Implementing PXE Boot using Intel® BLDK for Intel® Atom™ Processor based Boards

March 2012
Executive Summary

Intel provides an EFI-standard based Boot Loader Development Kit (BLDK) for various Intel® Atom™ processors. This white paper discusses how to modify the Intel® BLDK code base to support network booting (PXE boot). The resulting BLDK image is programmed onto an Intel® Atom™ Processor E6xx Series with Intel® Platform Controller Hub EG20T evaluation board codenamed Crown Bay.

The process involves installing the BLDK Integrated Development Environment (IDE) on a computer running Microsoft* Windows 7* to create the modified BLDK image. It also involves setting up a host computer from which the target unit remotely boots. The target unit boots into a MeeGo* image installed on the host computer running Fedora 16* as the server OS.

The final image boots the machine over the network via PXE boot. In this configuration a Linux* kernel is downloaded onto the target machine’s volatile memory and a root file system residing on the host is mounted over an NFS link. This layout enables a versatile environment for developing embedded systems.

The resulting BLDK image is programmed onto an Intel® Atom™ Processor E6xx Series with Intel® Platform Controller Hub EG20T evaluation board codenamed Crown Bay.

Although this paper is based on a particular Linux release and the Elilo operating system loader, the same principles can be
applied to other UEFI compliant operating systems and other loaders.

This white paper aims to lead the user through the process step-by-step and share other useful information during this process. However, experience compiling Linux-kernels and working with Linux in general is nearly mandatory in order to complete this guide.


§
### Contents

1. Getting Started ........................................................................................................6
   1.1 Notations and Terminology ........................................................................6
   1.2 Reference Documents .................................................................................8
   1.3 Software Resources ..................................................................................8
      1.3.1 SDKs, Drivers, Source code .................................................................9
   1.4 Hardware resources .....................................................................................9

2. Setup and configuration .........................................................................................10
   2.1 Development System set up instructions ......................................................10
      2.1.1 Installing Microsoft* Windows* 7 .........................................................11
      2.1.2 Preparing Intel® BLDK Environment under Windows* .......................11
      2.1.3 Installing Fedora* 16 .............................................................................12
      2.1.4 Installing the Target Unit OS .................................................................13
      2.1.5 Configuring NFS/DHCP/TFTP Servers in Fedora 16 .........................16

3. Creating the boot loader image ..............................................................................20
   3.1 Open BLDK Development Application .......................................................20
   3.2 Create a New Project ..................................................................................21
   3.3 Build the project..........................................................................................23
   3.4 Conclusion.....................................................................................................24

4. Integrate UEFI Network Stack .............................................................................25
   4.1 Modifications to CrownBayPlatformPkg.dsc file .......................................25
   4.2 Modifications to CrownBayPlatformPkg.fdf files .....................................27
   4.3 Installing the Network Card code base package ........................................28
   4.4 Building the Boot Loader image with PXE boot support ............................29
   4.5 Conclusion.....................................................................................................30

5. Programming the BLDK Binary Image ............................................................31
   5.1 Checking the Image.....................................................................................35

6. UEFI Shell .............................................................................................................36
   6.1 Conclusion.....................................................................................................41

7. EFI PXE Boot .......................................................................................................42
   7.1 Running PXE Boot .......................................................................................44
   7.2 Checking the configuration ..........................................................................45
   7.3 Conclusion.....................................................................................................46

8. Appendix ...............................................................................................................47
   8.1 CrownBayPlatformPkg.dsc ........................................................................47
Implementing PXE Boot using Intel® BLDK for Intel® Atom™ Processor based Boards

8.2  CrownBayPlatformPkg.fdf ........................................................... 48
8.3  DHCPD.CONF ................................................................. 49
8.4  TFTP ................................................................. 50
8.5  ELILO.CONF ................................................................. 50
1. Getting Started

1.1 Notations and Terminology

Table 1 Conventions

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric Constants</td>
<td>Represented as follows:</td>
</tr>
<tr>
<td></td>
<td>• Hexadecimal numbers are represented by a string of hexadecimal digits either beginning with &quot;0x&quot; or ending with the letter &quot;h.&quot;</td>
</tr>
<tr>
<td></td>
<td>• Decimal and binary numbers are represented by their customary notation, that is, 255 is a decimal number and 11111111b is a binary number. Binary numbers are identified by a prefix of &quot;0b&quot; or ending with the letter &quot;b.&quot;</td>
</tr>
<tr>
<td>Units of Measure</td>
<td>The following abbreviations are used to represent units of measure</td>
</tr>
<tr>
<td></td>
<td>• KB - kilobytes (1024 bytes)</td>
</tr>
<tr>
<td></td>
<td>• MB - megabytes (1048576 bytes)</td>
</tr>
<tr>
<td></td>
<td>• MHz - megahertz</td>
</tr>
<tr>
<td></td>
<td>• ms - milliseconds</td>
</tr>
<tr>
<td></td>
<td>• ns - nanoseconds</td>
</tr>
<tr>
<td>Typographic Conventions</td>
<td>The following conventions are used in this manual:</td>
</tr>
<tr>
<td></td>
<td>• Courier font - code examples and command line entries</td>
</tr>
<tr>
<td></td>
<td>• Italic text - filenames, API names, and parameters</td>
</tr>
<tr>
<td></td>
<td>• Bold text - graphical user interface entries and buttons</td>
</tr>
</tbody>
</table>

Table 2 Terminology

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACPI</td>
<td>Advanced Configuration and Power Interface</td>
</tr>
<tr>
<td>ATA</td>
<td>Advanced Technology Attachment</td>
</tr>
<tr>
<td>BSF</td>
<td>Boot Setting File</td>
</tr>
<tr>
<td>BSP</td>
<td>Board Support Package</td>
</tr>
<tr>
<td>CRB</td>
<td>Customer Reference Board</td>
</tr>
</tbody>
</table>
## Term Description

