# Release Notes for the QNX Neutrino 6.4.0 BSP for AMD Geode LXDB800 1.0.0#

## System requirements#

### Target system

- QNX Neutrino RTOS 6.4.0
- Board version: AMD Geode LXDB800
- X86 processor
- Hard Drive
- CD-ROM drive
- monitor
- PS/2 keyboard and mouse
- Ethernet connection

### 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: | 0x400000 |
| OS image begins execution at: | 0x400000 |

## Getting Started#

Since the AMD Geode LXDB800 has an x86 CPU, and contains a BIOS, it is possible to simply boot the board from a QNX Momentics CD, and install QNX directly to a hard drive. In fact, this is the first step in the process of installing this BSP. The BSP itself provides additional QNX device drivers (not currently shipped with the OS) for specific peripherals found on the LXDB800 board. The BSP also provides a sample build file for generating an OS image which is custom-tailored to the LXDB800 board. Although it is possible to use the LXDB800 as both a host development platform and as a target system, for the purposes of this document, we'll assume that you are using some other system as a development host, and are treating the LXDB800 board as a target only. Installing QNX Neutrino to the target's hard drive prior to installing the BSP simply facilitates the process of installing target OS images, and provides a more full-featured runtime image.

The basic method described in this document to use the LXDB800 BSP is as follows:

- using a standard QNX Momentics 6.4.0 CD-ROM, install QNX Neutrino to a hard drive on the LXDB800 board
- install the AMD Geode LXDB800 BSP on your host development system

- on the host system, build the LXDB800 BSP, and generate the OS image intended for the target
- establish a network connection between the host system and the LXDB800 board
- copy the generated OS image to the /.boot boot image of the hard drive on the LXDB800 board
- reboot the LXDB800 board, and select the generated OS image as the boot image

For information about hardware configurations or BIOS settings, refer to the documentation for your LXDB800 board.

**Starting Neutrino#**

**Step 1: Connect your hardware**

Connect a standard ATX power supply, monitor, CD-ROM drive, hard drive, ethernet cable, and PS/2 keyboard and mouse to the target system. Power up the system, and configure the BIOS to boot from CD-ROM.

**Step 2: Install QNX Neutrino 6.4.0 on the target**

Using a QNX Momentics 6.4.0 installation CD, install the QNX Neutrino Realtime Operating System v6.4.0 to the hard drive that is connected to the LXDB800 board. This will provide a base installation of the standard x86 QNX Neutrino device drivers and utilities. It is not necessary to install the development tools or the IDE. Once the installation is complete, verify that the LXDB800 board can boot into QNX Neutrino from the hard drive.

**Note**:
Instead of performing a new installation of QNX Neutrino on the target's hard drive, it is also possible to simply connect the LXDB800 to a hard drive which already has QNX Neutrino Realtime Operating System v6.4.0 installed, and boot the LXDB800 board from that hard drive.

**Step 3: 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 ifs image is in the **`images`** directory.
- When compiling using the IDE, the ifs image is by default at **`/Workspace_root_dir/bsp-amd_geode_lxdb800/images`**.

The default build file provided with this BSP will generate an OS image which does the following:

- boots QNX Neutrino
- starts the devb-eide disk driver, and mounts the QNX filesystem on the hard drive as **`/`**.
- starts the devn-rtl8169.so network driver
- starts the devg-lx800.so graphics driver
- starts the deva-ctrl-geode_cs5536.so audio driver
- starts the Photon MicroGUI

This build file can be modified as desired for your particular needs.

**Note**:
The default build file will attempt to auto-mount the QNX hard disk partition hd0t179 as **`/`**. If the hard drive that you have connected to the target board has a QNX partition with a different name, for example hd0t79 or hd0t78, you will need to modify the command line for devb-eide in the **`$BSP_WORKING_DIR/images/bios.lxdb800.build`** file.

**Step 4: Transfer the OS image to the target**

- on the target, start the network driver as follows (substitute your IP address for x.x.x.x): **`io-pkt-v4 -drtl8169 -ptcpip`**
- establish an FTP or NFS connection to your host development machine
- enter the $BSP_WORKING_DIR/images/ directory of the host
- transfer the amd-geode-lxdb800.ifs file to the target
- on the target, copy the amd-geode-lxdb800.ifs file to **`/.boot`**: **`cp amd-geode-lxdb800.ifs /.boot`**
- reboot the target system, and when prompted by the loader, select the generated OS image as the boot image

The LXDB800 board will start various device drivers, boot into the Photon MicroGUI, and start a pterm session. You can now test the OS simply by executing any shell builtin command or any command residing within the OS image (e.g. ls).

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

```
QNX v1.2a Boot Loader: amd-geode-lxdb800.ifs .....
Welcome to QNX Neturino 6.4.0 on the AMD Geode LXDB800 Board
```

## 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-bios` | . | . | `src/hardware/startup/boards/bios` |
| Serial | `devc-ser8250 -b115200` | devc-ser8250 | . | `src/hardware/devc/ser8250` |
| PCI | `pci-bios` | pci-bios<br>pci | . | `src/hardware/pci/bios` |
| Network | `io-pkt-v4 -drtl8169 -ptcpip` | io-pkt-v4<br>ifconfig<br>nicinfo<br>ping | devn-rtl8169.so<br>libsocket.so<br>devnp-shim.so | `src/hardware/devn/rtl8169` |
| Audio | `io-audio -d geode_cs5536` | io-audio<br>mix_ctl<br>wave | deva-ctrl-geode_cs5536.so | `src/hardware/deva/ctrl/geode_cs5536` |
| Graphics | `io-display -dvid=0x1022,did=0x2081` | io-display | devg-lx800.so | `src/hardware/devg/lx800` |

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.

## Known Issues for This BSP#

- At high baud rates, the serial driver devc-ser8250 may drop characters; to work around this issue, enable the receive fifo with a trigger level of 4 or greater, using the -t option, as shown in the example

below (refer to the QNX Neutrino Utilities Reference for additional information on options to devc-ser8250):**`devc-ser8250 -e -F -b115200 -t4`**

- There is an undocumented limitation on the size of the OS image that can be properly loaded on x86 systems (ref. PR#45838). Ensure that OS images generated do not exceed 3M (uncompressed) in size.
- In those instances where the the ROM monitor's MAC address is different from the one you pass in when running 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-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.