# Embedded Development with the Intel® Boot Loader Development Kit

**By Sean D. Liming & John R. Malin**
Annabooks

Published in the United States by

**SJJ Embedded Micro Solutions, LLC.**
**Dba**
**Annabooks**
6432 Glendale Dr.
Yorba Linda, CA 92886 USA

www.sjjmicro.com
www.annabooks.com

# Table of Contents

# 1 The BIOS Evolution

Developing a custom embedded PC board was commonplace in the 80s and 90s. A developer chose a processor and a super I/O chip, and then added some memory and maybe a programmable chip for custom I/O. Board layout, firmware, and OS bring-up were the major development milestones. For firmware, a Basic Input/Output System or BIOS was then configured for the chosen hardware. One could buy and license a BIOS solution from companies like Phoenix Technologies, AMI, Award, and SystemSoft, or custom build a BIOS using General Software BIOS or Annabios (previously called the XT BIOS).

The move from 16-bit and 32-bit processors added a new level of complexity. Designing in new technology like PCI and USB also added to the challenge. Operating systems played an important role to address memory management and provide a foundation to develop applications. Eventually embedded board manufacturers eliminated the need to build a custom PC with lower cost and long life motherboard solutions. Embedded development became simpler. Customers would pick an off-the-shelf PC board and design custom ISA or PCI add-on cards to address their application needs. A developer's only contact with BIOS firmware would be to make change requests to the board vendor for features like a custom boot splash screen, boot order, or other hardware configuration presets. At the turn of the century BIOS development had become a thing for only board manufactures to focus on.

## *1.1 Rise of the Unified Extensible Firmware Interface (UEFI)*

The old 16-bit BIOS had limitations such as drive size limitations, memory space, and 16-bit development tools were going away. The biggest issue was board testing. A board vendor's cheapest method to test a board was to make sure that the Windows operating system could boot successfully. Intel started an initiative in the late 90s to start addressing the old BIOS problems with their high-end server processors. The Extensible Firmware Interface (EFI) focused on the Itanium processors, and in 2005 the Unified Extensible Firmware Interface (UEFI) Forum (www.uefi.org) was created to continue specification development with the larger community. Today, more PCs are implementing the boot firmware using UEFI.

The old BIOS would perform a basic chip configuration, power-on-self-test, and then look to boot an operating system from one of the internal or attached drives. This was performed in x86 real-mode, which only had the 1MB addressable memory space. The OS was responsible for switching the system from real mode to protected mode to access the full memory range.

UEFI changes the process by putting the system into protected mode right from the start. An EFI shell allows board vendors, technicians, and end users to run EFI applications that test the board features. In essence, UEFI + EFI shell are a mini operating system. There are many other features that UEFI offers such as modular design, faster boot time, and a replacement for the old master boot recorder (MBR) to allow access of drives over 2TB.

The old BIOS hasn't gone away. Many operating systems still rely on BIOS services to perform certain operations. BIOS vendors implement the basic BIOS functionality as part of UEFI.

## *1.2 Intel® Atom™ Game Changer – E6xx Series*

In 2008, Intel introduced the Intel® Atom™ processor, and it has been a game changer in the embedded market ever since. With low power, multiple performance options, and low cost, Intel began to create new

classes of devices like Netbooks. Nearly every embedded PC board vendor offers an Intel® Atom™ solution. Different series offer different solutions:

D Series is for desktop systems like thin clients. N and Z Series are for Netbooks and tablets that require low power.

The E-Series is the new game changer for the market. Harkening the days of the 80386EX processor, but with a much more complete and stable solution, the E-Series is a system-on-a-chip, SoC, device targeted for embedded systems. The new SoC implementation includes graphics engine, memory controller, GPIO, and SPI port for SPI flash BIOS. The proprietary bus chipset interfaces from older processors has been replaces with open PCIe standard. Developers can now add PCIe FPGA solutions to support a variety of I/O and applications from mobile to industrial control.