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DHCP</td>
<td>Dynamic Host Configuration Protocol</td>
</tr>
<tr>
<td>DMA</td>
<td>Direct Memory Access</td>
</tr>
<tr>
<td>EFI</td>
<td>Extensible Firmware Interface</td>
</tr>
<tr>
<td>FWH</td>
<td>Firmware Hub</td>
</tr>
<tr>
<td>GDB</td>
<td>GNU* Debugger</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>ICH</td>
<td>Input/output Controller Hub</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>Intel® BLDK</td>
<td>Intel® Boot Loader Development Kit</td>
</tr>
<tr>
<td>IVI</td>
<td>In-Vehicle Infotainment</td>
</tr>
<tr>
<td>MAC</td>
<td>Media access control</td>
</tr>
<tr>
<td>NFS</td>
<td>Network File System</td>
</tr>
<tr>
<td>PCI</td>
<td>Peripheral Component Interface</td>
</tr>
<tr>
<td>PXE</td>
<td>Pre-boot Execution Environment</td>
</tr>
<tr>
<td>ROM</td>
<td>Read Only Memory</td>
</tr>
<tr>
<td>RPM</td>
<td>RPM Package Manager</td>
</tr>
<tr>
<td>RTOS</td>
<td>Real-Time Operating System</td>
</tr>
<tr>
<td>SATA</td>
<td>Serial Advanced Technology Attachment</td>
</tr>
<tr>
<td>SPI</td>
<td>Serial Peripheral Interface</td>
</tr>
<tr>
<td>TFTP</td>
<td>Trivial File Transfer Protocol</td>
</tr>
<tr>
<td>UDK</td>
<td>Unified Extensible Firmware Interface Development Kit</td>
</tr>
<tr>
<td>UEFI</td>
<td>Unified Extensible Firmware Interface</td>
</tr>
<tr>
<td>WinDDK</td>
<td>Windows* Driver Development Kit</td>
</tr>
<tr>
<td>YUM</td>
<td>Yellow-dog Update Manager</td>
</tr>
</tbody>
</table>
1.2 Reference Documents

Break Away with Intel® Atom™ Processors

A Guide to Architecture Migration
By Lori Matassa and Max Domeika
Intel Press (December 16, 2010)
ISBN-10: 1934053376

Harnessing the UEFI Shell

Moving the platform beyond DOS
By Michael Rothman, Tim Lewis, Vincent Zimmer and Robert Hale
Publisher: Intel Press; 1ST edition (January 26, 2010)
ISBN-10: 1934053147

Beyond BIOS 2nd Edition

Developing with the Unified Extensible Firmware Interface
By Vincent Zimmer, Michael Rothman, and Suresh Marisetty
Publisher: Intel Press; 2nd edition (January 19, 2011)
ISBN-10: 1934053295

1.3 Software Resources

This section provides the links to all of the software applications, source code and packages required through this white paper.

<table>
<thead>
<tr>
<th>Package</th>
<th>OS</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Windows 7</td>
<td>Windows</td>
<td>Development platform</td>
</tr>
<tr>
<td>Microsoft Visual Studio</td>
<td>Windows</td>
<td>IDE – BLDK tool uses nmake.exe and cl.exe</td>
</tr>
<tr>
<td>Microsoft Windows DDK</td>
<td>Windows</td>
<td>Required by BLDK IDE</td>
</tr>
<tr>
<td>Intel® BLDK DevApp 2.0.1</td>
<td>Windows</td>
<td>Development application to generate ROM image</td>
</tr>
<tr>
<td>ACPI compiler</td>
<td>Windows</td>
<td>Required by BLDK IDE</td>
</tr>
<tr>
<td>DediProg SF</td>
<td>Windows</td>
<td>SPI Flash programmer</td>
</tr>
<tr>
<td>Fedora 16 DVD</td>
<td>Linux</td>
<td>Boot server platform</td>
</tr>
<tr>
<td>MeeGo 1.2 IVI</td>
<td>Linux</td>
<td>Target system OS</td>
</tr>
<tr>
<td>MeeGo IVI Source</td>
<td>Linux</td>
<td>Target OS Source Code</td>
</tr>
<tr>
<td>ELILO</td>
<td>Linux</td>
<td>Target system OS loader</td>
</tr>
</tbody>
</table>
1.3.1 SDKs, Drivers, Source code

1. Intel® BLDK Core for Crown Bay—Windows*

2. OptionRom Package: Contains code for networking functionality.

   (https://businessportal.intel.com/irj/portal) access

1.4 Hardware resources

1. Intel® Atom™ Processor E660 with Intel® Platform Controller Hub EG20T Development Kit.

   Codename: Crown Bay (Platform) = Tunnel Creek (CPU) + Topcliff (ICH)
   Additional information available at the Intel Embedded website:
   http://www.intel.com/design/intarch/devkits/index.htm?id=embed_portal+hdprod_dev_kits&s1=all&s2=all&s3=Intel%AE%20Platform%20Controller%20Hub%20EG20T

2. DediProg SF100

   In-Circuit-Serial-Programming tool used for SPI flash programming.
   IMPORTANT: Please verify that your development board uses the same interface!
   Additional information available from the DediProg website:

3. Development PC

   PC for development purposes with Ethernet, 120GB HDD/SSD, preferably a serial port for debugging and cables respectively. If no serial ports present refer to 4.
   Note: This guide was tested and performed on a HP 8540 Elite-book

4. USB Serial adapter

   USB adapter used for connecting RS232 serial ports to hosts without serial connectors.

5. 2x 8GB USB Memory Sticks
2. Setup and configuration

2.1 Development System set up instructions

The host computer used in this white paper runs both Microsoft Windows* and Fedora 16* in a dual-boot configuration.

Windows* (XP or higher) is used to run the BLDK IDE and to flash the generated .rom BLDK image onto the target board’s SPI flash memory with the DediProg* programmer.

The Fedora* Linux* environment accommodates 3 services (DHCP-, TFTP-, and NFS-server) and additionally the files for the target operating system (Meego*). This allows the target unit to PXE-boot Meego* with root mounted as a NFS share on the host PC.

MeeGo is a Linux-based open source mobile/embedded OS targeted at embedded devices such as mobiles, netbooks, tablets, IVI systems and many more and is an ideal candidate for the purpose of this guide.

Dual Boot System Note:
The BLDK development tool has been released for Linux as version 2.0.0 (as opposed to Windows version 2.0.1). This guide works with the Windows version, however all BLDK image creation steps are valid for both systems. Please note 3rd party tools may not be compatible with Linux (e.g. DediProg*).

The steps are:


2. Install Fedora* core 16 in dual-boot configuration

3. Configure target machine operating system:
   - Extract MeeGo file structure to hard disk
   - Download MeeGo kernel source, modify PCH_GBE driver, compile
   - Patch and configure ELILO

4. Install and configure:
   - DHCP server
Implementing PXE Boot using Intel® BLDK for Intel® Atom™ Processor based Boards

- TFTP server
- NFS environment
- Terminal emulator (Putty*)

5. Configure BLDK boot image
   - Incorporate network stack and drivers
   - Program onto target board

6. Test system

Having experience with Linux, Fedora 16 and most importantly compiling Linux-kernels in particular will be of advantage in completing these instructions. As always, there are multiple ways you can achieve the same end result. These instructions describe only one such route.

