the following at the OMAP5912 OSK prompt:

setenv bootcmd tftpboot 10010000 /xfer/ifs-osk.raw\; go 10010000
saveenv

To boot the board, enter:

`bootd`

Step 4: Replace the U-Boot bootloader with a native QNX IPL and OS image in flash#

At some point, you may wish to replace the U-Boot bootloader with the native QNX IPL code. This may be desirable once you've tweaked the OS image exactly the way you want it, and you want the board to boot the image automatically, immediately on power up.

1. Modify your buildfile to generate a binary image.

2. Run the mkflashimage script, inside the /images directory of the BSP. The output file from this script is a combined IPL/OS image called **`ipl-ifs-osk59xx.bin`**. You'll download this file to the board's memory using the bootloader, and then burn the image into the board's flash. The IPL is padded to 16K because if the IPL is scanning for an OS image in flash, it begins scanning at offset 16K in the flash. The mkflashimage script:

- converts the format of ipl-osk (generated in an earlier step) to S-record, and finally to binary
- pads the binary IPL to 16K
- generates a raw format OS image called ifs-osk59xx.raw using the osk.build file
- combines the binary IPL with the raw OS image creating a file called ipl-ifs-osk59xx.bin.

Here is the mkflashimage script:

```
#!/bin/sh
# script to build a binary IPL and boot image for the OMAP 5912 OSK board
set -v

#convert IPL into an S-Record
${QNX_HOST}/usr/bin/ntoarm-objcopy -Osrec ../install/armle/boot/sys/ipl-osk ipl-tmp-osk.srec

#convert S-Record IPL to binary
${QNX_HOST}/usr/bin/ntoarm-objcopy -Obinary ipl-tmp-osk.srec ipl.tmp
mkrec -r -ffull -s16k ipl.tmp > ipl-tmp-osk.bin

#generate binary boot image
make

#combine ipl with boot image
cat ipl-tmp-osk.bin ifs-osk59xx.raw > ipl-ifs-osk59xx.bin
echo "done!!!!!!!"
Instead of starting the image, we'll now transfer the bootable flash image (IPL/OS) to the board. The bootloader can convert to bi
and then write it to flash.
```

3. Boot the board as described above, using U-Boot to TFTP-download the ifs-osk59xx.raw boot image.

4. If it's not already started, start the network driver as follows, substituting your own IP address for x.x.x.x:

io-pkt-v4 -dsmc9000 ioport=0x04800000,irq=200
ifconfig en0:x.x.x.x

5. Start fs-nfs2, establishing an NFS connection to the host machine where your ipl_ifs-osk.bin image resides:

`fs-nfs2 x.x.x.x:/mount_dir /nfs`

Note: Ensure that you can "see" the ipl-ifs-osk59xx.bin file over the NFS connection, because in the next step, the bootloader will be erased. If the programming of the combined IPL and OS image fails or is interrupted, it will be necessary to reprogram U-Boot with Code Composer Studio.

6. Start the flash filesystem driver and erase the first 1MB of flash as follows (erase a larger area if the size of your combined image exceeds 1MB):

devf-generic -s0,32M
flashctl -p/dev/fs0 -l1M -ev

7. Copy ipl-ifs-osk59xx.bin to the flash, as follows:

`cp -V /nfs/ipl-ifs-osk59xx.bin /dev/fs0`

When the copy is complete, you can reboot; it should now boot from the native QNX IPL. You should see output as follows:

```
QNX Neutrino Initial Program Loader for Texas Instruments OSK
Commands:

Press 'S' for serial download, using the 'sendnto' utility
Press 'F' to boot an OS image in flash
```

8. Enter f or F, and the board will boot from the OS image in flash.

You'll see output similar to the following:

```
found image, calling image setup...
image_setup OK, calling image start...

Dcache: 256x32 WB
Icache: 512x32
arm926 rev 3 192MHz
System page at phys:101bd000 user:fc404000 kern:fc404000
Starting next program at vfe01c72c
Welcome to QNX Neutrino on the Texas Instruments OMAP Starter Kit     (OSK)
Board
#
```

If desired, the IPL code can be modified to eliminate the prompt and automatically boot from the flash without user intervention.