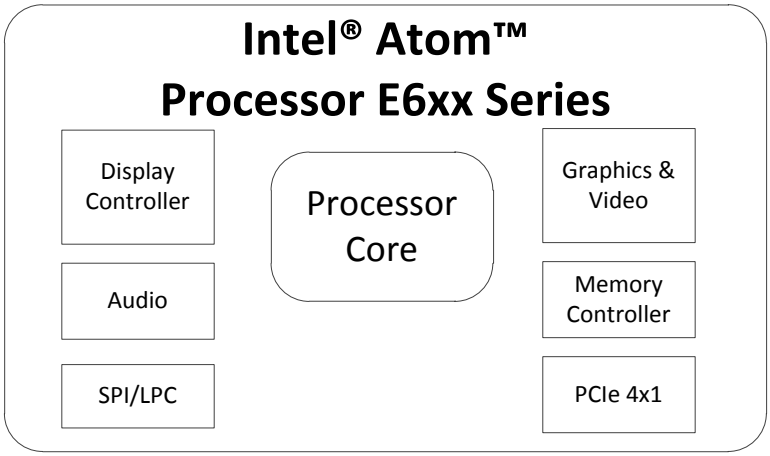



**Figure 1.1 E6xx Series Integrated Solution**

The integration and flexibility of the new E6xx series, allows embedded developers to create their own unique IA solutions for fixed-function devices.

## *1.3 Simplified Firmware: Intel® Boot Loader Development Kit*

Designing with the E6xx series means firmware comes back into the equation. A traditional BIOS is too slow and challenging to develop for a fixed function device. Again, UEFI offers a fresh open solution, but it takes time to develop custom firmware from scratch.

To help speed up firmware development, Intel has developed the Intel Boot Loader Development Kit (BLDK). The BLDK is derived from the Intel® UEFI Development Kit 2010 (Intel® UDK2010). The UDK2010 is based on the latest UEFI specification. As of this writing the version supported is UEFI version 2.3. The BLDK also supports the Intel® UEFI Development Kit Debugger Tool and UEFI 2.0 shell.

Since the focus is to develop fixed-function devices rather than a full desktop system, the BLDK provides the basic common low-level CPU and chipset initialization in library format. Traditional BIOS services are not available, which limits the operating systems that it can boot. The rest of the firmware is available as source code for modification. Firmware packages are supplied as a starting point. Currently, firmware packages available for different reference boards are: E6xx with EG20T controller hub known as Crown Bay and the E6x5C known as Foxbrook.

The BLDK comes with a GUI configuration tool to build the firmware image and apply customizations to the firmware, such as fast boot with silent startup, preset the boot order, and preset messages to be displayed.

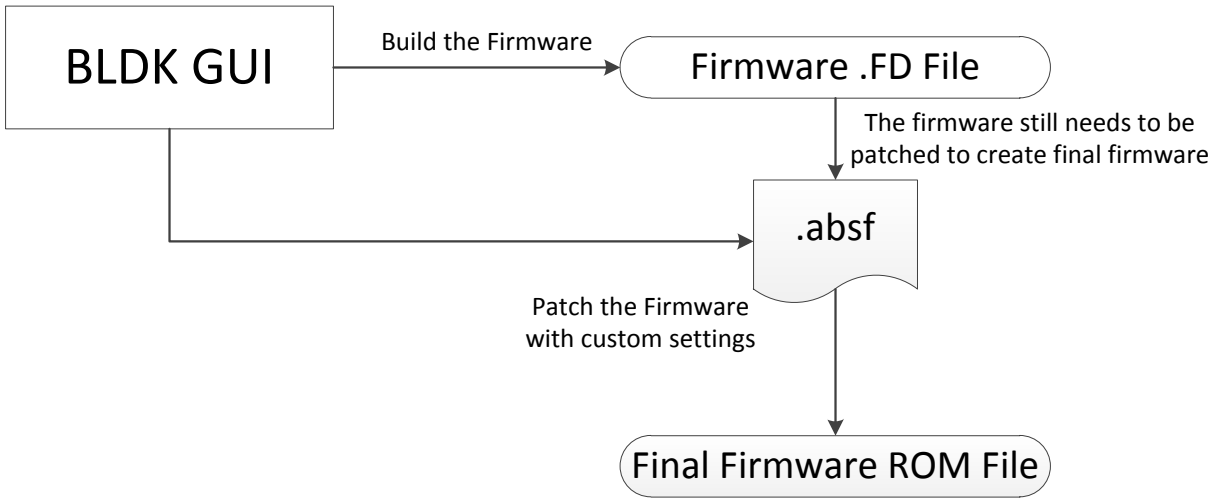



**Figure 1.2 BLDK Development Process**

The BLDK can boot an operating system from SATA, SD flash drive, and USB flash drive. The supported kernels and operating systems include UEFI Shell 2.0, Linux operating systems, and Windows® CE. Since there isn't a traditional BIOS module, operating systems like DOS and Windows® XP/Vista/7 desktop, which rely on BIOS services, are not supported.