### 2.1.1 Installing Microsoft* Windows* 7

1. Take a laptop with 120GB or larger HDD.
2. Insert Windows 7 DVD to start installation
4. After the installation, go to Start -> Computer (right click) -> Manage -> Disk management. Right click on OSDisk -> Shrink Volume. Shrink OSDisk volume to share the disk space approximately equally between Windows and Linux.
5. Install the required drivers to ensure you have correct graphics, wired and wireless network operation (this depends on the exact laptop system you are using. Default Windows 7 installation may not include drivers for the latest laptop platforms based on 2nd generation Intel® Core™ processors).

### 2.1.2 Preparing Intel® BLDK Environment under Windows*


1. Under Windows* install MS Visual Studio* 2008. Select "Full installation" (This will probably take up to an hour to complete). The Visual Studio documentation is not required.
2. Install Windows DDK 3790.1830 to `c:\winddk\3790.1830`. All options enabled by default.
3. Extract the iASL compiler v20070508 to `C:\ASL`. 
4. Add nmake.exe path to systems path. This should be located in "C:\Program Files\Microsoft Visual Studio 9.0\VC\bin". Right click Computer, select Properties, select Advanced System Settings, click Environment Variables button, select Path and click Edit and add the full path to nmake to this variable.

5. Install drivers and software for DediProg SF 100 programmer. To install the drivers you will need to connect the DediProg SF100 hardware via USB.

6. Create a working directory called BLDK_II under C: drive

7. Extract the Intel® BLDK Core to C:\BLDK_II (See link in Section 1.3 Software Resources)

8. Install the Intel® BLDK Development Application 2.0.1 (IDE)

### 2.1.3 Installing Fedora* 16

1. Download the Fedora* 16 32bit DVD image (Fedora-16-i386-DVD.iso) and burn a DVD, then run the installation program from the DVD. Do not use the live installation as it historically does not contain all the required services by default.

2. Skip the media test.

3. Select English installation

4. Select appropriate keyboard layout

5. Select Basic Storage Devices

6. When asked for type of installation select *Use Free Space or Replace Existing Linux System* if a previous Linux install exists.

7. On the "Customize Now" screen that lets you select the role of the system, please check the Customize Now radio button then click **Next**. Select Servers, Network Servers & Server Configuration Tools, Base System and System Tools. Click Next – system will now install all selected packages. This could take between 30 minutes to 1 hour depending on speed of DVD drive (and other factors...)

8. When asked, set Root password: **intelbldk**

9. Once the installation has completed and the system is rebooted you should be presented the GRUB menu. Two items are listed:

Fedora
Other

10. After Reboot – create this user at the appropriate screen.

    Username: **intel** (add to administrators group)
    Password: **bldk**
2.1.4 Installing the Target Unit OS

This section covers extracting and deploying Meego’s file system to a directory within Fedora 16. Additionally a MeeGo IVI kernel is modified and compiled to suit our needs.

As extracted from the release notes (http://edc.intel.com/Link.aspx?id=5403) Intel BLDK does not support all OSes. For this reason MeeGo 1.2.0 kernel version 2.6.37.6 (kernel-adaptation-intel-automotive-2.6.37.6-20.1.src.rpm) is used as the target unit OS. Alternatively, you may choose to use Fedora 14 Timesys. However the instructions have not been validated for FC14 Timesys.

As the firmware implemented by the BLDK image is EFI compliant, an EFI compliant second stage boot-loader must be used. For this purpose ELILO is used.

2.1.4.1 Preparing the Target Unit File-System

1. Download the MeeGo live image and install to a USB stick (for details on how to make a bootable USB go to https://meego.com/devices/netbook/installing-meego-your-netbook)
2. Reboot development PC into the MeeGo-USB-stick, select installation only and install to 2nd USB stick (>4GB)
3. Restart and boot into Fedora 16
4. Copy the file structure from 2nd USB-Stick to /nfsroot and another copy to /devel
5. Modify fstab file under /nfsroot/etc/ as MeeGo was setup to boot from a USB stick
   a) Enter following into a terminal
      [intel@localhost ~]$ su
      Password: intelbldk
      [root@localhost]# vi /nfsroot/etc/fstab
   b) Comment out all lines with a pattern similar to “/dev/sd<num>”
   c) Add line (blue indicates change), save and close
      /dev/nfs / nfs defaults 0 0
      tmpfs /dev/shm tmpfs defaults 0 0
      devpts /dev/pts devpts gid=5,mode=620 0 0
      sysfs /sys sysfs defaults 0 0
      proc /proc proc defaults 0 0
2.1.4.2 Preparing the Target Unit Kernel

This stage entails downloading the MeeGo-kernel source code, modifying the code to integrate specific drivers and customizing these drivers as part of a workaround.

1. Download MeeGo source code from repo.meego.com (refer to Software Resources for link) and copy to /devel/root

2. Install Mic-chroot
   a) Add MIC2 repo as user root:
      ```
      # cat <<REPO > /etc/yum.repos.d/meego-tools.repo
      [meego-tools]
      name=MeeGo Tools for Fedora
      baseurl=http://repo.meego.com/MeeGo/tools/repos/fedora/15
      enabled=1
      gpgcheck=0
      gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-meego
      REPO
      ```
      b) Install mic2 as root user
      ```
      # sudo yum install mic2 --nogpgcheck
      ```

3. Change root path to /devel
   ```
   mic-chroot /devel
   ```

4. Install source code, which will extract the kernel source code to “/root/rpmbuild”
   ```
   rpm –ivh <kernel-image.src.rpm>
   ```

5. Install additional packages
   ```
   # zypper install binutils rpm-build make linux-firmware elfutils-libelf-devel binutils-devel newt-devel
   ```

6. Prepare build environment. Built under “/root/rpmbuild/SPECS”
   ```
   rpmbuild -bp kernel-adaptation-intel-automotive.spec
   ```

Note: When the process gets to the stage of displaying warnings and does not seem to continue, hit enter a few times.
7. Workaround for MAC-Address bug:

We are investigating why the PCH_GBE driver retrieves an invalid MAC address of 00-00-00-00-00-00 and consequentially terminates itself. This behavior is only present with machines booted from the network. To provide a workaround for this glitch, perform the following steps:

Warning: This machine will boot with a MAC address of 0. Only one machine can be present on the network with this address at any time.

a) Under “~/rpmbuild/BUILD/kernel-X.X.XX.X/linux-X.X.XX” browse to “drivers/net/pch_gbe”
b) Open pch_gbe_main.c with vi
c) Search for the line “Invalid MAC Address”
d) Comment the if statement out

8. Run make menuconfig under “rpmbuild/.../linuxX.X.XXX” to customize kernel image

a) In Device Drivers->Network Device Support->Ethernet (1000Mbit) – ensure the PCH Gigabit Ethernet is selected. This is the driver for the EG20T internal MAC.
b) In Networking Support->Networking Options – ensure IP: kernel level auto-configuration is selected. This enables root file system over NFS to be visible in the File Systems > Network File Systems menu. (The options for DHCP, BOOTP, RARP are your choice).
c) In File Systems->Network File Systems – select NFS Client Support, select Root file system on NFS. (You may select support for NFS Client V3 & V4 if you wish, but most NFS servers should be happy with V2 client support).