To do this, modify the <main.c> file of the IPL source, located under:

$BSP_PATH/osk/src/hardware/ipl/boards/osk/

You can now test the OS, simply by executing any shell builtin command or any command residing within the OS image (e.g. ls).

Creating a flash partition#

1. Enter the following command to start the flash filesystem driver:

```
devf-generic -s0,32M
```

2. Unlock the entire flash, except for the first megabyte:

```
flashctl -p/dev/fs0 -o1M -l31M -U
```

3. Erase the flash, except for the first megabyte:

```
flashctl -p/dev/fs0 -o1M -l31M -ve
```

4. Format the partition:

```
flashctl -p/dev/fs0p0 -o1M -l31M -vf
```

5. Slay, then restart the driver:

```
slay devf-generic
devf-generic -s0,32M &
```

You should now have a /fs0p1 directory automounted.

Driver Command Summary#

The following table summarizes the commands to launch the various drivers.

Component	Buildfile Command	Required Binaries	Required Libraries	Source Location
Startup	`startup-osk`	.	.	`src/hardware/startup/boards/osk`
Serial	`devc-seromap -e -b115200 0xfffb0000^2,46 0xfffb0800^2,47 0xfffb9800^2,15,k`	devc-seromap	.	`src/hardware/devc/seromap`
Flash (NOR)	`devf-generic -s0,32M`	devf-generic flashctl	.	`src/hardware/flash/boards/generic`
Network	`io-pkt-v4 -dsmc9000 ioport=0x04800000,irq=200`	io-pkt-v4 ifconfig mount* ping* cat*	devn-smc9000.so libsocket.so devnp-shim.so	`src/hardware/devn/smc9000`
USB	`osk_usb_init` `io-usb -dohci ioport=0xfffba000,irq=38`	io-usb usb* osk_usb_init	devu-ohci.so libusbdi.so	*QNX SPD 6.4.x* (Binary Only)
I2C	`i2c-omap59xx`	i2c-omap59xx	.	`src/hardware/i2c/omap59xx`
Graphics	`io-graphics -domap5912 photon,xres=240,yres=320,bitpp=16,mode_opts=/etc/omap5912.conf -pphoton`	io-graphics omap5912.conf	devg-omap5912.so	`src/hardware/devg/omap5912`
Audio	`dspmgr -i /proc/boot/app.out -p30 -s320 -b5912 &` `io-audio -d omap i2c_addr=27,rate=16000 &`	dspmgr io-audio	deva-ctrl-omap.so	`src/hardware/deva/ctrl/omap`

EIDE	`devb-eide eide ioport=0x06800000:0x0700000c,irq=21,stride=2,binaryevent`
	`mount -t filesystem / dev/hd0txx / myfs`
		devb-eide	`libcam.so`
			`cam-ata.so`
			`fs-qnx4.so`
				QNX SPD 6.4.x (Binary Only)

Compactflash (SanDisk device)	`devb-eide eide ioport=0x08001000,irq=262,noslave`
	`mount -t filesystem / dev/hd0txx / myfs`
		devb-eide	`libcam.so`
			`cam-ata.so`
			`fs-qnx4.so`
				QNX SPD 6.4.x (Binary Only)

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

USB, I2C and Audio have additional details:

The I2C driver needs to be running before you can use the USB driver and the Audio driver.

To start the USB or audio, you'll need to execute the commands in the order indicated above.

Audio components#

The audio system on the OMAP board consists of hardware and software components. The hardware components are:

- the TLV320AIC23 codec
- the DSP core in the OMAP 5912.

An audio application uses the DSP to transfer audio data to the codec. The software components are:

- The QNX Sound Architecture -- it's composed of io-audio, the asound library, and the OMAP device driver. The deva-ctrl-omap.so driver is the low-level driver that's responsible for communicating with the other components.
- The dspmgr driver -- this driver manages the DSP and is responsible for loading the DSP code onto the DSP and for supporting the DSP LINK interface. This interface allows data to be transferred to and from the DSP.
- The i2c-omap59xx manager -- this manager controls the I2C data bus, and is used by QSA to control the codec.

