Release Notes of the QNX 6.4.0 BSP for Freescale MPC85x0 ADS Trunk#

System requirements#

Target system

- QNX Neutrino RTOS 6.4.0
- Board version: Freescale MPC8560 ADS or Freescale MPC8540 ADS
- 16 MB NOR flash
- ROM Monitor version U-Boot 1.1.2

Host development system

- QNX Momentics 6.4.0
- Terminal emulation program (Qtalk, Momentics IDE Terminal, tip, HyperTerminal, etc.)
- RS-232 serial port and Straight-through 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:	0x00100000
OS image begins execution at:	0x00101e38
Flash base address	0xff000000
Monitor Flash offset	0xff800000
TSEC1 base address	0xe0002400 (IRQ: 13,14,18)
TSEC2 base address	0xe0002500 (IRQ: 19,20,24)
Serial base address (8540)	0xe0004500 (IRQ: 26)
Serial base address (8560)	SCC1 IRQ(104, 112, 113) or SCC2 IRQ(105, 114, 115)

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.

Step 2: Connect your hardware
1. Set up the board.

This BSP is set up for a PCI clock frequency of 33MHz. If you're using this setup, then the clock rates should be:
CPU: 825 MHz
CCB: 330 MHz

DDR: 165 MHz
LBC: 82 MHz

Note: To change the PCI frequency for the Freescale MPC8540 ADS board to 66 MHz:

1. Change the jumper settings as described in the hardware manual.
2. Compile the ROM monitor to use a frequency of 66 MHz.
3. Add the -t66000000 option to the startup command line in your buildfile and rebuild your OS image.

---

2. Connect one end of the serial cable to the P35 serial port 1.

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).

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.

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

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

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

```
U-Boot 1.1.0(pq3-20040423-r1) (Sep 29 2004 - 08:10:20)

Freescale PowerPC
   Core: E500, Version: 2.0, (0x80200020)
   System: 8540, Version: 2.0, (0x80300020)
   Clocks: CPU: 825 MHz, CCB: 330 MHz, DDR: 165 MHz, LBC:  82 MHz
   L1 D-cache 32KB, L1 I-cache 32KB enabled.
Board: ADS
I2C:   ready
DRAM:      SDRAM: 64 MB
128 MB
FLASH: 16 MB
L2 cache enabled: 256KB
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Net:   MOTO ENET0: PHY is Marvell 88E1011S (1410c62)
MOTO ENET1: PHY is Marvell 88E1011S (1410c62)
MOTO ENET2: PHY is Davicom DM9161E (181b881)
MOTO ENET0, MOTO ENET1, MOTO ENET2
Hit any key to stop autoboot: 10
MPC8540ADS=> setenv ipaddr 192.168.200.2
```

Note: The version number for U-Boot is displayed as 1.1.0, but it's really 1.1.2.

Step 3: Setup the environment

On your target, type the following, filling in the appropriate IP addresses and ifs file:

```
MPC8540ADS=> setenv ipaddr 192.168.200.2
MPC8540ADS=> setenv serverip 192.168.200.1
MPC8540ADS=> setenv bootfile ifs-85x0ads.8540.raw
MPC8540ADS=> setenv loadaddr 0x100000
MPC8540ADS=> setenv bootcmd 'tftpboot $loadaddr $bootfile; go $loadaddr'
MPC8540ADS=> setenv bootdelay 2
MPC8540ADS=> saveenv
Saving Environment to Flash...
Un-Protected 1 sectors
Erasing Flash...
flash erase done
Erased 1 sectors
Writing to Flash... done
Protected 1 sectors
```

Step 4: Boot the IFS image

You can use TFTP download (the default) or serial download to transfer the image from your host to the target:

Step 4A: TFTP download

This method requires a raw image, which the buildfile creates by default.