9. Create kernel with “make binrpm-pkg”

10. Retrieve generated kernel image from “./arch/x86/boot/bzImage” and copy to the host filesystem “/var/lib/tftpboot/efi” (under original root!)

11. This image can be checked by installing it onto an existing file system (HDD or similar) on the target machine and booting into it.

2.1.4.3 Second Stage Boot Loader

During the boot-process BLDK passes control to an EFI compliant boot loader which downloads the kernel image from the server, decompresses it and initializes it. In this guide ELILO is used as the second stage boot loader.

Normally Elilo only allocates a memory range of 0x100000 – 0x500000 (~4MB) for the kernel image. If your kernel exceeds this size, the image will not boot properly. As a result it is important to use the patched Elilo version as described in the white paper Setting Up OS Environments Using EDKII Based Intel® BLDK.
Implementing PXE Boot using Intel® BLDK for Intel® Atom™ Processor based Boards


2) Copy elilo_ia32.efi to “/var/lib/tftpboot/efi/”

3) Create a file named elilo.conf in the same folder as elilo_ia32.efi

4) Insert the following into the file:

   legacy-free
   prompt
   timeout=1
   default=linux
   verbose=5

   image=/efi/bzImage
   label=linux
   read-only
   append="root=/dev/nfs single console=ttys0 console=ttyS0,115200n8
   nfsroot=192.168.0.2:/nfsroot
   ip=192.168.0.9::192.168.0.4:255.255.255.0:crownbay:eth0:off"

The folder “/var/lib/tftpboot/efi/” should contain three files: elilo_ia32.efi, elilo.conf, and bzImage.

2.1.5 Configuring NFS/DHCP/TFTP Servers in Fedora 16

1. Login as user intel

2. Open a terminal and login as root using the su command.

3. Disable the firewall using these commands:

   systemctl stop iptables.service
   systemctl disable iptables.service

4. Set up DHCP server:

   a) If for some reason DHCP server is not found, install with:

      yum -y install dhcp

   b) Change DHCP server settings by adding the following to the file
Implementing PXE Boot using Intel® BLDK for Intel® Atom™ Processor based Boards

/etc/dhcp/dhcpd.conf

allow bootp;
filename "/efi/elilo_ia32.efi";
authoritative;
ddns-update-style none;
next-server 192.168.0.2;

subnet 192.168.0.0 netmask 255.255.255.0 {
    range 192.168.0.4 192.168.0.10;
    default-lease-time 3600;
    max-lease-time 4800;
}

DHCP Service Notes:
In order for Fedora to function correctly as a DHCP server, the wired Ethernet interface needs to be set to a static IP address. dhcpd.service will not start successfully unless a network interface is active. Hence, the service will return FAILED.

c) To auto-start the DHCP service at boot time run the command:

    systemctl enable dhcpd.service

5. Install TFTP server:

a) If you did not include the package during installation, install with:

    yum -y install tftp-server

b) To configure the tftp service, edit the file /etc/xinetd.d/tftp:

    Change line disable = yes to disable = no

    Change line server_args = -s /var/lib/tftpboot to server_args = -s /var/lib/tftpboot -v -v -v -c for debug verbosity.

Tftpboot directory (/var/lib/tftpboot/efi/) should have been populated with bzImage, elilo.conf and elilo_ia32.efi in section 2.1.4.1 Preparing the Target Unit File-System

c) Change permissions for the content of /var/lib/tftpboot/efi folder by running the command:

    chmod 777 -R /var/lib/tftpboot/

    chmod 777 -R /var/lib/tftpboot/efi/

6. Add the following line to the file /etc/exports
/nfsroot *(rw,no_root_squash,no_subtree_check)

7. To enable autostart run the command:

```
systemctl enable nfs-server.service
```

**Fedora 16 Note:**
- nfs.service has been renamed to **nfs-server.service**.
- To auto-start services use `systemctl enable <service-name>`

8. Change permissions for the folder /nfsroot made in section 2.1.4.1 Preparing the Target Unit File-System

```
chmod 777 -R /nfsroot
```

**Serial Ports:**
- If you do not have a serial port you can use a USB-to-serial converter. In this case replace /dev/ttyS0 with /dev/ttyUSB0. `dmesg | grep tty` will return all serial devices connected to the PC.
- Note that you must be root to read/write to serial ports.

9. Install a suitable terminal emulator. Communication with BLDK2 is via serial link and requires a terminal emulator.

```
yum -y install putty
```

a) After installing putty, run it and set the serial ports settings like this:

- **Port** = /dev/ttyS0
- **Baud rate** = 115200
- **8 data bits**
- **1 stop bit**
- **No parity**
- **No flow control**

b) Save these settings so that you can recall them again next time you invoke putty.

10. Open up the wired connection configuration from system settings. Edit the network address to be manual and add the following address under the IPv4 tab.

   **Address:** **192.168.0.2**

   **Netmask:** **255.255.255.0**
Implementing PXE Boot using Intel® BLDK for Intel® Atom™ Processor based Boards

Gateway: 192.168.0.2

Check the **Connect automatically, Require IPv4** and the **Available to all users**.

Note: In order to change the configuration, an Ethernet cable must be plugged into a router or similar. Please note that these settings will impair internet access and may need to be switched back if internet access is required.

11. Check that all required services are running:

   a) Check the wired interface is working correctly: `ifconfig`

   b) Check the wired interface has the address 192.168.0.2 assigned

   c) Check dhcp is running: `ps -eal | grep dhcp`

   d) Check this lists one entry for dhcpd.

   e) Check xinetd is running (this will start tftp when required): `ps -eal | grep xinetd`

   f) Check there is one entry for xinetd listed.

   g) Check nfs is running: `ps -eal | grep nfs`

   h) Check there are several nfs entries listed.

