Release Notes of the QNX 6.4.0 BSP for AMCC PPC440 EP/GR Evaluation Kit (EVK) Trunk#

System requirements#

Target system

- QNX Neutrino RTOS 6.4.0
- Board version: AMCC PPC440 EP (Yosemite) and AMCC PPC440 GR (Yellowstone)
- 64 MB NOR flash
- ROM Monitor version U-Boot 1.1.4

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:	0x00200000
OS image begins execution at:	0x002028e8
Flash base address	0xFC000000
Monitor Flash offset	0xFD400000
EMAC base addresses	0xEF600E00 / 0xEF600F00 (IRQ: 11,33,34)
Serial base address (UART 0)	0xEF600300 (IRQ: 0)
Serial base address (UART 1)	0xEF600400 (IRQ: 1)
USB (440EP Reference board only)	0xEF601000 (IRQ: 40)
IIC	0xEF600700 (IRQ: 2)
SPI	0xEF600900 (IRQ: 8)

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. Connect one end of the null-modem serial cable to the RS-232 port labeled P2 "UART0" the AMCC Reference board.

2. 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 1.1.4-ga2c95a72 (Jul 28 2006 - 19:33:10)

CPU:   AMCC PowerPC 440GR Rev. B at 533.333 MHz (PLB=133, OPB=66, EBC=66 MHz)
       I2C boot EEPROM enabled
       Internal PCI arbiter disabled, PCI async ext clock used
       32 kB I-Cache 32 kB D-Cache
Board: Yellowstone - AMCC PPC440GR Evaluation Board
I2C:   ready
DRAM:  256 MB
FLASH: 64 MB
PCI:   Bus Dev VenId DevId Class Int
In:    serial
Out:   serial
Err:   serial
Net:   ppc_4xx_eth0, ppc_4xx_eth1

Type "run flash_nfs" to mount root filesystem over NFS

Hit any key to stop autoboot:  0
=>
```

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:

```
=> setenv ipaddr 192.168.200.2
=> setenv serverip 192.168.200.1
=> setenv bootfile root/ifs-ep440c.bin
=> setenv loadaddr 0x200000
=> setenv bootcmd 'tftpboot $loadaddr $bootfile; go $loadaddr'
=> saveenv
Saving Environment to Flash...
Un-Protected 1 sectors
Un-Protected 1 sectors
Erasing Flash...
. done
Erased 1 sectors
Writing to Flash... done
Protected 1 sectors
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:

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 => prompt to download the image:

```
=> boot
```

or

```
=> tftpboot 0x200000; go 0x200000
```

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:

```
=> tftpboot 0x200000; go 0x200000
ENET Speed is 100 Mbps - FULL duplex connection
Using ppc_4xx_eth0 device
TFTP from server 10.42.97.136; our IP address is 10.42.104.42
Filename '/root/ifs-ep440c.bin'.
Load address: 0x200000
Loading: ###################################################################
         ###################################################################
         ###################################################################
         ###################################################################
         ##################################################################
done
Bytes transferred = 1644888 (191958 hex)
## Starting application at 0x00200000 ...

System page at phys:0000b000 user:0000b000 kern:0000b000
Starting next program at v0024615c
Welcome to QNX Neutrino Trunk on the PPC 440 EP/GR Reference 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 -s0xFC000000,64M
```

2. To prepare the area for the partition,, enter the following command:

> **Caution**:
> Be aware of the ROM Monitor offset in flash. These commands will erase the ROM Monitor if not properly watched. The ROM Monitor / Linux image reside in the last 20MB of flash.

flashctl -p/dev/fs0 -l8M -ve

3. Format the partition:

flashctl -p/dev/fs0p0 -l8M -vf

4. Slay, then restart the driver:

slay devf-generic &
devf-generic -s0xFC000000,64M &

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-ep440c`	.	.	`src/hardware/startup/boards/ep440c`
Serial	`devc-ser8250 -e -F -c11059200 -b115200 0xEF600300,0x0 0xEF600400,1`	devc-ser8250	.	`src/hardware/devc/ser8250`
Flash (NOR)	`devf-generic -s0xFC000000,64M`	devf-generic flashctl	.	`src/hardware/flash/boards/generic`
PCI	`pci-ppc440rb`	pci-ppc440rb pci	.	`src/hardware/pci/ppc440rb`
Network	`io-pkt-v4 -dppc405-ep440c mac=001122334455,rmii -ptcpip`	io-pkt-v4-hc ifconfig	devnp-ppc405-ep440c.so libsocket.so devnp-shim.so	`src/hardware/devn/ppc405`
I2C	`i2c-ppc405 -p0xEF600700 -i2`	i2c-ppc405	.	`src/hardware/i2c/ppc405`
SPI static library	.	.	libep440c-spi.a	`src/hardware/spi/ep440c`
SPI test utility	`ep440-spi`	ep440-spi	libep440c-spi.a	`src/hardware/support/ep440-spi`
USB (440EP Reference board only)	`io-usb -dohci-440ep ioport=0xEF601000,irq=40`	io-usb usb	devu-ohci-440ep.so libusbdi.so	*prebuilt only* `/prebuilt/ppcbe/lib/dl/`

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.

FPU support#

This BSP support AMCC PPC440 EP/GR 440 FPU.

SPI#

SPI support is included as a static library (libep440c-spi.a) which needs to be linked into the application which requires SPI functionality.

The ep440-spi sample utility included in this BSP has already been linked with this library.

The following is the expected output of the sample utility when run on the target:

```
# ep440-spi
Welcome to PPC440 SPI API testing program
bit rate is set to 125000 but is read back as 125313
if the error is too big, please set another value
Transmit successful
Transmit successful
Transmit successful
...
```

This utility assumes the following:

- There is a test device connected to the SPI bus which will repeat the data received.
- The bitrate of the test device is set to 125 KHz.
- The polarity of the test device is set to rising/falling.
- The phase of the test device is set to setup/sample.
- The bit order of the test device is MSB.

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.