Once the above setup is complete, you can run the load command at the ~MPC8540ADS=> prompt to download the image:

```
MPC8540ADS=> boot
```

At this point you should see the ROM monitor download the boot image, indicated by a series of number signs. You'll also see output similar to this when it completes downloading:

```
Speed: 100, full duplex
Using MOTO ENET0 device
TFTP from server 192.168.200.1; our IP address is 192.168.200.2
Filename 'ifs-85x0ads.8540.raw'
Load address: 0x100000
Loading: #################################################################
   #################################################################
   #################################################################
   #################################################################
   #################################################################
   #################################################################
   #################################################################
   #################################################################
   #################################################################
   #################################################################
   #################################################################
   ###############################################################
done

done
Bytes transferred = 3956736 (3c6000 hex)
## Starting application at 0x00100000 ...
```

Step 4B: Serial download

This method requires an SREC image. You have to modify the buildfile to create this format. Change this:

[virtual=ppcbe,raw]

to this:

[virtual=ppcbe,srec]

Rebuild the image. On your target, type:

```
MPC8540ADS=> setenv loads_echo 0
MPC8540ADS=> saveenv
MPC8540ADS=> loads
```

On your host, copy the image to the serial port that's connected to the board. For example, on a Neutrino host:

```
cp ifs-85x0ads.8540.srec /dev/ser1
```

On a Windows host, you can use Hyperterminal's transfer feature to copy the image as a text file.

Note: The serial line shouldn't already be in use.

At this point, you should see the ROM monitor download the boot image, indicated by a series of dots. You'll also see output similar to this when it finishes downloading:

```
## First Load Addr = 0x00100000
## Last  Load Addr = 0x0023955B
## Total Size      = 0x0013955C = 1283420 Bytes
## Start Addr      = 0x00101E38
MPC8540ADS=>
```

Type: **`go start_addr`**

Note: The start_addr is the startup entry point address. You can find this address from the mkifs utility (you'll need to use the __

For example:

```
mkifs -v -r../install 85x0ads.8540.build ifs-85x0ads.8540.srec
        Offset   Size     Entry   Ramoff Target=Host
        100000    100         0     --- /usr/qnx630/target/qnx6/ppcbe/boot/sys/raw.boot
        100100    100     ----      --- Startup-header
        100200   10108   1022a0     --- /tmp/DAA426034
        ...
```

In this example, 1022a0 is the address to use.