---

**Eth0 Note:**
From Fedora 15 and up eth0 has been replaced with BIOS device names to easier correlate OS network connections with physical devices. E.g. eth0 may be listed as p3p1 (if PCI device) [http://fedoraproject.org/wiki/Features/ConsistentNetworkDeviceNaming](http://fedoraproject.org/wiki/Features/ConsistentNetworkDeviceNaming)
3. Creating the boot loader image

This section guides you through building a standard boot loader image for the Intel® Atom™ Processor E6xx Series with Intel® Platform Controller Hub EG20T evaluation board.


3.1 Open BLDK Development Application

1. Boot the laptop into Windows (select other from the GRUB menu)

2. Login to the intel account (password: bldk)

3. Run BLDK from Start -> All Programs -> Intel Boot Loader Development Kit -> Intel Boot Loader Development Kit or double click on the icon on the desktop.

The BLDK environment opens,
3.2 Create a New Project

Click **Project -> New Project** on the menu bar. Fill in your project name and all the project components as follows.

- **Project name** - Name of your project.  
  **Type in, crownbay**

- **Project file directory** - Location of the project file being created.  
  **Browse to C:BLDK_II (if not already set)**

- **Workspace directory** - Location of the source and binary files used to generate the firmware image.  
  **Browse to C:BLDK_II**

- **Image configuration file** - Location of the Boot Setting File (BSF).  
  **Browse to C:BLDK_II\CrownBayPlatformPkg\FV\tc.bsf**

Make sure the paths are the same as in the image shown here,
Then click Start Configuration. The Create a new project window is displayed, as shown here,

![Create a new project window](image)

Tips

Two different build modes are available,

- **Release**: standard build mode with full compiler optimizations enabled. (Default)
- **Debug**: special debug version of the firmware that enables output of debug messages via serial cable and also builds symbol information required for source level debugging.

Make sure Source Debug is set to Disabled.

Click Create Project.

Upon completion, the project is created and a list of files will be displayed in the Navigation pane, on the Standard tab.
3.3 Build the project

Let’s now build the project. Select Build > Build

Build messages are displayed in the bottom window of the Development application.

Upon successful completion the following window is displayed.

The firmware image CROWNBAY.fd generated by the Build command is located in directory,

   C:\BLDK_II\Build\CrownBayPlatform\RELEASE_VS2008\FV\

In the BLDK development application select Build > Create Final Firmware Image.

Click browse to select a binary file. Make sure the All Files option is selected.

   C:\BLDK_II\Build\CrownBayPlatform\RELEASE_VS2008\FV\CROWNBAY.fd.
Select file and Click Open. The Firmware file name is filled-in automatically to crownbay.rom.

Click OK. A message window will appear upon successful completion.

3.4 Conclusion

Now that we have successfully built a boot loader image, we could use a FLASH programmer to upload the boot loader image on to an Intel® Atom™ Processor E6xx Series with Intel® Platform Controller Hub EG20T evaluation board. However the standard image does not implement the networking resources we need to support PXE boot. In the next section we will learn how to implement PXE boot in the boot loader image.
4. **Integrate UEFI Network Stack**

The UEFI Network code base package is already integrated in to the *Intel® BLDK Core for Crown Bay (UEFI Standard Based)-Gold Release*. However in order to support PXE boot, we need to integrate the network drivers in our project. This is achieved by making modifications to the project description file *CrownBayPlatformPkg.dsc* and to the flash description file *CrownBayPlatformPkg.fdf*.

We also need to integrate the driver for the onboard network card we are using on the *Intel® Atom™ Processor E6xx Series with Intel® Platform Controller Hub EG20T evaluation board*.

**UEFI Network Stack:**

You can learn more about the UEFI network stack by following these two links. The first link opens the Network-IO project homepage, and the second link provides access to a white paper.


### 4.1 Modifications to *CrownBayPlatformPkg.dsc* file

The files are accessible from the BLDK development environment. Select tab *Workspace* and expand directory *CrownBayPlatformPkg*. Double click on *CrownBayPlatformPkg.dsc* to open it.
Implementing PXE Boot using Intel® BLDK for Intel® Atom™ Processor based Boards

Modifications consist of adding the following lines at about line#1010 in the CrownBayPlatformPkg.dsc file.

```
# # PCI card
# CrownBayPlatformPkg/PciPlatformDxe/PciPlatformDxe.inf
MdeModulePkg/Bus/Pci/PciBusDxe/PciBusDxe.inf
TopcliffPkg/IohInitDxe/IohInitDxe.inf
TopcliffPkg/IohSerialDxe/SerialDxe.inf
OptionRomPkg/UndiRuntimeDxe/UndiRuntimeDxe.inf

# # Network

MdeModulePkg/Universal/Network/UefiPxeBcDxe/UefiPxeBcDxe.inf
MdeModulePkg/Universal/Network/Mtftp4Dxe/Mtftp4Dxe.inf
MdeModulePkg/Universal/Network/Dhcp4Dxe/Dhcp4Dxe.inf
MdeModulePkg/Universal/Network/Udp4Dxe/Udp4Dxe.inf
MdeModulePkg/Universal/Network/Ip4Dxe/Ip4Dxe.inf
MdeModulePkg/Universal/Network/ArpDxe/ArpDxe.inf
MdeModulePkg/Universal/Network/Ip4ConfigDxe/Ip4ConfigDxe.inf
MdeModulePkg/Universal/Network/MnpDxe/MnpDxe.inf
MdeModulePkg/Universal/Network/SnpDxe/SnpDxe.inf
MdeModulePkg/Universal/Network/DpcDxe/DpcDxe.inf
```
4.2 Modifications to CrownBayPlatformPkg.fdf files

The following modifications were made to the file CrownBayPlatformPkg.fdf to enable native UEFI PXE boot with the BLDK EDKII tree on Intel® Atom™ Processor E6xx Series with Intel® Platform Controller Hub EG20T platforms. The PXE boot uses DHCP for IP allocation and TFTP protocol for file transferring which was configured in section 2.1.5 Configuring NFS/DHCP/TFTP Servers in Fedora 16.

FVRECOVERY2 was commented out to save space for the network driver.

Copy the Undi.efi driver file for Topcliff MAC into the CrownBayPlatformPkg folder under C:\BLDK_II.

Line number may differ slightly.