Note: In order to run the audio driver, you'll need to use the DSP manager to download a DSP image to the OSK board. This DSP image is only available from Texas Instruments. The DSP image must match the Link library version 1.11 provided with this BSP. Please contact your Texas Instruments sales team to obtain it.

Known Issues for this BSP#

- In those instances where the the ROM monitor's MAC address is different from the one you pass in when running **`io-net`** and/or **`io-pkt`** , the host can cache the ROM monitor's address. This can result in a loss of connectivity. **`Workaround`**: If you need to specify a MAC address to **`io-net`** and/or **`io-pkt`**, we recommend that you use the same MAC address that the ROM monitor uses. This will ensure that if the host caches the ROM monitor's MAC address, you'll still be able to communicate with the target. Otherwise you might need to delete the target's arp entry on your host.
- Simply uncommenting the io-graphics line in the buildfile isn't enough to run Photon on the target; this line shows how to run the graphics driver only. For more information on embedding Photon on your board, see the Photon in Embedded Systems appendix of the Photon Programmer's Guide.

To test or evaluate Photon on your target before embedding it you could make the Photon environment accessible to the target to use an NFS or CIFS client on the target depending on the type of server running on your host platform. This example shows

1. Uncommment the following sections from the build file: network, usb and graphics
2. Add the following at the end of the network section:

```
fs-nfs3 10.0.0.1:$QNX_TARGET/cpu/ /
   10.0.0.1:$QNX_TARGET/etc /etc
   10.0.0.1:$QNX_TARGET/usr/photon /usr/photon &
waitfor /usr/bin
waitfor /usr/photon
waitfor /etc
```

Where 10.0.0.1 is the server IP address

Note: The first three lines of the fs-nfs command comprise one command.

3. You can now run the Photon drivers.
To run the graphics driver:

```
/usr/photon/bin/Photon &
waitfor /dev/photon
/usr/photon/bin/io-graphics -d...
```

(See the "Devices supported" chapter in the BSP documentation for details on the graphics driver options).
To use a usb mouse and keyboard:

```
/sbin/io-hid -dusb
/usr/photon/bin/devi-hid kbd mouse &
```

- If you specify the **d** and **p** options for io-graphics, you must put the **d** option before the **d**, or else io-graphics fails. (Ref# 22670)
- The ROM monitor version (UBoot) that comes with the board initializes the CPU frequency to 96MHz, but it should be 192MHz. To avoid timing issues, be sure to get the latest version of the UBoot ROM Monitor. (Ref# 22446)

The IPL sets the frequency to 192MHz.

- Pressing the reset button will not reset the board properly. This is a bug in the ROM monitor. (Ref# 22447)

Workaround: Use the IPL instead, or power down the board then power up again.

- The TCP/IP stack obtains a timer from the process manager. This timer starts at 0. If the TCP/IP stack and a TCP/IP application that tries to connect to a remote host start executing too soon, the TCP/IP stack may apply a time of 0 seconds to ARP cache entry structures. If this occurs, you may end up with a permanent ARP entry (i.e. one that never times out). You can also end up with permanent, incomplete ARP entries that never time out, and that the TCP/IP stack doesn't attempt to resolve. If this happens, your host won't be able to communicate with one or (possibly) more remote hosts (i.e. the ones the TCP/IP application in the OS image is trying to reach).

You can check for permanent ARP entries by running the arp -an command and examining the output. The only permanent en
there shouldn't be any permanent, incomplete entries. If you find a permanent entry that isn't for the IP address of an interface
could be encountering this problem. (Ref# 21395)

Workaround: In the buildfile for your OS image, delay the start of the TCP/IP stack or the first TCP/IP application by at least one second, by using the sleep command (e.g. sleep 1) or some other delay mechanism.

- Audio, Graphics, EIDE and Compact flash have not been tested. DSP manager only build from command line.

- Graphic driver will not work with 6.4.0.