Release Notes for the QNX Neutrino 6.4.0 BSP for Freescale i.MX35 PDK (previously known as Stack Development System (3DS))1.0.0#

System requirements#

Target system

- QNX Neutrino RTOS 6.4.0
- Board version: i.MX35 PDK
- ARM1136 processor
- 128 MB DDR SDRAM
- 64 MB NOR flash
- 2 GB NAND flash

Host development system

- QNX Momentics 6.4.0
- Terminal emulation program (Qtalk, Momentics IDE Terminal, tip, HyperTerminal, etc.)
- RS-232 serial port or a USB-to-serial adapter, and a straight-through serial cable
- Ethernet link

System Layout#

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

Memory layout

Item	Address
OS image loaded at	0x00100000
NOR flash base address	0xA0000000

The interrupt vector table can be found in the buildfile located at `src/hardware/startup/boards/3dsmx35/build`

Getting Started#

Starting Neutrino#

Step 1: Build the BSP
You can build a BSP OS image from the source code. For instructions about building a BSP OS image, please refer to the chapter Working with a BSP in the Building Embedded Systems manual.

Copy or transfer the IFS image into your tftp server's directory.

- When compiling using the command line, the ifs image is in the `images` directory.
- When compiling using the IDE, the IFS image is by default at `/Workspace_root_dir/bsp-freescale-3dsmx35/Images`.

Step 2: Connect your hardware

1. Set up the board to 3-Stack mode. Refer to the manual for the correct default jumper settings to use for the Personality and Debug boards.
2. Connect one end of the serial cable to the CON4 (UART-DCE) serial port on the debug board.
3. Connect the other end of the serial cable to the first available serial port of your host machine (e.g. ser1 on a Neutrino host).
4. Connect an RJ-45 Ethernet cable to the J1 10/100 BaseT Ethernet RJ45 Connector on the debug board.
5. Connect the other end of the Ethernet cable to the Ethernet network where a TFTP server (which you'll use to transfer the boot image) exists.

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

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

Then, apply power to the target. You should see output similar to the following:

```
++... Read from 0x07ee0000-0x07f00000 at 0xa3fe0000: .
... Read from 0x07ed3000-0x07ed4000 at 0xa3fff000: .
**Warning** FLASH configuration checksum error or invalid key
Use 'fconfig -i' to [re]initialize database
PMIC ID: 0x00000010 [Rev: 1.0]
Ethernet FEC MAC address: is not set
hardware reset by POR

Clock input is 24 MHz
Booting from [NOR flash]

Ringo Chip is working in auto mode

LAN92xx Driver version 1.1
SMSC LAN9217: ID = 0x117a REV = 0x0
[Warning] FEC not connect right PHY: ID=fffffc
FEC: [ FULL_DUPLEX ] [ connected ] [ 100M bps ]:
... waiting for BOOTP information
Ethernet eth0: MAC address 00:04:9f:00:94:fa
IP: 192.168.1.202/255.255.255.0, Gateway: 192.168.1.1
Default server: 192.168.1.15

RedBoot(tm) bootstrap and debug environment [ROMRAM]
Non-certified release, version FSL 200814 - built 16:47:08, May 19 2008

Platform: Freescale (i.MX35 )  PASS 1.0 [x32 DDR]
Copyright (C) 2000, 2001, 2002, 2003, 2004 Red Hat, Inc.

RAM: 0x00000000-0x07f00000, [0x00025680-0x07ed1000] available
FLASH: 0xa0000000 - 0xa4000000, 512 blocks of 0x00020000 bytes each.
RedBoot>
#
```

Step 3: Setup the environment

At the RedBoot prompt, issue the **`fconfig`** command to change the current environment.

The current configurations will be displayed; change the configuration if you want.

```
Run script at boot: false
Use BOOTP for network configuration: false
```

```
Gateway IP address: 192.168.1.1
Local IP address: 192.168.1.202
Local IP address mask: 255.255.255.0
Default server IP address: 192.168.1.15
Board specifics: 0
Console baud rate: 115200
Set eth0 network hardware address [MAC]: false
GDB connection port: 9000
Force console for special debug messages: false
Network debug at boot time: false
```

Step 4: Boot the IFS image

Once the above setup is complete, you can run the load command at the RedBoot prompt to download the image: **`load -r -b 0x00100000 -h 192.168.1.15 /tftpboot/ifs-3dsmx35.bin`**

Replace **`192.168.1.15`** with the IP address of your TFTP server and **`/tftpboot/ifs-3dsmx35.bin`** with the path of the image on the TFTP server.