## *1.4 About this Paper*

The BLDK has several different documents available. The purpose of this paper is to bring together the different steps to setup the tools and build/deploy the firmware. The BLDK supports Windows and Linux development systems, but using a Windows development machine will be the focus of this paper.

***Note**: Expect future revisions of the BLDK to require different tools and support. Please, check with the Intel Embedded Developers website for the latest information: edc.intel.com.*

## *1.5 Conclusion: Full Circle*

Custom building a PC was once thought a thing of the past, but the Intel® Atom™ E6xx series offer new opportunities to create custom devices. The BLDK brings back firmware development with more features and easier build tools. The following sections discuss downloading the tools, setup of the build environment, and building the firmware.

# 2 Development Requirements and Downloads

There are a number of tools needed to build and debug the UEFI firmware. This section lists the different tools needed and options for Windows host machine.

## *2.1 BLDK Downloads*

The first set of tools to download is the BLDK Development Application and any available Reference Firmware packages.

| Description | Website |
|---|---|
| Intel® BLDK Development Application (Windows) | http://edc.intel.com |
| Intel® BLDK Code Bases (Reference Firmware Packages)<br><br>As of this writing there are a couple of Reference Firmware Packages available:<br><br>• Intel® BLDK Core for Crown Bay - Windows<br>• Intel® BLDK Core for Foxbrook | http://edc.intel.com |

You should also download the release notes for the application and the packages.

## *2.2 Development Machine Software*

For a Windows development environment, the following table lists the required tools. The tools are older versions, but these have been tested with the BLDK.

| Description | Website |
|---|---|
| Visual Studio 2008 | This is an older version of Visual Studio and it is available as part of MSDN subscription:<br><br>http://msdn.microsoft.com<br><br>You will have to burn the ISO to a CD. |
| Windows Device Driver Development Kit version 3790.1830 | This is an older version of the DDK required to build the firmware image:<br><br>http://download.microsoft.com/download/9/0/f/90f019ac-8243-48d3-91cf-81fc4093ecfd/1830_usa_ddk.iso<br><br>You will have to burn the ISO to a CD. |

| Description | Website |
|---|---|
| ACPI Component Architecture tools for Windows Version 20070508 | http://www.acpica.org/downloads/Version_20070508.php<br><br>The ACPI Component Architecture Project defines a reference implementation of ACPI. Version 20070508 is the oldest version available for downloaded. The are 3 files available to download.<br><br>Only iasl-win-20070508.zip is needed. |

## 2.3 Flash Programming Tools

A flash programmer comes in handy if changes made to the firmware fail to boot the image. You can always reprogram a firmware version that works. The Intel® BLDK documentation mentions DediProg in-system flash programmers SF100 or SF600 (or similar device): http://www.dediprog.com/home.

## 2.4 Debug Tools

If you are going to add on or modify the firmware, you will need some debugging tools for development. There are 3 levels of debugging.

### 2.4.1 Port 80 / Serial

The first is the simple Port 80 and serial output. Port 80 cards were popular for BIOS development and they can still be used with UEFI. You can see where issues arise by monitoring the Port 80 output. Most motherboards don't have a port 80 output built on board, but there are different solutions for ISA and PCI available via searching the Internet. Elston Systems, Inc (http://www.elstonsystems.com/) is a company that specializes in PC test equipment.

Serial output can be viewed with a serial terminal program. Print statements in the code can be output information. Starting with Windows Vista, Hyperterminal is no longer available so you will have to download and/or purchase a terminal program:

| Terminal Program | Website |
|---|---|
| HyperTerminal | http://www.hilgraeve.com/ |
| TeraTerm | http://ttssh2.sourceforge.jp/ |
| PuTTY | http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html |

### 2.4.2 Software Debugger

Source code level debugging is supported by the BLDK. The Intel® UEFI Development Kit Debugger Tool interacts with WinDBG from the Microsoft® Debugging tools from Windows to provide the ability to step through the code. The firmware will have to include the debug agent in the build to interact with the remote debugger.

| Tool | Website |
|---|---|
| Intel UEFI Development Kit Debugger Tool | You will want to download both the Windows version and the User Manual.<br><br>http://www.intel.com/content/www/us/en/architecture-and-technology/unified-extensible-firmware-interface/intel-uefi-development-kit-debugger-tool.html |
| Debugging Tools for Windows (WinDbg)<br>32-bit Version 6.11.1.404 | If you do search for these tools, you will be lead to the newer Windows 7/8 DDK. A specific version is need, which can be download from previous version of the tools only site:<br><br>http://msdn.microsoft.com/en-us/windows/hardware/gg463016 |