---- line 100 ---------------
#0x000A8000|0x00008000
#FV = FVRECOVERY2

---- line 178 ---------------
0x00000000|0x001800000x000B0000
gCrownBayPlatformTokenSpaceGuid.PcdFlashFvMainBase|gCrownBayPlatformTokenSpaceGuid.PcdFlashFvMainSize
FV = FVMAIN_COMPACT
#0x00180000|0x00010000
#FV = FVRECOVERY2

---- line 245 ---------------
#0x001B0000|0x000B0000
0x00180000|0x00010000
FILE = TunnelCreekPkg/Binary/TCMicrocode/C0_060510_GvdCGDis.BIN
FILE = TunnelCreekPkg/Binary/TCMicrocode/C0_22211.BIN
#FV_UMG_OPTIONAL

---- line 333 ---------------
[FV.FVRECOVERY2]
#BlockSize = 0x8000
#FvAlignment = 16 #FV alignment and FV attributes setting.
#ERASE_POLARITY = 1
#MEMORY_MAPPED = TRUE
#STICKY_WRITE = TRUE
#LOCK_CAP = TRUE
#LOCK_STATUS = TRUE
#WRITE_DISABLED_CAP = TRUE
#WRITE_ENABLED_CAP  = TRUE
#WRITE_STATUS       = TRUE
#WRITE_LOCK_CAP     = TRUE
#WRITE_LOCK_STATUS  = TRUE
#READ_DISABLED_CAP  = TRUE
#READ_ENABLED_CAP   = TRUE
#READ_STATUS        = TRUE
#READ_LOCK_CAP      = TRUE
#READ_LOCK_STATUS   = TRUE

---- line 385 -------------------
#INF  TianoModulePkg/Universal/Disk/FileSystem/FatPei/FatPei.inf
#INF  TianoModulePkg/Universal/Disk/FileSystem/CDExpressPei/CdExpressPei.inf

---- line 484 -------------------
# # PCI card
# INF OptionRomPkg/UndiRuntimeDxe/UndiRuntimeDxe.inf
#
# # Network
#
INF MdeModulePkg/Universal/Network/UefiPxeBcDxe/UefiPxeBcDxe.inf
INF MdeModulePkg/Universal/Network/Mtp4Dxe/Mtp4Dxe.inf
INF MdeModulePkg/Universal/Network/Dhcp4Dxe/Dhcp4Dxe.inf
INF MdeModulePkg/Universal/Network/Udp4Dxe/Udp4Dxe.inf
INF MdeModulePkg/Universal/Network/Ip4Dxe/Ip4Dxe.inf
INF MdeModulePkg/Universal/Network/ArpDxe/ArpDxe.inf
INF MdeModulePkg/Universal/Network/Ip4ConfigDxe/Ip4ConfigDxe.inf
INF MdeModulePkg/Universal/Network/SnpDxe/SnpDxe.inf
INF MdeModulePkg/Universal/Network/MnpDxe/MnpDxe.inf
INF MdeModulePkg/Universal/Network/DpcDxe/DpcDxe.inf

# UNDI driver for MAC of Topcliff
FILE DRIVER = C9FD3B90-0DBD-4e1a-8A2A-75CB0A3C825E { 
SECTION PE32 = CrownBayPlatformPkg/Undi.efi }

4.3 Installing the Network Card code base package

The code base package required to support the network card is called OptionRomPkg
and is available from the Source Forge website (link provided under chapter “Getting
Started > Software Resources”.
Open the zip file UDK2010.UP4.zip > UDK2010.UP4 and extract OptionRomPkg into C:\BLDK_II to obtain the following directory tree.

4.4 Building the Boot Loader image with PXE boot support

We are now ready to create a boot loader image with PXE boot support for the Intel® Atom™ Processor E6xx Series with Intel® Platform Controller Hub EG20T development kit.

In the BLDK IDE and open the project Project > Recently Open Projects > ...crownbay.ews and click OK in the pop up window.

Rebuild the project to integrate the modifications, Build > Clean All and then Build > Build.

Once the compiler has finished building the image select Build > Create Final Firmware Image. A window pops up. Browse the .fv file. Make sure the All Files option is selected.

For the Binary file, browse to C:\BLDK_II\Build\CrownBayPlatform\RELEASE_VS2008\FV\CROWNAY.fd.

Select file and Click Open. The Firmware file name is filled-in automatically to crownbay.rom.

Click OK. A message window will appear upon successful completion.
4.5 Conclusion

In the next section we will program the SPI-Flash device with the boot loader image we have created.
5. **Programming the BLDK Binary Image**

In this section, we will be using the Dediprog® SF100 device to program the BLDK binary image created in the previous sections onto the Intel® Atom™ Processor E6xx Series with Intel® Platform Controller Hub EG20T development kit SPI FLASH.

![Figure 1. SF100 with header programmer used for Intel® Atom™ Processor E6xx Series with Intel® Platform Controller Hub EG20T Development Kits](image1)

**Figure 1. SF100 with header programmer used for Intel® Atom™ Processor E6xx Series with Intel® Platform Controller Hub EG20T Development Kits**

![Figure 2. SF200 with DIP to SO8N(150mil) adaptor as used by some OEMs](image2)

**Figure 2. SF200 with DIP to SO8N(150mil) adaptor as used by some OEMs**

**SPI Flash Note:**
The Intel® Atom™ Processor E6xx Series with Intel® Platform Controller Hub EG20T CRB uses a PC board pin header connector (as shown in the figure below). Please compare with your motherboard’s datasheet if you are not using an Intel® CRB.
The SF100 USB cable is connected to the host machine running Windows®. The 8-pin connector is connected to connector J1A2 on the Intel® Atom™ Processor E6xx Series with Intel® Platform Controller Hub EG20T board. The red wire on the cable must be aligned with the white arrow on the board. The target board power supply must be switched off.

Double-click the Dediprog* shortcut located on the Windows* desktop. The SPI FLASH memory device should be recognised and the following pop-up window should appear.
Implementing PXE Boot using Intel® BLDK for Intel® Atom™ Processor based Boards

Click OK, and the application window opens.

Important Note!
Before rewriting the SPI flash with the BLDK image, please make sure to make a backup of the original BIOS. To retrieve the current image from the SPI flash click Edit in the toolbar, then “Read...” (Source > Chip area) in the window which pops up. Finally store the image to your hard disk with “Chip Buffer to File”.

Click the Config icon on the right hand side of the Dediprog* main menu. Click Prog on the left hand side bar of the pop-up window. Make sure the second radio button is selected and that the Starting Address is set to 0x0100000 (5 zeros). Click OK at the bottom of the window.
Implementing PXE Boot using Intel® BLDK for Intel® Atom™ Processor based Boards

Back in the main window click File in the main menu bar and use the Find button to navigate to
C:\BLDK_II\Build\CrownBayPlatform\RELEASE_VS2008\FV\CROWNBAY.rom

Why 0x0100000?
The SPI-flash located on the board is the SST25VF016BA (http://www.sst.com/dotAsset/40371.pdf) with 16Mbits (2MB) of memory (address range 0x00000000 to 0x01FFFFFF).
As the BLDK image only requires 1MB we instruct it to upload the image half way (0x0100000 = 1048576 bytes ≈ 1MB) so that only the second half of the chip is populated. Lower half is commonly used for debug versions. Release versions on the other hand use the upper half. The rest of the chip is padded with value FF.