RedBoot will display the follow message and start downloading the boot image:

```
Using default protocol (TFTP)
```

If the image is successfully loaded RedBoot will display:

```
Raw file loaded 0x00100000-0x000c9124, assumed entry at 0x00100000
```

Type **`run`** to jump to startup and boot the IFS image. You should see QNX Neutrino boot, followed by the welcome message on your terminal screen:

```
CPU0: Dcache: 512x32 WB
CPU0: Icache: 512x32
CPU0: VFP 410120b3
CPU0: 4117b363: arm1136 rev 3 399MHz

System page at phys:80010000 user:fc404000 kern:fc404000
Starting next program at vfe03edf0
cpu_startnext: cpu0 -> fe03edf0
coproc_attach(10): replacing fe060044 with fe05f928
coproc_attach(11): replacing fe060044 with fe05f928
Welcome to Neutrino 6.4 on the Freescale i.MX35 3DS (ARM 1136 core) Board
#
```

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

Step 5: Replace the ~Redboot bootloader with a native QNX IPL and OS image in flash

At some point, you may wish to replace the ~Redboot 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.

To replace the ~Redboot bootloader:

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-3dsmx35.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 `mkflashimage` script:

- converts the format of `ipl-3dsmx35` (generated in an earlier step) to binary
- pads the binary IPL to 16K
- combines the binary IPL with the binary OS image creating a file called `ipl-ifs-3dsmx35.bin`

Here is the `mkflashimage` script:

```
#!/bin/sh
#
# script to build a binary IPL image for Freescale i.MX35 3DS board and
# combine with the binary OS image
#

set -v
rm -f ipl-3dsmx35.bin ipl-ifs-3dsmx35.bin

# convert IPL into BINARY format
${QNX_HOST}/usr/bin/ntoarm-objcopy --input-format=elf32-littlearm --output-format=binary -R.data ../install/armle/b

# pad BINARY IPL
mkrec -r -ffull -s16k ipl-tmp-3dsmx35.bin > ipl-3dsmx35.bin

# create a combined IPL and IFS image
cat ipl-3dsmx35.bin ifs-3dsmx35.bin > ipl-ifs-3dsmx35.bin

# clean up temporary files
rm -f *tmp*
```

Instead of starting the image, we'll now transfer the bootable flash image (IPL/OS) to the board. The bootloader can convert to binary format the last image it downloaded and then write it to flash.

3.Boot the board as described above, using ~Redboot to TFTP-download the `ifs-3dsmx35.bin` 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-hc -dmx35 mac=00e02991234e -ptcpip
ifconfig en0 x.x.x.x
```

5.Start fs-nfs2, establishing an NFS connection to the host machine where your `ipl-ifs-3dsmx35.bin` image resides:

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

Ensure that you can "see" the `ipl-ifs-3dsmx35.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 ~Redboot using the Freescale ATK.

6.Start the flash filesystem driver and erase the first 4MB of flash as follows (erase a larger area if the size of your combined image exceeds 4MB):

```
devf-generic -s0xA0000000,64M
flashctl -p/dev/fs0 -o0M -l4M -ve
```

7.Copy `ipl-ifs-3dsmx35.bin` to the flash, as follows:

```
dd if=/nfs/ipl-ifs-3dsmx35.bin of=/dev/fs0 bs=1k seek=0k
```

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 IPL for 3DSMX35 board
Commands:
 d: download image to RAM
 f: scan flash for image
ipl>
```