### 2.4.3 JTAG Debugging

The most complete debugging solution is going through the JTAG port. The debug agent is not needed in the firmware image. Where software debuggers only work when the debug agent is loaded, JTAG debuggers start working from the reset vector. There are a number of JTAG vendors that have support for the Intel® Atom™ processor.

| JTAG Vendor | Website |
|---|---|
| Arium | http://www.arium.com/ |
| Green Hills Software | http://www.ghs.com/ |
| Lauterbach | http://www.lauterbach.com/frames.html?home.html |
| Macraigor Systems | http://www.macraigor.com/ |
| Wind River | http://www.windriver.com/ |

## *2.5 Target Hardware*

Actual target hardware is needed to test the firmware. The BLDK has reference firmware packages for Crownbay (E6xx Series with Intel® Platform Controller Hub EG20T) and Foxbrook. The Crownbay platform is available from Intel - http://intel.ententeweb.com/bsps/products.asp. A less expensive version is available from Inforce Computing, Inc. - SYS9400 Reference Platform - http://www.inforcecomputing.com/SYS940X_ECX.html. The SYS9400 will be used as the target for the firmware deployment exercises.

# 3 Development System Setup – Step-by-Step

The following section covers setting up the Intel® BLDK development environment on a Windows 7 64-bit machine. Make sure that you have downloaded Visual Studio 2008, Windows Device Driver Development Kit version 3790.1830, ACPI Component Architecture tools for Windows Version 20070508 (iasl-win-20070508.zip), Intel® BLDK Development Application (Windows), and an Intel® BLDK Code Base Reference Firmware Package. For this example the Crownbay package will be used for the Inforce® SYS9400 Reference Platform. There are 5 steps for development system setup.

## *3.1 Step 1 Visual Studio Installation*

The current BLDK calls out Visual Studio 2008. As of this writing, Visual Studio 2010 hasn't been tested. The following are the steps to install Visual Studio 2008.

1. Insert the CD that contains Visual Studio 2008. For this example, I used the Visual Studio 2008 Standard edition.
2. If asked to run the autrun.inf, select yes.
3. Setup will start, click Next on the first screen to begin.
4. Read the licensing terms. If you agree select accept the terms, and click Next.
5. The next screen presents you with some installation options. Only Visual C++ and Visual C# are needed for installation. Uncheck the other options, and click Install to begin the installation process.