Click OK, then click Prog in the main menu bar. Upon completion, Programming OK appears in the message window as shown here.
5.1 Checking the Image

Disconnect the Dediprog* device from the board. **Switch ON** the Intel® Atom™ Processor E6xx Series with Intel® Platform Controller Hub EG20T development kit power supply and then **press the power up button** located near the PCI slot.

The 7-Segment LED display should read b6. If it reads 00 or any other value then the image is probably corrupt and malfunctioning. Please revise the previous steps to ensure you have performed all instructions correctly.

**Switch OFF** the development kit Power Supply.

The Intel® Boot Loader image was installed successfully onto the SPI FLASH memory.
6. **UEFI Shell**

In this section the Intel® Atom™ Processor E6xx Series with Intel® Platform Controller Hub EG20T development kit is powered up and a connection established with the Intel® Boot Loader Development Kit over a serial link.

**Close all applications** and **restart** the development PC. Let it boot to Fedora 16.

**Log in** to **intel** user account with password **bldk**

Fedora 16 desktop will appear as follows:

![Fedora 16 Desktop](image)

**Click on Activities** in the top left hand corner to make the side bar appear and open a **Terminal**.
In the Terminal session window **enter the following commands** to open the Serial Link interface.

```
[intel@localhost ~]$ su
Password: intelbldk
[root@localhost intel]# putty
```

The Putty set up window appears. **Select crownbay** under saved sessions and **click Load**. The serial link configuration is loaded (/dev/ttyS0 or /dev/ttyUSB0 with 115200 baud). **Click Open** to start the Putty session.
Connect the null modem serial link cable from the host computer to the target board. The Serial Link socket next to the purple PS2 socket must be used.

Make sure an Ethernet Cable is plugged between the target board and the host computer.
Connect power cable to the target board and then press the power up button. (Located bottom left of the carrier board by the PCI slot) (Keep the button pressed for 5 seconds to switch the board off).

After about 10 seconds, the Intel® Boot Loader Development Kit banner and Shell> prompt should appear in the putty window as shown below. You have successfully created and installed a boot loader on the target board.

You are now in the UEFI Shell. Type help in order to see the list of available commands.
Type `exit` at the Shell prompt in order to exit the UEFI Shell. This will force the board to enumerate all bootable devices. The Intel® Platform Controller Hub EG20T MAC on the development lot board will appear as “EFI Network”. It will take about 10 seconds for the boot loader to enumerate the boot options.

Type 1 to go back to the Shell> prompt.
6.1 Conclusion

A boot loader image with PXE boot support has been created and installed on the target board. This verifies that the BLDK image is working. The target is now ready to receive an OS from the server.
7. **EFI PXE Boot**

Booting the target board over an Ethernet link is also known as PXE boot. The following figure below shows an illustration of the PXE boot process as implemented. The host computer running Fedora has been set up as a DHCP server with a static network address 192.168.0.2. The Intel® Boot Loader Development Kit running on the target is EFI compliant and uses EFI compliant commands to connect to the host.
Implementing PXE Boot using Intel® BLDK for Intel® Atom™ Processor based Boards

Simplified PXE boot process

Host

- Client IP address
- tftp server address
- name of file to request

Target

- bootp request
- 198.168.0.4 (from range 4 to 10)
- 198.168.0.2
- name = /efi/elilo_ia32.efi

- tftp request for file /efi/elilo_ia32.efi

- tftp server sends file
- file = elilo_ia32.efi

- execute elilo_ia32.efi
  Path to OS image to boot is provided in file elilo.conf

- tftp request for OS kernel image to boot

- tftp server sends file
- file = bzimage

- Target install OS image RAM memory
- OS kernel runs

- OS kernel makes an NFS request to mount the file system

- NFS server provides path to file system to mount

- file system path = HOST/nfsroot

The OS kernel running in memory on the target is using a file system located on the host over the Ethernet link.

1 as defined in configuration file /etc/dhcp/dhcpd.conf
2 elilo_ia32.efi is an initial loader working in the efi environment
3 as defined in /etc/exports
7.1 Running PXE Boot

If you have followed the steps in the document so far, the desktop on the host computer should be similar to the figure below. Pay attention to the network icon which should show an active connection. An active network icon looks like this:

At the boot selection menu enter 3 to boot from the EFI Network.

The Linux* kernel is downloaded and boots from the target machine. The file system "/nfsroot" residing on the host computer is mounted.

Finally the Linux* prompt [SH-3.2/]# appears.

Type ls command to see the file system directories.
7.2 Checking the configuration

In a Terminal session window on the host pc, type the following commands to create a new directory called Debug

```
[intel@localhost ~]$ cd /nfsroot
[intel@localhost ~]$ mkdir debug
[intel@localhost ~]$ ls
```

Check *debug* has been created.

```
[root@localhost intel]# cd /nfsroot
[root@localhost nfsroot]# ls
bin  etc  home  lib  media  opt  proc  root  sbin  srv  tmp  var
```

In the Putty session window, type the following command and check that the debug directory created on the host computer is seen by the target.

```
[SH-3.2/]# ls
```
7.3 Conclusion

A working debug environment for the Intel® Atom™ Processor E6xx Series with Intel® Platform Controller Hub EG20T development kit based on Intel® BLDK Core for CrownBay (UEFI Standard Based) has been created. This potentially allows multiple diskless embedded devices to boot from a centralized server, thus enabling administrators to update one location to control content on all connected devices.

