Release Notes of the QNX 6.4.0 BSP for Freescale i.MX21 ADS Trunk#

System requirements#

Target system#

- QNX Neutrino RTOS 6.4.0
- Board version: Freescale i.MX21 ADS
- ARM926EJ-S processor
- 64MB SDRAM
- 32MB burst-mode NOR flash
- 64MB NAND 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.

Memory layout

Item	Address
OS image loaded at:	0x00100000
Ethernet base address	0xCC000300 (IRQ: 203)
Burst flash base address	0xC8000000

Flash layout

Flash layout	Flash start address	Size
RedBoot	0xC8000000	1 MB
Empty	0xC8100000	31 MB - 128 KB
RedBoot configuration data	0xC9FE0000	128 KB

The interrupt vector table can be found in the buildfile located at `src/hardware/startup/boards/imx21ads/build`

Getting Started#

Step 1: Connect your hardware#

- Setup the board. Refer to the manual for the correct default jumper settings to use.
- Connect one end of the serial NULL-modem cable to the P1 serial port.
- Connect the other end of the serial NULL-modem cable to the first available serial port of your host machine (e.g. ser1 on a Neutrino host).

    - 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.
- Connect an RJ-45 Ethernet cable to P9 (10 Mbit Ethernet RJ-45 connector).
- 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.

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#

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

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

1. Make sure the host machine is capable of handling TFTP requests (see inetd documentation for details).
2. Verify the serial link between the host machine and the target.
3. Verify that an Ethernet link exists between the host machine and the target.
4. Apply power to the target board.

You should see output similar to the following:

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

Platform: MX21 ADS (Freescale i.MX21 based) PASS 1.1 [x32 DDR]
Copyright (C) 2000, 2001, 2002, 2003, 2004 Red Hat, Inc.

RAM: 0x00000000-0x03f00000, [0x00013e80-0x03ed1000] available
FLASH: 0xc8000000 - 0xca000000, 256 blocks of 0x00020000 bytes each.
RedBoot>
```

Step 4: Setting up the environment and using TFTP download#

At the RedBoot prompt, issue the following command to change the current environment:

`fconfig`

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.
```

Once the above setup is complete, you can run the load command at the RedBoot prompt to download the image **`load -r -b 0x00100000 /images/ifs-mx21.bin`** RedBoot will display the follow message and start downloading the boot image:

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

If the image is successfully loaded, you will see the following message from the RedBoot prompt:

```
Raw file loaded 0x00100000-0x0020b29f, assumed entry at 0x00100000
```

Type: **`run 0x00100000`**

You should now see the QNX Neutrino welcome message on your terminal screen:

```
Welcome to Neutrino on the i.MX21ADS (ARM 926 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.

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 mkimage script, inside the /images directory of the BSP. The output file from this script is a combined IPL/OS image called ipl-ifs-mx21.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 4K because if the IPL is scanning for an OS image in flash, it begins scanning at offset 4K in the flash.

The mkimage script:

- converts the format of ipl-mx21ads (generated in an earlier step) to binary
- pads the binary IPL to 4K
- generates a raw format OS image called ifs-mx21.bin using the mx21ads.build file
- combines the binary IPL with the raw OS image creating a file called ipl-ifs-mx21.bin.

Here is the mkimage script:

```
#!/bin/sh
#
# Generate a binary image from IPL code for the i.MX21ads
#
# board and combine with the binary OS image
set -v
# Convert IPL into BINARY format
$\{QNX_HOST\}/usr/bin/ntoarm-objcopy -Obinary  ../install/armle/boot/sys/ipl-mx21ads ipl-tmp-mx21.bin
```

```
# Pad BINARY IPL
mkrec -r -ffull  -s0x1000  ipl-tmp-mx21.bin > ipl-mx21-pad.bin
#Cleaning up temporary files
rm  ipl-tmp-mx21.bin
#Combine the BINARY IPL with the BINARY OS Image
cat ipl-mx21-pad.bin ifs-mx21.bin > ipl-ifs-mx21.bin
#Cleaning up temporary files
rm ipl-mx21-pad.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 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-mx21.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 -dcrys8900 ioport=0xcc000300,irq=203,mac=00e02991234e -pttcpip if=en0:x.x.x.x
```

5.Start fs-nfs2, establishing an NFS connection to the host machine where your ipl_mx21.bin image resides:

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

Ensure that you can "see" the ipl-ifs-mx21.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 HAB toolkit.

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 1MB):

```
devf-mx31ads -s0xc8000000,32M
flashctl -p/dev/fs0 -o 0M -l 4M -ve
```

7.Copy ipl-ifs-mx21.bin to the flash, as follows:

```
dd if=/nfs/ipl-ifs-mx21.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 M9328MX21ADS
Commands:
        d: download image to RAM
        f: scan flash for image
ipl>
```

8.Enter f or F, and the board will boot from the OS image in flash. You'll see output similar to the following:

```
Scanning flash at 0xc8001000
found image    @ 0xc8001000
Jumping to startup @ 0xc0101c00

Welcome to QNX Neutrino on the Freescale MX21ADS (ARM 926 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/mx21ads/

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

Putting the OS image into the flash#

1.Generate OS image (using the method in Step 2:Build the BSP).
2.Transfer the OS image to the target memory and burn it into flash:

RedBoot is located at the first megabyte offset in flash (i.e. 0xC8000000). The last 128KB are reserved for RedBoot's configuration data. In the following example we will store a QNX boot image of up to 3MB immediately after RedBoot (at the offset: 0xC8100000).

From the RedBoot prompt, type the following commands:

Transfer the OS image to RAM

RedBoot> load -r -b 0x00100000 /full_path_to_image/ifs-mx21.bin

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

RedBoot> fis erase -f 0xc8100000 -l 0x00300000

In this example, we assume that the OS image size is smaller than 3 MB.Copy the OS image from RAM (0x00100000) to flash (0xa0100000):

RedBoot> fis write -f 0xc8100000 -b 0x00100000 -l 0x00300000

Now you can load OS image directly from flash:

RedBoot> mcopy -s 0xc8100000 -d 0x00100000 -l 0x00300000
RedBoot> run 0x00100000

```
Welcome to Neutrino on the i.MX21ADS (ARM 926 core) Board
#
```

Creating a flash partition#

1.Enter the following command to start the NOR flash filesystem driver:

```
devf-mx31ads
```

2.To prepare the area for the partition, enter the following command: Because the rom monitor and the QNX boot image are in the flash, you may not want to erase them. Use the -l (length) and -o (offset) options to avoid these areas. Assuming that the OS image has a maximum size of 3 MB and we want to create a 16M 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 directory which you can copy files to.

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-mx21ads`	.	.	`src/hardware/startup/boards/mx21ads`
Serial	`devc-sermx1 -e -F -b115200 -c44333333 0x1000a000,20 0x1000b000,19`	devc-sermx1	.	`src/hardware/devc/sermx1`
SPI	`spi-master -d mx21 base=0x1000F000,irq=15,period=32K=2`	spi-master	spi-mx21.so	`src/hardware/spi/mx21`
Flash (NOR)	`devf-mx31ads -s0xc8000000,32M`	devf-mx31ads flashctl	.	`src/hardware/flash/boards/mx31ads`
ETFS Flash (NAND)	`fs-etfs-mx21ads -r4096 -e -m /fs/etfs` (first time use) `fs-etfs-mx21ads -r4096 -m /fs/etfs` (later use)	fs-etfs-imx31ads etfsctl	.	`src/hardware/etfs/nand512/imx31ads_512`
Network	`io-pkt-v4 -dcrys8900 ioport=0xcc000300,irq=203,mac=00e02991234e -ptcpip`	io-pkt-v4 ifconfig ifup* ping* cat*	devn-crys8900.so libsocket.so devnp-shim.so	`src/hardware/devn/crys8900`
USB	`io-usb -d tdi-mx21 ioport=0x10024000,irq=55,verbose=4`	io-usb usb*	devu-tdi-mx21.so libusbdi.so	*prebuilt only*
I2C	`i2c-mx21`	i2c-mx21	.	`src/hardware/i2c/mx21`
Graphics	`io-graphics -dimx21 xres=240,yres=320,bitpp=16,photon,mode_opts=/proc/boot/imx21.conf -pphoton`	io-graphics imx21.conf	devg-imx21.so	`src/hardware/devg/imx21` prebuilt/armle/sbin/imx21.conf

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 has addtional details:

Startup#

startup-mx21ads [startup-options]

Note:

- 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.
- Some of these modules have pin conflicts, so they can't be used at the same time. For more detail, please refer the the Freescale i.MX21ADS user's manual.

To enable	Use this option	
I2C	-c	
SPI	-s1 or -s2or -s3	
USB Host	-U	

Known Issues for this BSP#

- The serial driver: devc-sermx1 doesn't support hardware flow control, and will be investigated for a future release . (Ref# 57988)
- Some warnings may be displayed when compiling this BSP with one or both supported compilers. These warnings are benign and do not affect the functionality of the resulting binaries.
- 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.