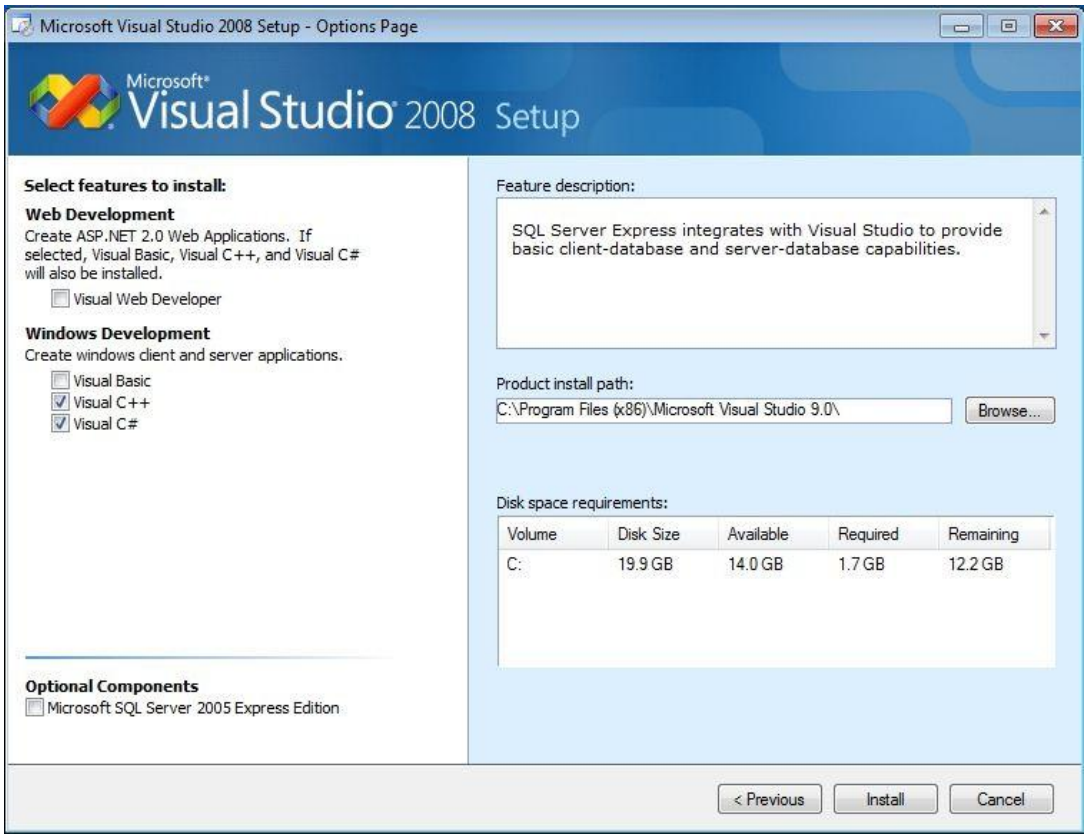



**Figure 3.1 Installing Visual Studio 2008**

6. Once the installation is completed, click Finish.

The BLDK Setup Guide recommends setting the PATH variable to include the path to nmake. The BLDK application doesn't have a hard coded path to nmake. If you try to build the firmware, you will get a build error.

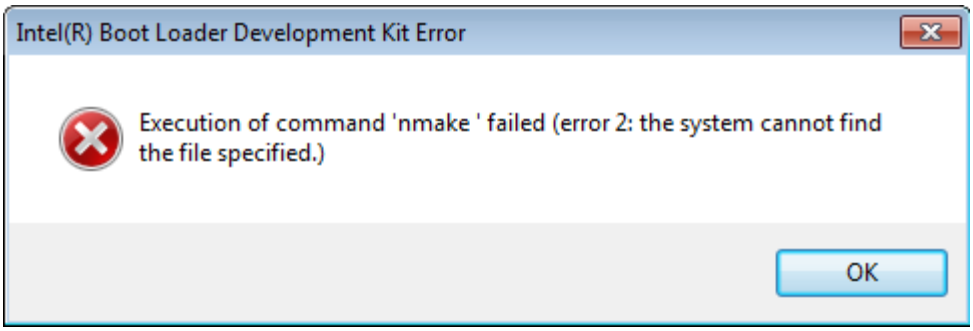



**Figure 3.2 Path variable not set correctly**

Since the E6xx series is a 32-bit processor, the 32-bit build tools must be used. The Path to the 32-bit version is located at C:\Program Files (x86)\Microsoft Visual Studio 9.0\VC\bin. Below are the steps to add to the system path:

7. Open Control Panel, switch to icons to make the usage easier.
8. Open System control Panel.
9. Click on Advanced system settings.
10. A System Properties dialog appears. Click on the Environment Variables button.
11. The Environment Variables dialog appears. Scroll down the list and select the Path Variable.

