Release Notes for the QNX Neutrino 6.4.0 BSP for Freescale i.MX31 ADS 1.0.0#

System requirements#

Target system

- QNX Neutrino RTOS 6.4.0
- Board version: i.MX31 ADS
- ARM1136 processor
- 128 MB DDR SDRAM
- 32 MB burst-mode NOR flash
- 128 MB NAND flash

Host development system

- QNX Momentics 6.4.0
- Terminal emulation program (Qtalk, Momentics IDE Terminal, tip, HyperTerminal, etc.)
- RS-232 serial port and serial cable, or a USB-to-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
Ethernet base address	0xB4020000 (IRQ: 168)
CPLD base address	0xB4000000
Burst flash base address	0xA0000000

Flash layout

Flash layout	Flash start address	Size
RedBoot	0xA0000000	1 MB
Empty	0xA0100000	31 MB - 128 KB
RedBoot configuration data	0xA1FE0000	128 KB

The interrupt vector table can be found in the buildfile located at `src/hardware/startup/boards/imx31/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 is ifs image is in the `images` directory.
- When compiling using the IDE the IFS image is by default at `/Workspace_root_dir/bsp-freescale-mx31ads/Images`.

Step 2: Connect your hardware

1. Set up the board. Refer to the manual for the correct default jumper settings to use.
2. Connect one end of the serial cable to the P7B serial port.
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 T6 (10 Mbit Ethernet RJ-45 connector).
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 0xa1fe0000: .
... Read from 0x07ed3000-0x07ed4000 at 0xa1fff000: .

Clock input is 27 MHz
Booting from [NOR flash]

Ethernet eth0: MAC address 00:01:02:03:03:04
IP: 10.42.103.53/255.255.0.0, Gateway: 10.42.98.230
Default server: 10.42.98.230

RedBoot(tm) bootstrap and debug environment [ROMRAM]
Non-certified release, version FSL 200631 - built 15:19:21, Aug 25 2006

Platform: MX31 ADS (Freescale i.MX31 based) PASS 1.1 [x32 DDR]
Copyright (C) 2000, 2001, 2002, 2003, 2004 Red Hat, Inc.

RAM: 0x00000000-0x07f00000, [0x00013f00-0x07ed1000] available
FLASH: 0xa0000000 - 0xa2000000, 256 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: 10.42.98.230
Local IP address: 10.42.103.53
Local IP address mask: 255.255.0.0
Default server IP address: 10.42.98.230
Board specifics: 0
Console baud rate: 115200
```

```
Set eth0 network hardware address [MAC]: true
eth0 network hardware address [MAC]: 0x00:0x01:0x02:0x03:0x03:0x04
GDB connection port: 9000
Force console for special debug messages: false
Network debug at boot time: false
Update RedBoot non-volatile configuration - continue (y/n)?
```

Type **`y`** to accept the new configuration; RedBoot will write the new configuration to the flash.

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 /xfer/ifs-mx31ads.raw`**

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 0x00100000-0x002300d3, assumed entry at 0x00100000
```

Type **`run 0x00100000`** to jump to startup and boot the IFS image. You should see the QNX Neutrino welcome message on your terminal screen:

```
Welcome to Neutrino on the i.MX31ADS (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).

Once the initial image is running, you can update the OS image using the network and flash drivers. For sample command lines, please see the "Summary of driver commands" section.

Writing the OS image into the Flash using `RedBoot`#

RedBoot is located at the first megabyte offset in flash (i.e. 0xA0000000). The last 128 KB are reserved for RedBoot's configuration data. In the following example, we store a Neutrino boot image of up to 3 MB immediately after RedBoot (at the offset 0xA0100000). From the `RedBoot>` prompt, follow the following procedure to transfer the OS image to the target memory and burn it into flash:

1. Transfer the OS image to RAM, at the `RedBoot>` prompt:

load -r -b 0x00100000 /full_path_to_image/ifs-mx31ads.raw

2. Erase the flash from the offset 1 MB to the offset 4 MB:

fis erase -f 0xa0100000 -l 0x00300000

3. Copy the OS image from RAM (0x00100000) to flash (0xa0100000):

fis write -f 0xa0100000 -b 0x00100000 -l 0x00300000

4. Now you can load the OS image directly from flash on power on, as follows:

mcopy -s 0xa0100000 -d 0x00100000 -l 0x00300000
run 0x00100000

Creating a flash partition#

1. Start the NOR flash filesystem driver by issuing the **`devf-mx31ads`** 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-mx31ads
devf-mx31ads

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-mx31ads`	.	.	`src/hardware/startup/boards/mx31ads`
Serial	`devc-sermx1 -e -F -c66500000 0x43F90000,45 0x43F94000,32`	devc-sermx1	.	`src/hardware/devc/sermx1`
SPI	`spi-master -d mx31 base=0x50010000,irq=13,waitstate=2`	spi-master	spi-mx31.so	`src/hardware/spi/mx31`
SDMA	`mx31_dma_cfg -c`	mx31_dma_cfg}	libdma-imx31v2.so	`src/utils/m/mx31_dma_cfg`
Flash (NOR)	`devf-mx31ads`	devf-mx31ads flashctl	.	`src/hardware/flash/boards/mx31ads`
ETFS Flash (NAND)	`fs-etfs-imx31ads`	fs-etfs-imx31ads etfsctl	.	`src/hardware/etfs/nand512/imx31ads_512`
Network	`io-pkt-v4 -dcrys8900-mx31ads ioport=0xb4020300,irq=168`	io-pkt-v4 ifconfig nicinfo* ping* cat*	devn-crys8900-mx31ads.so libsocket.so devnp-shim.so	`src/hardware/devn/crys8900`
USB	`io-usb -d ehci-mx31`	io-ubs usb*	devu-ehci-mx31.so libusbdi.so	*prebuilt only*

	`ioport=0x43f88300,irq=35` (host 1) `io-usb -d ehci-mx31 ioport=0x43f88500,irq=36` (host 2)		class drivers	
I2C	`i2c-imx31ads`	i2c-imx31ads	.	`src/hardware/i2c/imx31ads`
Audio	`mx31_dma_cfg -c` `io-audio -d imx31ads`	io-audio mx31_dma_cfg	deva-ctrl-imx31ads.so libasound.so libdma-imx31v2.so	`src/hardware/deva/ctrl/imx31ads`
ATA (EIDE)	`devb-eide eide ioport=0x43f8c0a0,irq=15,stride=4,nbslave`	devb-eide	libcam.so cam-disk.so cam-cdrom.so io-blk.so fs-qnx4.so fs-dos.so	*momentics*
Graphics	`io-display -dvid=0,did=0`	io-display	devg-imx31.so libgf.so.1 libGLES_CM.so.1 libffb.so.2 libm.so.2	*prebuilt 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.

Startup, USB, Graphics and Audio have addtional details:

Startup#

`startup-mx31ads [startup-options]`

Some modules are not enabled by default on power 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	
I2C	-c	.	
SPI	-s	.	
ATA	-a	Cannot use USB Host 2 when -a (ATA) is enabled.	
Audio	-s	This enables SPI which is required for Audio	
Graphics	-L	.	
USB Host 1	-U1	.	
USB Host 2	-U2	Cannot be used when -a (ATA) is enabled. The nand-flash board must also be removed.	

For audio support uses:
`startup-mx31ads -s -v`

For graphics support uses:

```
startup-mx31ads -L -v -U1 -le -r0x86000000,0x02000000,1
```

The Power VR MBX Lite accelerator found on the iMX31 requires that a memory aperture be reserved for its use.(-r option) The physical base address and size (in bytes) reserverd here must be specified to the graphics driver using the graphics driver configuration file. See 'Graphics' section below for details.

USB#

USB can be run either on the USB HOST Serial Interface (FS/LS) or the USB HOST ULPI Interface (HS), which are referred to as Host 1 and Host 2 respectively.

To use Host 1:

- Start **`startup-mx31ads`** with the **`-U1`** option.
- Start io-usb as **`io-usb -d ehci-mx31 ioport=0x43f88300,irq=35`**

The USB host port is `j5`.

To use Host 2:

- Start **`startup-mx31ads`** with the **`-U2`** option, and without the **`-a`** option.
- Start io-usb as **`io-usb -d ehci-mx31 ioport=0x43f88500,irq=36`**

> **Note**:
> In order to use Host 2, The ATA module must be disabled and the NAND flash card must be removed .

The USB host port is `j4`.

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

Graphics:#

See the startup section above for the correct startup arguments.

To start the driver.

```
io-display -dvid=0,did=0
```

Note: Starting io-display simply starts the display server. Nothing will appear on the display until a graphical application is run (i.e. vsync or egl-gears).

Required Config Files:

- /etc/system/config/display.conf
- /etc/system/config/imx31.conf

Note: The graphics driver requires a reserved memory aperture for the MBX accelerator to run. (done at startup) The physical base address and size (in bytes) of this aperture must be specified to the driver using the vidbase and vidsize options in imx31.conf. (driver defaults are vidbase=0x86000000,vidsize=0x2000000)

The `/etc/system/config/display.conf` configuration file describes the graphics memory interface settings. It should contain the following for a 240x320 resolution:

```
device {
  drivername=imx31
  vid=0
```

```
  did=0
  display {
    xres=240
    yres=320
    refresh=60
    pixel_format=rgb565
  }
}
```

Audio #

> **Note**:
> Only playback is supported

To enable audio, ensure that the MC13783 board has the following switch settings:

S8 1: OFF, 2: OFF, 3: ON, 4: OFF, 5: OFF, 6: OFF
S7 1: OFF, 2: OFF, 3: OFF, 4: OFF, 5: ON, 6: ON

Then in the build file, launch the sdma configuration utility, **`mx31_dma_cfg -c`**, and the SPI driver before starting **`io-audio`**.

The stereo output jack is located at `J2`.

Known Issues for This BSP#

- Wrong link definition for **/bin/sh** in build file. Currently the link definition is pointing to an invalid executable and will cause other application invoking "sh" to fail. In order to fix the problem you need to modify the build file to point to the proper "sh" application.
  - Existing line : **`[type=link] /bin/sh=/proc/boot/sh`**
  - Replace with : **`[type=link] /bin/sh=/proc/boot/ksh`**
- Any intensive activity on the crys8900 can lead to a crash of **devn-crys8900.so** or could fail to **ping** with large packets on any platform. Dropped packets are due to a known defect with the crys8900 chip.
- The resolution of the playback/capture positional information returned by the audio driver **deva-ctrl-imx31ads.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.
- Running the Open GLES conformance tests will cause the devg-imx31.so to lockup. This behaviour has not appeared with other 3D applications and will be investigated for a future release.
- The serial driver: devc-sermx1 doesn't support hardware flow control, and will be investigated for a future release . (Ref# 57988)