Release Notes for the QNX Neutrino 6.4.0 BSP for LiPPERT CoreExpress-ECO 1.0.0#

System requirements#

Target system#

- QNX Neutrino RTOS 6.4.0
- Board version: LiPPERT CoreExpress-ECO evaluation kit.

> **Note**:
> Please check CoreExpress-ECO serials against serial numbers of the kit QNX BSP was tested on:
> carrier board serial number 810-0007-00, module serial 813-0002-XX.

Host development system#

- QNX Momentics 6.4.0
- Terminal emulation program (Qtalk, Momentics IDE Terminal, tip, HyperTerminal, minicom, etc.)
- RS-232 serial port
- NULL-modem serial cable
- Ethernet link

Getting Started#

Step 1: Connect your hardware#

1. Connect the serial cable to the serial port of the LiPPERT CoreExpress-ECO board and to the first serial port on the host machine (e.g. ser1 on a Neutrino host).

On your host, run your terminal application with the following configuration:

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

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.

The startup-coreexpress-eco will add 1Gb of system memory by default. If needed edit the `src/hardware/startup/boards/coreexpress-eco/main.c` file
for adding proper amount of system memory and setting MTRR table with write-combining range for graphical driver.

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 - (BIOS Boot): Transfer the OS image to a bootable USB memory stick#

1. On a Neutrino host, plug the USB key in a USB port (the following steps assume the USB key device name is /dev/hd1, but it may be /dev/umass0 for an unformatted key). At a command prompt:
2. fdisk /dev/hd1
3. Delete existing partitions (hit d, then s)
4. Create a new type 79 partition (hit c, then 79, then 0, then numCylinders-1, then s)
5. Make the partition bootable (hit b, then s)
6. Slay and restart devb-umass
7. dinit /dev/hd1t79
8. dloader /dev/hd1 pc1
9. dloader /dev/hd1t79 pc2
10. Slay and restart devb-umass

Note that steps 1-10 only need to be done once.

11. mount /dev/hd1t79 /stick
12. cp -V ifs-coreexpress-eco.raw /stick/.boot

> **Note**:
> The IFS filename will be different if compiled from the IDE, ex: bsp-lippert-CoreExpress-ECO.ifs

13. Ensure the LiPPERT CoreExpress BIOS is configured to boot from the USB stick
14. Plug the USB key in one of the LiPPERT CoreExpress-ECO USB ports
15. Restart the LiPPERT CoreExpress-ECO

Step 3 - (Fastboot)#

1. Ensure your image is small enough to fit in EEPROM with the microcode (images/CMC.bin) at 0xD0000. If you have an EEPROM chip of 1MB,
that means the entire image must fit in the first 0xD0000 (832K). A larger EEPROM chip will allow you to have a larger image,
so long as the microcode is located at 0xD0000 and the IPL is at 0xF0000.
2. Go to the images subdirectory of the BSP and run:
3. **`make ifs-fastboot.raw`**

Note that this invokes the mkrom.sh script which was written for a 1MB EEPROM chip, you'll have to change it for larger chips. Its contents are as follows:

```
#!/bin/sh
# This script assumes an EEPROM size of 1MB
# Example usage: ./mkrom.sh ifs-coreexpress_nobios.bin

if [ "$1" == "" ]; then
   echo "Must specify the image file"
   exit 1;
fi

#pad it out to 0xD0000
mkrec -s832k -r -ffull $1 > tmp1.bin

#tack on 64k microcode
cat tmp1.bin CMC_lippert.bin > tmp2.bin

#pad image out to 0xF0000
mkrec -s960k -ffull -r tmp2.bin > tmp3.bin

#tack on 64k IPL
cat tmp3.bin ../install/x86/boot/sys/ipl-coreexpress-eco > fastboot.rom
```

```
rm tmp1.bin tmp2.bin tmp3.bin
```

4. This will create a fastboot.rom image.
4a. Image can be written to the EEPROM using an EEPROM burner
4b. Image can be written to the EEPROM using devf-fwh-poulsbo driver. Run following commands:

 # devf-fwh-poulsbo
 # flashctl -p /dev/fs0 -U
 # flashctl -p /dev/fs0 -e
 # cp -V fastboot.rom /dev/fs0

> **Note**:
> Be sure to backup CoreExpress-ECO BIOS or use another EEPROM for programming Fastboot QNX image.