8. **Appendix**

8.1 **CrownBayPlatformPkg.dsc**

C:\BLDK_II\CrownBayPlatformPkg\CrownBayPlatformPkg.fdf

#  
# PCI card  
#  
CrownBayPlatformPkg/PciPlatformDxe/PciPlatformDxe.inf  
MdeModulePkg/Bus/Pci/PciBusDxe/PciBusDxe.inf  
TopcliffPkg/IohInitDxe/IohInitDxe.inf  
TopcliffPkg/IohSerialDxe/SerialDxe.inf  
OptionRomPkg/UndiRuntimeDxe/UndiRuntimeDxe.inf  

#  
# Network  
#  
MdeModulePkg/Universal/Network/UefiPxeBcDxe/UefiPxeBcDxe.inf  
MdeModulePkg/Universal/Network/Mtftp4Dxe/Mtftp4Dxe.inf  
MdeModulePkg/Universal/Network/Dhcp4Dxe/Dhcp4Dxe.inf  
MdeModulePkg/Universal/Network/Udp4Dxe/Udp4Dxe.inf  
MdeModulePkg/Universal/Network/Ip4Dxe/Ip4Dxe.inf  
MdeModulePkg/Universal/Network/ArpDxe/ArpDxe.inf  

MdeModulePkg/Universal/Network/Ip4ConfigDxe/Ip4ConfigDxe.inf  
MdeModulePkg/Universal/Network/MnpDxe/MnpDxe.inf  
MdeModulePkg/Universal/Network/SnpDxe/SnpDxe.inf  
MdeModulePkg/Universal/Network/DpcDxe/DpcDxe.inf
8.2 CrownBayPlatformPkg.fdf

Implementing PXE Boot using Intel® BLDK for Intel® Atom™ Processor based Boards

C:\BLDK_II\CrownBayPlatformPkg\CrownBayPlatformPkg.fdf

Line number may differ.

--- line 100 ---------------------
#0x000A8000|0x00080000
#FV = FVRECOVERY2

--- line 178 ---------------------
0x00000000|0x00180000|0x000B0000
#gCrownBayPlatformTokenSpaceGuid.PcdFlashFvMainBase|gCrownBayPlatformTokenSpaceGuid.PcdFlashFvMainSize
FV = FVMAIN_COMPACT
#0x00180000|0x00010000
#FV = FVRECOVERY2

--- line 245 ---------------------
0x001B0000|0x000B0000|0x00010000
#gCrownBayPlatformTokenSpaceGuid.PcdFlashTCMicrocodeBase|gCrownBayPlatformTokenSpaceGuid.PcdFlashTCMicrocodeSize
FILE = TunnelCreekPkg/Binary/TCMicrocode/C0_060510_GvdCGDis.BIN
FILE = TunnelCreekPkg/Binary/TCMicrocode/C0_22211.BIN
#FV_UMG_OPTIONAL

--- line 333 ---------------------
#FV.FVRECOVERY2]
#BlockSize = 0x8000
#FvAlignment = 16  #FV alignment and FV attributes setting.
#ERASE_POLARITY = 1
#MEMORY_MAPPED = TRUE
#STICKY_WRITE = TRUE
#LOCK_CAP = TRUE
#LOCK_STATUS = TRUE
#WRITE_DISABLED_CAP = TRUE
#WRITE_ENABLED_CAP = TRUE
#WRITE_STATUS = TRUE
#WRITE_LOCK_CAP = TRUE
#WRITE_LOCK_STATUS = TRUE
#READ_DISABLED_CAP = TRUE
#READ_ENABLED_CAP = TRUE
#READ_STATUS = TRUE
#READ_LOCK_CAP = TRUE
#READ_LOCK_STATUS = TRUE

--- line 385 ---------------------
#INF TianoModulePkg/Universal/Disk/FileSystem/FatPei/FatPei.inf
# INF
TianoModulePkg/Universal/Disk/FileSystem/CDExpressPei/CDExpressPei.inf

--- line 484 ---------------
#
# PCI card
#
INF OptionRomPkg/UndiRuntimeDxe/UndiRuntimeDxe.inf

#
#
INF
MdeModulePkg/Universal/Network/UefiPxeBcDxe/UefiPxeBcDxe.inf
INF MdeModulePkg/Universal/Network/Mtftp4Dxe/Mtftp4Dxe.inf
INF MdeModulePkg/Universal/Network/Dhcp4Dxe/Dhcp4Dxe.inf
INF MdeModulePkg/Universal/Network/Udp4Dxe/Udp4Dxe.inf
INF MdeModulePkg/Universal/Network/Ip4Dxe/Ip4Dxe.inf
INF MdeModulePkg/Universal/Network/ArpDxe/ArpDxe.inf
INF MdeModulePkg/Universal/Network/Ip4ConfigDxe/Ip4ConfigDxe.inf
INF MdeModulePkg/Universal/Network/SnpDxe/SnpDxe.inf
INF MdeModulePkg/Universal/Network/MnpDxe/MnpDxe.inf
INF MdeModulePkg/Universal/Network/DpcDxe/DpcDxe.inf

# UNDI driver for MAC of Topcliff
FILE DRIVER = C9FD3B90-0DBD-4e1a-8A2A-75CB0A3C825E {
SECTION PE32 = CrownBayPlatformPkg/Undi.efi
}

8.3 DHCPD.CONF
/etc/dhcp/dhcpd.conf

allow bootp;
filename "/efi/elilo_ia32.efi";
authoritative;
ddns-update-style none;
next-server 192.168.0.2;

subnet 192.168.0.0 netmask 255.255.255.0 {
range 192.168.0.4 192.168.0.10;
default-lease-time 3600;
max-lease-time 4800;
}


8.4 TFTP

/etc/xinet.d/tftp

# default: off
# description: The tftp server serves files using the trivial 
# file transfer \ 
# protocol. The tftp protocol is often used to boot diskless \ 
# workstations, download configuration files to network-aware 
# printers, \ 
# and to start the installation process for some operating 
# systems.

service tftp
{
    socket_type  = dgram
    protocol    = udp
    wait        = yes
    user        = root
    server      = /usr/sbin/in.tftpd
    server_args = -s /var/lib/tftpboot -v -v -v -c
disable      = no
    per_source  = 11
cps          = 100 2
    flags       = IPv4
}

8.5 ELILO.CONF

/var/lib/tftpboot/efi/elilo.conf

legacy-free
prompt
timeout=1
default=linux
verbose=5

image=/efi/bzImage
    label=linux
    read-only
    append="root=/dev/nfs single console=tt0 console=ttyS0,115200n8
nfsroot=192.168.0.2:/nfsroot
ip=192.168.0.9::192.168.0.4:255.255.255.0:crownbay:eth0:off"

Authors

Dmitry Tarakanov, John A.D. Mallinder, Regis Cheval, Steve Cutler 
are Technical Marketing Engineers at Intel Corporation.
Implementing PXE Boot using Intel® BLDK for Intel® Atom™ Processor based Boards

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENCE IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to:
http://www.intel.com/design/literature.htm

This paper is for informational purposes only. THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. Intel disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein.

Intel, the Intel logo, Atom, Core, and Xeon are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2012 Intel Corporation. All rights reserved.