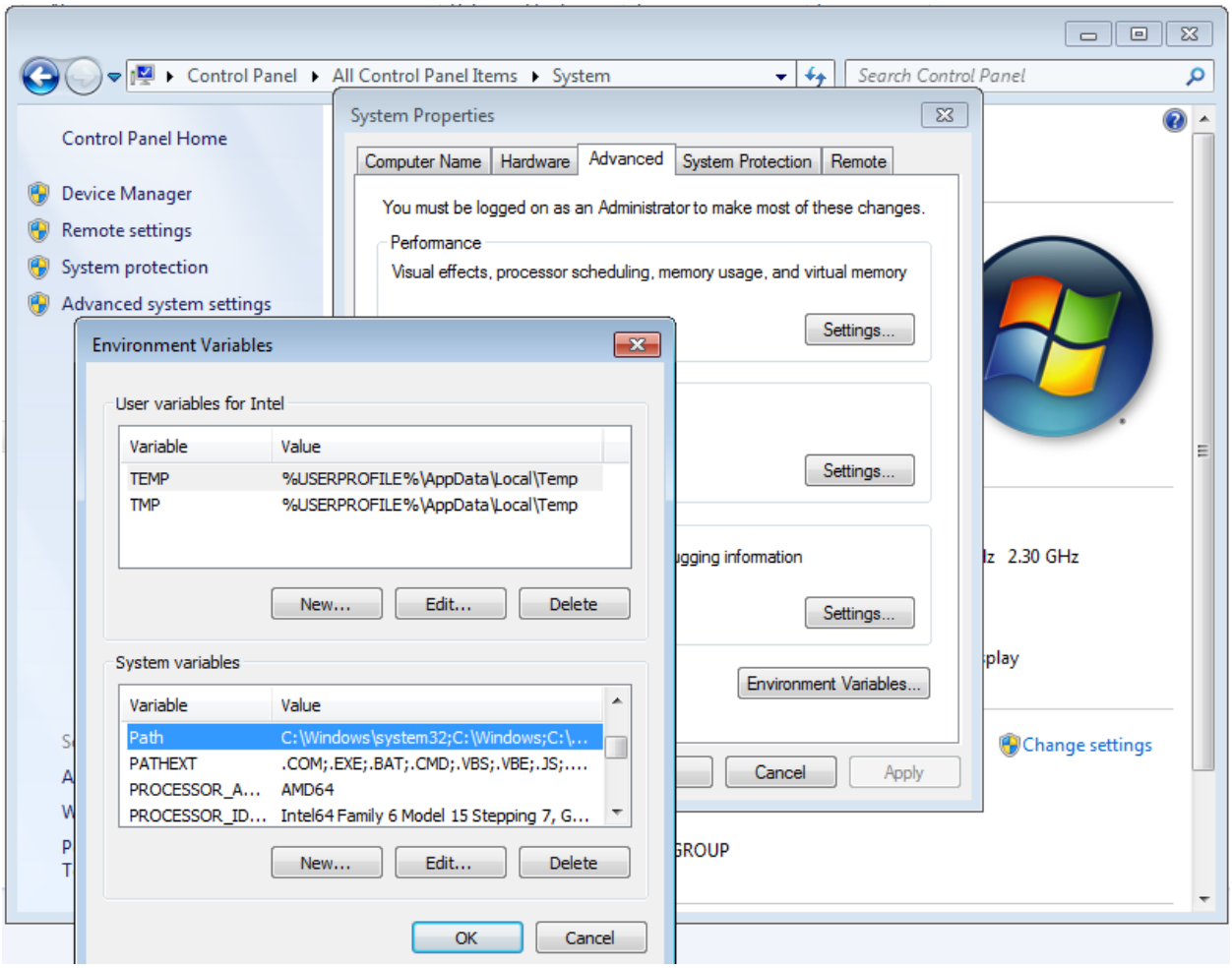



**Figure 3.3 Setting the Path variable**

12. Click Edit.
13. Add the following to the end of the line:

    **;C:\Program Files (x86)\Microsoft Visual Studio 9.0\VC\bin**

14. Click OK.
15. Click OK to close the Environment Variables dialog.
16. Click Ok to close the System Properties dialog.

## *3.2 Step 2: Windows DDK Installation*

The current BLDK specifically calls out the Windows Server 2003 SP1 DDK. Like Visual Studio, the latest versions haven't been tested. The following are the steps to install the DDK.

1. Insert the CD that contains the DDK.
2. The autorun will come up. If you let the system autorun, you will get a webpage, where you can run setup from. The alternative is to run setup.exe from the root of the CD. Run Setup to install the DDK.
3. You will be asked for security information to run setup.exe click OK to continue.

4. Click Next on the first wizard screen.
5. Read the licensing terms. If you agree select “I Agree”, and click Next.
6. The BLDK setup guide wants the DDK to be in the default path: C:\WINDDK\3790.1830. Click Next.