5. Boot the LiPPERT CoreExpress-ECO with the BIOS disabled
6. The IPL will ask you if you want to scan flash for an image or receive an image over the serial link.
6a. Select "f" to boot the image in EEPPROM
6b. Select "d" to receive an image over the serial link. Then run **`sendnto -d/dev/ser1 -b115200 ifs-coreexpress-eco.raw`** from the host connected to the target.

> **Note**:
> For booting a Fastboot based image, the image must be built with the nobios option, ie:
> **`[virtual=x86,nobios +compress]`**

Driver Command Summary#

The driver command lines below are specific to the LiPPERT CoreExpress-ECO board. See the online docs for each driver for additional command-line options and other details.

> **Note**:
> The BSP comes with 3 example build files:
>
> 1. `coreexpress-eco.fastboot.build` is an example Fastboot build file which produces an image under 832KB in size and uses a USB and NFS filesystem to boot to a fully functional QNX OS.
> 2. `coreexpress-eco.bios.build` is an example BIOS build file which produces a self-contained bootable image that uses the BIOS for startup and PCI resource allocation.
> 3. `coreexpress-eco.build` is an example BIOS build file which produces a self-contained bootable image that uses the board specific startup and Poulsbo PCI server for PCI resource allocation. This build file can be useful for developers because it uses the same startup and PCI code as the Fastboot image but boots from the BIOS.

Component	Buildfile Command	Required Binaries	Required Libraries	Source Location
Startup	`startup-coreexpress-eco`	.	.	`src/hardware/startup/boards/coreexpress-eco`
Serial	`devc-ser8250 -F -e -b115200`	devc-ser8250	.	`src/hardware/devc/ser8250`

	`-u1 0xde00,3 -u2 0xdd00,3`			
PCI	`pci-poulsbo &`	pci-poulsbo	.	`src/hardware/pci/poulsbo`
Network	`io-pkt-v4-hc -dspeedo -ptcpip`	io-pkt-v4-hc ifconfig	devnp-speedo.so libsocket.so devnp-shim.so	.}
USB	`io-usb -duhci -dehci`	io-usb usb	devu-ohci.so devu-ehci.so libusbdi.so	QNX SPD 6.4.x (Binary Only)
SD/MMC	devb-mmcsd blk cache=1M mmcsd vid=0x8086,did=0x811c	devb-mmcsd	libcam.so io-blk.so cam-disk.so fs-qnx4.so	`src/hardware/devb/mmcsd`
Audio	`io-audio -dintel_hda`	io-audio mix_ctl wave waverec	libasound.so deva-ctrl-intel_hda.so deva-mixer-hda.so deva-util-restore.so	`src/hardware/deva/ctrl/intel_hda`
Graphics	`io-display -d vid=0x8086,did=0x8108`	io-display	devg-poulsbo.so libgf.so.1 libGLES_CL.so.1 libffb.so.2\libm.so.2	`src/hardware/devg/poulsbo`
EIDE	`devb-eide`	devb-eide	libcam.so.2 io-blk.so cam-disk.so fs-qnx4.so fs-dos.so	QNX SPD 6.4.x (Binary Only)
Flash	`devf-fwh-poulsbo`	devf-fwh-poulsbo	.	`src/hardware/flash/boards/fwh`

Note:
Some of these drivers can be 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.

Additional Notes for Graphics#

1. See the `prebuilt/etc/system/config/poulsbo.conf` and `prebuilt/etc/system/config/display.conf` files for configuration of the graphics driver.

Additional Notes for Flash#

1. The flash driver currently only supports the SST 49LF00xA devices. This limitation is because other devices were found to have their polling features behave differently than documented.

Known Issues#

Compile failures when building from IDE 4.5 (ref #67407)#

Workaround: See Building with the IDE 4.5
note: This issue has been resolved within IDE 4.6