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

```
System page at phys:0000c000 user:0000c000 kern:0000c000
Starting next program at v00133af4
Welcome to QNX Neutrino 6.3 on the PPC 8560ADS 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.

Creating a flash partition#

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

devf-generic -s0xff000000,16M

2. To prepare the area for the partition.

Caution: Do not erase the last 8M -- it contains the ROM monitor. Use the -l (length) option to avoid these areas. To create a 1 MB partition, enter the following command:

flashctl -p/dev/fs0 -l1M -ve

3. Format the partition:

flashctl -p/dev/fs0p0 -l1M -vf

4. Slay, then restart the driver:

slay devf-generic &
devf-generic -s0xff000000,16M &

You should now have a /fs0p0 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-85x0ads`	.	.	`src/hardware/startup/boards/85x0ads`
Serial for 8540 ADS	`devc-ser8250-mpc8540 -e -c330000000 -b115200 0xe0004500,26 0xe0004600,26`	devc-ser8250-mpc8540	.	`src/hardware/devc/ser8250`
Serial for 8560 ADS	`devc-serppc8260 -e -b115200 scc1^1`	devc-serppc8260	.	`src/hardware/devc/serppc8260`
Flash (NOR)	`devf-generic -s0xff000000,16M`	devf-generic flashctl	.	`src/hardware/flash/boards/generic`
PCI	`pci-mpc85xx`	pci-mpc85xx pci	.	`src/hardware/pci/mpc85xx`
Network	`io-pkt-v4-hc -dmpc85xx mac=xxxxxxxxxxxx,verbose -ptcpip`	io-pkt-v4-hc ifconfig	devnp-mpc85xx.so libsocket.so	"Binary form only:" `prebuilt/ppcbe/lib/dll/devnp-mpc85xx.so`
Network:MPC Security Engine (AKA SEC)	`io-pkt-v4-hc -dmpcsec -p tcpip-v6 ipsec -dmpc85xx mac=00112233AABB`	io-pkt-v4-hc ifconfig	devnp-mpc85xx.so devnp-mpcsec.so libsocket.so	"Binary form only:" `prebuilt/ppcbe/lib/dll/devnp-mpcsec.so`

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.

Network:#

without encryption:
`io-pkt-v4-hc -dmpc85xx mac=xxxxxxxxxxxx,verbose -ptcpip`

with encryption in software:
`io-pkt-v4-hc -p tcpip-v6 ipsec -dmpc85xx mac=00112233AABB`

with encryption in hardware:
`io-pkt-v4-hc -dmpcsec.so -p tcpip-v6 ipsec -dmpc85xx.so mac=00112233AABB`

> **Note**:
> The latest sources for `devnp-mpc85xx.so` and `devnp-mpcsec.so` are available from the networking project.

Known Issues#

- 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-v4`** , 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.

- 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 entries listed should be for the IP addresses assigned to your host's interfaces; there shouldn't be any permanent, incomplete entries. If you find a permanent entry that isn't for the IP address of an interface on your host, and you didn't explicitly create a permanent entry, then you 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.
- Some Rev-Pilot boards have problems running Ethernet at half-duplex mode. The second Ethernet port may not work either. (Ref# 23333).The Rev-A boards do not exhibit these problems.**`Workaround`**: Run the Ethernet driver in full-duplex mode on the first Ethernet port.
- When a "break" is detected by the hardware, the Line Status interrupt is asserted. This interrupt is cleared by reading the line status register (which also gives you the cause of the interrupt). From the documentation for the line status register (Section 12.13.9 of the 8540 User Manual): "Note that the ULSRBI is set immediately after ULSR is read if bus remains zero and no mark state followed by a valid new character has been detected." In other words, as long as there's no input to the board after a break, the LS interrupt constantly gets cleared and reasserted resulting in the out-of-interrupt events being generated.

`Workarounds`: There are a few workarounds:

1. Ensure that no breaks are delivered to the target (if one is sent, sending a character to the target will clear the interrupt). See option 2. Note that a break can be also generated to the target (under the conditions of having mis-matched baud rates between the target and host).
2. Always ensure that at least one character is sent to the target immediately following the delivery of a break.
3. Disable the line status interrupt:

In the source file init.c, change line 84: write_8250(port[REG_IE], 0x0f);

to

write_8250(port[REG_IE], 0x0b);

This results in an unfortunate side effect: breaks, framing errors, parity errors, overrun errors, data ready interrupts Rx FIFO errors, Tx empty, Tx holding register empty will not generate Line Status interrupts. However, the LSR is also read with every character received, so processing of line status changes will not be completely disabled, just delayed until a character is received.

- This BSP only tested on Freescale MPC8540 ADS board.

Appendix: CPM memory layout#

The following information applies to the Freescale 8x60 (MPC8260 and MPC8560) processors, which contain the Communications Processor Module (CPM). Neutrino supports various devices found within these processors, namely the SMC channels on the 8260 CPM, as well as the SCC and FCC channels found on both the 8260 and 8560 CPM. The 8x60 CPM module contains 32K of dual-port RAM, which is divided up as follows:

- The first 16K is divided into eight banks, each 2K in size, that can be used for buffer descriptors or data buffers for the various CPM peripherals.
- The next 4K is reserved for parameter RAM, used to store operating parameters for the various CPM devices.
- The remaining 14K is divided up differently, depending on whether the processor is an 8260 or an 8560. On the 8260, the area from offset 20K to 28K is reserved; on the 8560, it's assigned to additional Buffer Descriptor or data buffers. Then, on both the 8260 and 8560, the final 4K is assigned to FCC data.

The Buffer Descriptor Tables and Data Buffers for the SMC and SCC channels are allocated as follows:

- For SMC 1 and 2, bank 7 of the dual port RAM BD/Data area is used for Buffer Descriptor Tables, Data Buffers, and Parameter RAM, as follows:

Item	Space
SMC1 Buffer Descriptor Table	0x100 bytes
SMC1 Data Buffers	0x100 bytes
SMC2 Buffer Descriptor Table	0x100 bytes
SMC2 Data Buffers	0x100 bytes
SMC1 Parameter RAM	0x100 bytes
SMC2 Parameter RAM	0x100 bytes

- For the SCC channels, bank 8 of the dual port RAM BD/Data area is used for Buffer Descriptor Tables and Data buffers. The SCC channels have their own dedicated area for parameter RAM, within the Parameter RAM area. Bank 8 is divided as follows:

Item	Space
SCC1 Buffer Descriptor Table	0x100 bytes

SCC1 Data Buffers	0x100 bytes
SCC2 Buffer Descriptor Table	0x100 bytes
SCC2 Data Buffers	0x100 bytes
SCC3 Buffer Descriptor Table	0x100 bytes
SCC3 Data Buffers	0x100 bytes
SCC4 Buffer Descriptor Table	0x100 bytes
SCC4 Data Buffers	0x100 bytes