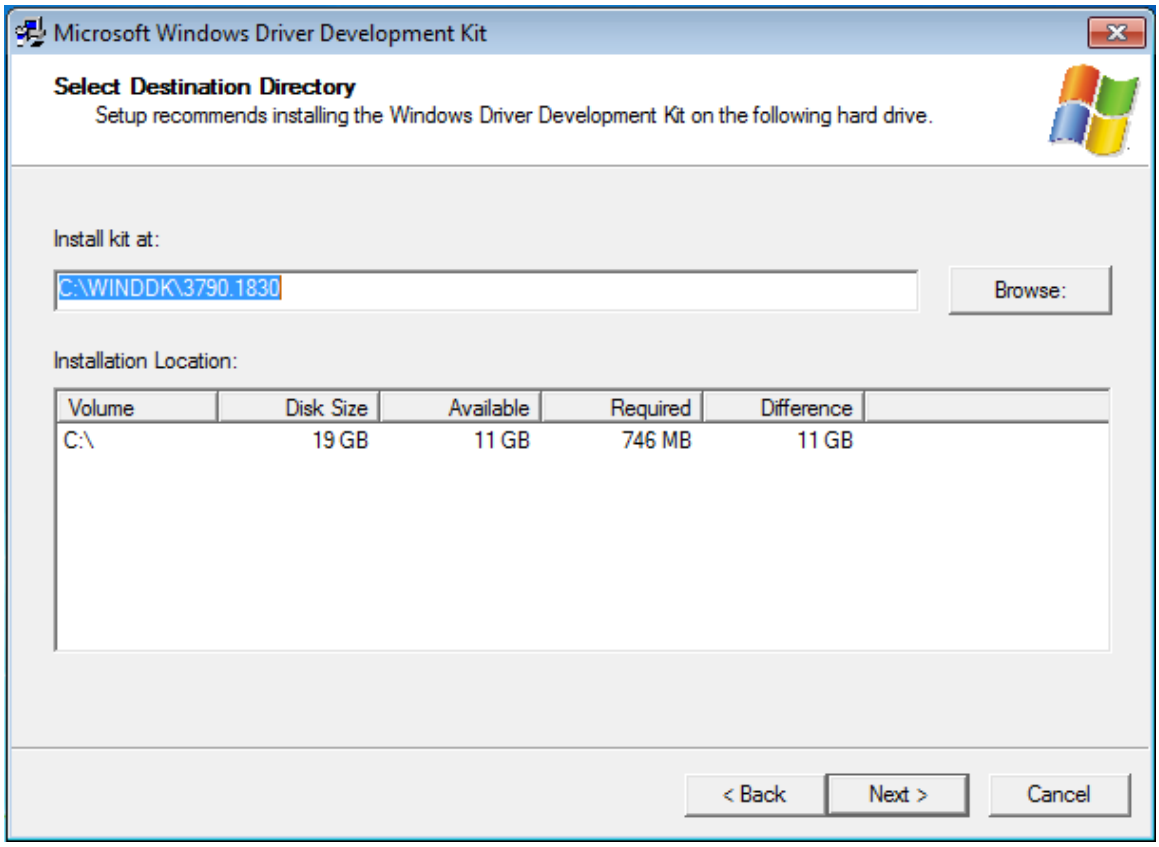



**Figure 3.4 Installing the DDK**

7. The next screen will ask if you want to install some addition components. Keep the defaults and click Next.

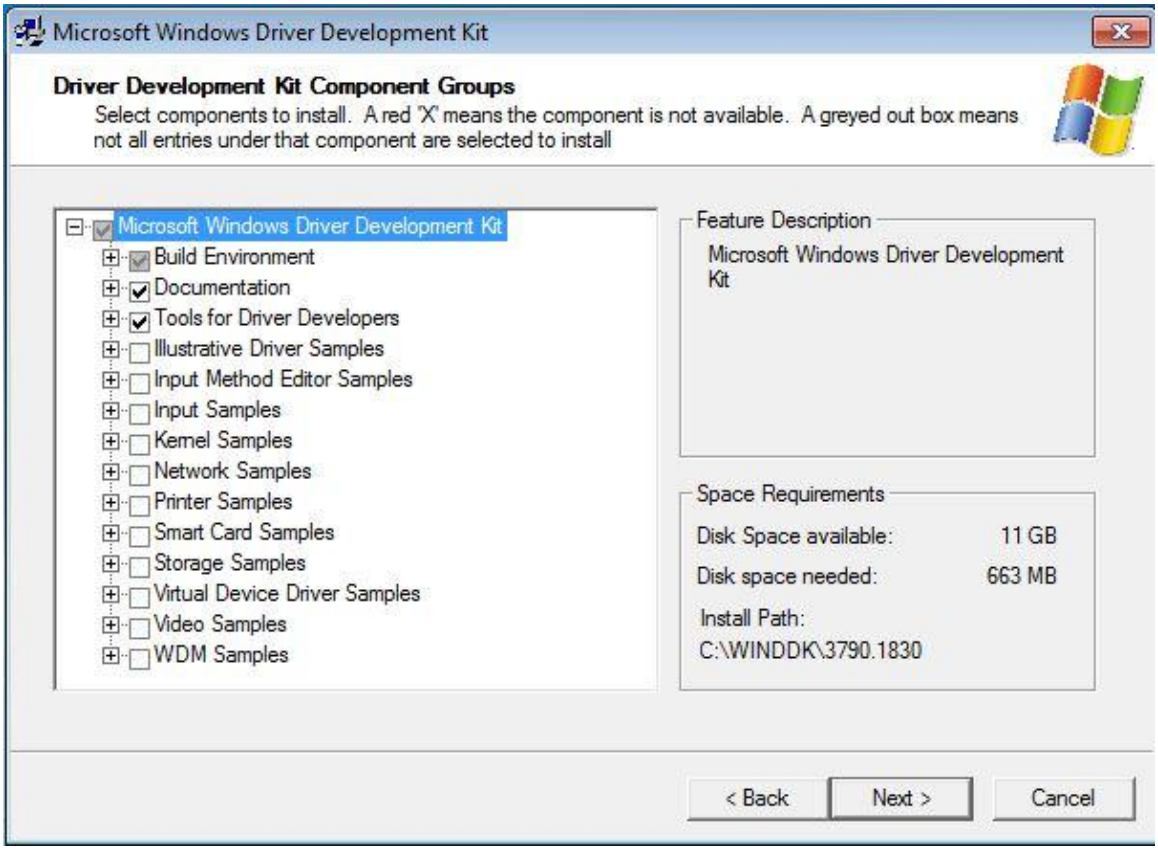



**Figure 3.5 Keep the DDK defaults**

8. On the confirmation page click Next to begin the installation.
9. Click Finish when the installation is completed.

## *3.3 Step 3: ACPI Component Architecture Tools Installation*

The setup steps for the ACPI Component Architecture Tools are as follows:

1. Open file Explorer
2. Create a directory at the root of C:\ASL