8.Enter f or F, and the board should boot from the OS image in flash. You'll see the familiar Neutrino welcome message on your terminal screen:

```
Welcome to Neutrino 6.4 on the Freescale i.MX35 3DS (ARM 1136 core) 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/src/hardware/ipl/boards/3dsmx35/

Creating a flash partition#

1. Start the NOR flash filesystem driver by issuing the **`devf-generic -s0xA0000000,64M`** command at the `ksh` prompt, or in the startup script.

2. Prepare the area for the partition. Because the ROM monitor and IPL/OS image are in the first 4MB of flash, you will not want to erase them. Use the -l (length) and -o (offset) options to avoid these areas. Assuming that the ROM monitor and IPL/OS image have a maximum size of 4 MB and we want to create a 16 MB partition:

flashctl -p/dev/fs0 -o4M -l16M -ve

3. Format the partition:

flashctl -p/dev/fs0p0 -o4M -l16M -vf

4. Slay, then restart the driver:

slay devf-generic
devf-generic -s0xA0000000,64M

In this example, you have a 16 MB flash partition starting at the end of the OS image (4 MB offset). You should now have a `/fs0p1` mount on the target to which you can copy files.

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-3dsmx35`	.	.	`src/hardware/startup/boards/3dsmx35`
Serial	`devc-sermx1 -e -F -c100000000`	devc-sermx1	.	`src/hardware/devc/sermx1`

	0x43F90000,45 0x43F94000,32			
I2C	i2c-mx35	i2c-mx35	.	src/hardware/i2c/mx35
SPI	spi-master -d mx35 base=0x43fa4000,irq=14,waitstate=2	spi-master	spi-mx35.so	src/hardware/spi/mx35
FEC Network	i2c-mx35 io-pkt-v4-hc -dmx35 mac=001122334455 -ptcpip	io-pkt-v4-hc ifconfig i2c-mx35 sicinfo* ping*	devn-mx35.so libsocket.so devnp-shim.so	src/hardware/devn/mx35
LAN9217 Network	io-pkt-v4-hc -dsmc9118 ioport=0xb6000000,irq=160 -ptcpip	io-pkt-v4-hc ifconfig mii-tool* ping*	devn-smc9118.so libsocket.so devnp-shim.so	src/hardware/devn/smc9118
USB	io-usb -d ehci-mx31 ioport=0x53ff4500,irq=35 (host 2)	io-usb usb* class drivers	devu-ehci-mx31.so libusbdi.so	*prebuilt only*
NAND	fs-etfs-3dsmx35_2048 -m /fs/etfs	fs-etfs-3dsmx35_2048 etfsctl	.	src/hardware/etfs/nand2048/3dsmx35_20
NOR	devf-generic -s0xA0000000,64M	devf-generic flashctl	.	src/hardware/flash/boards/generic
SDMA	mx35_dma_cfg -c	mx35_dma_cfg	libdma-imx35v2.so	src/utils/m/mx35_dma_cfg
Audio	i2c-mx35 mx35_dma_cfg -c io-audio -d 3dsmx35	io-audio i2c-mx35 mx35_dma_cfg	deva-ctrl-3dsmx35.so libasound.so libdma-imx35v2.so	src/hardware/deva/ctrl/3dsmx35
SPDIF	mx35_dma_cfg -c io-audio -d mx35_spdif	io-audio mx35_dma_cfg	deva-ctrl-mx35_spdif.so libasound.so libdma-imx35v2.so	src/hardware/deva/ctrl/mx35_spdif
Graphics	i2c-mx35 io-display -dvid=0,did=0	io-display i2c-mx35	devg-imx35.so libgf.so.1 libGLES_CM.so.1 libffb.so.2 libm.so.2 libOpenVG.so.1 libOpenVG-G12.so.1	src/hardware/devg/imx35
Touchscreen for PDK	i2c-mx35 pmic_mc13892_cfg 2 devi-mc13892 -r -P -v touch abs -s799,479	devi-mc13892 i2c-mx35 pmic_mc13892_cfg	.	src/hardware/devi/mc13892
Touchscreen for 3DS	i2c-mx35 devi-tsc2007 -r -P tsc2007 abs -s799,479	devi-tsc2007 i2c-mx35	.	src/hardware/devi/tsc2007

SD	`i2c-mx35`
	`devb-mmcsd-3dsmx35`
	`cam quiet`
		devb-mmcsd-3dsmx35
		i2c-mx35
		mount
		umount
			libcam.so
			cam-disk.so
			io-blk.so
			fs-dos.so
			fs-qnx4.so
				`src/hardware/devb/mmcsd`

EIDE	`devb-eide eide ioport=0x500200,irq=15,stride=4,ata`	devb-eide mount umount	libcam.so cam-disk.so io-blk.so fs-dos.so fs-qnx4.so	*momentics*
CAN	`dev-can-mx35 can0`	dev-can-mx35 canctl*	.	`src/hardware/can/mx35`
RTC	`i2c-mx35` `rtc hw`	rtc i2c-mx35 date*	.	`src/utils/r/rtc`

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.

Startup, USB, and Graphics have additional details:

Startup#

`startup-3dsmx35 [`*`startup-options`*`]`

Some modules aren't enabled after boot up, so you need to use command line options to startup to enable them. These options must be passed before any other startup options.

To enable	Use this option	Comment	
ATA	-a	.	
I2C	-c	I2C is required for RTC, FEC, Audio, Graphics and Touchscreen drivers	
SPI	-s	No on-board SPI slaves	
USB Host	-U2	Cannot use CAN when -U2 (USB Host 2) is enabled.	
NAND	-n	.	
FEC	-E	.	
LCD	-L	.	
CAN	-C	Cannot use USB Host 2 when -C (CAN) is enabled.	

CAN#

> **Note**:
> There are two CAN interfaces on the i.MX35 PDK board: CON1 connector(CAN-1) and P2 connector(CAN-2). But we only test the CAN driver on the CAN-1 interface.

- dev-can-mx35 can0

USB#

Running the USB driver requires the following hardware modifications to be performed on the i.MX35 3DS board (not required for PDK board) :

- Remove resistors R1145 and R1148 on personality board
- Add resistors R1154 and R1155 on personality board
- Short jumper J26

> **Note**:
> This hardware modification will disable the access to BT by USB host.

You will want to include class drivers, such as **`devb-umass`** in order to make use of attached USB devices.

Graphics#

The graphics driver in this BSP now contains acceleration for OpenVG rendering. The OpenVG header files and libOpenVG.so are included in this BSP. (prebuilt/usr/include/VG and prebuilt/armle/usr/lib respectively)

OpenVG applications have a dependency on libEGL.so.1. This library is part of the QNX Neutrino RTOS Core Graphics Composition Manger (Patch ID 1347) which can be found at the QNX Download Center. To run the OpenVG accelerated Flash player you will need to obtain QNX Aviage HMI Suite 2.0.0.

Known Issues for This BSP#

- 44 PIN EIDE connector on i.MX35 3DS V1.1 does not supply the correct voltage for the drive (3.3V required 1.8V observed). We tested the EIDE interface using a 40 Pin EIDE dirve, 44 Piin to 40 Pin converter cable, and an external power supply. This problem may have been resolved by Freescale on later boards.
- Although the AK4647 audio driver included in this BSP supports microphone record functionality, microphone record does not work on this board due to a hardware issue.
- The i.MX35 3DS board is populated with 2GB of NAND. Due to the log-structured design of ETFS, startup times may be more (of the order of tens of seconds) for large filesystems.
- The -c option to `etfsctl` does not function correctly and may cause the ETFS driver to crash. **Workaround:** Instead of using the -c option to resume the filesystem after an erase/format, slay and restart the driver.
- The i.MX35 3DS board design does not include battery-backup for the RTC (MC9S08), so the RTC does not retain its contents over power cycles.
- Card insertion and removal detection does not function correctly with the `devb-mmcsd-3dsmx35` driver, and will be investigated for a future release. (Ref# 74724)
- The resolution of the playback/capture positional information returned by the audio driver `deva-ctrl-3dsmx35.so` to the client is limited to the fragment size since we are unable to get the transfer count of the current DMA operation from the SDMA microcode. (Hardware limitation, Ref# 70535)
- The graphics driver `devg-imx35.so` does not link against the graphics libraries when built in the IDE. **Workaround:** Modify the QNX C/C++ Project Properties of the graphics driver: in the Linker tab, under the Extra libraries category, add two entries **ffb** and **disputil** as type Dynamic. Save and rebuild the graphics driver.
- The serial driver `devc-sermx1` doesn't support hardware flow control, and will be investigated for a future release. (Ref# 57988)
- The i.MX35 (ARM1136) processor doesn't support unaligned accesses in hardware. If an application (e.g. pwmopts) tries to access data that isn't aligned on a 32-bit boundary, a bus error will occur. To avoid this memory fault, you can enable the software emulation of unaligned accesses by starting procnto with the -ae option. (Ref# 71252)