3. Extract the contents of the iasl-win-20070508.zip file and copy the contents to c:\ASL.

## *3.4 Step 4: Intel® BLDK Application Installation*

1. Extract the contents from the BLDK Application zip file.
2. Run the Intel(R)_Boot_Loader_Development_Kit.exe file to start the installation.
3. A screen will appear asking to extract the files to a specific folder, click Extract.

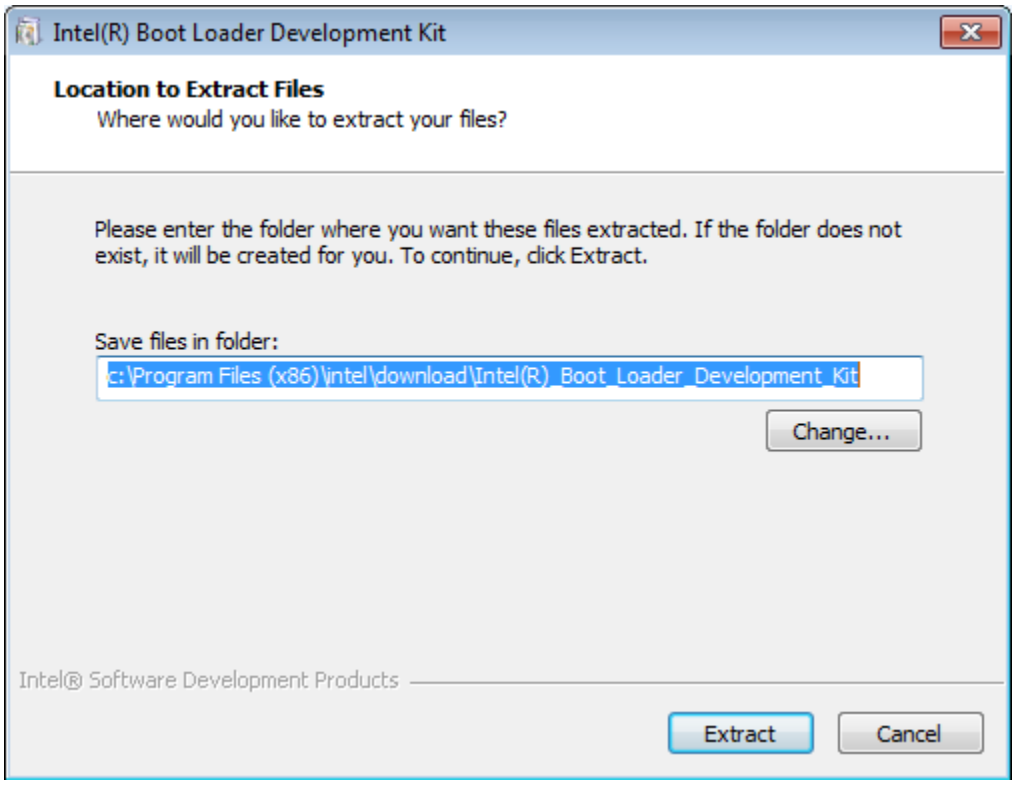



**Figure 3.6 Installing the BLDK to the default directory**

4. A wizard will then start to begin the installation process. Click Next.

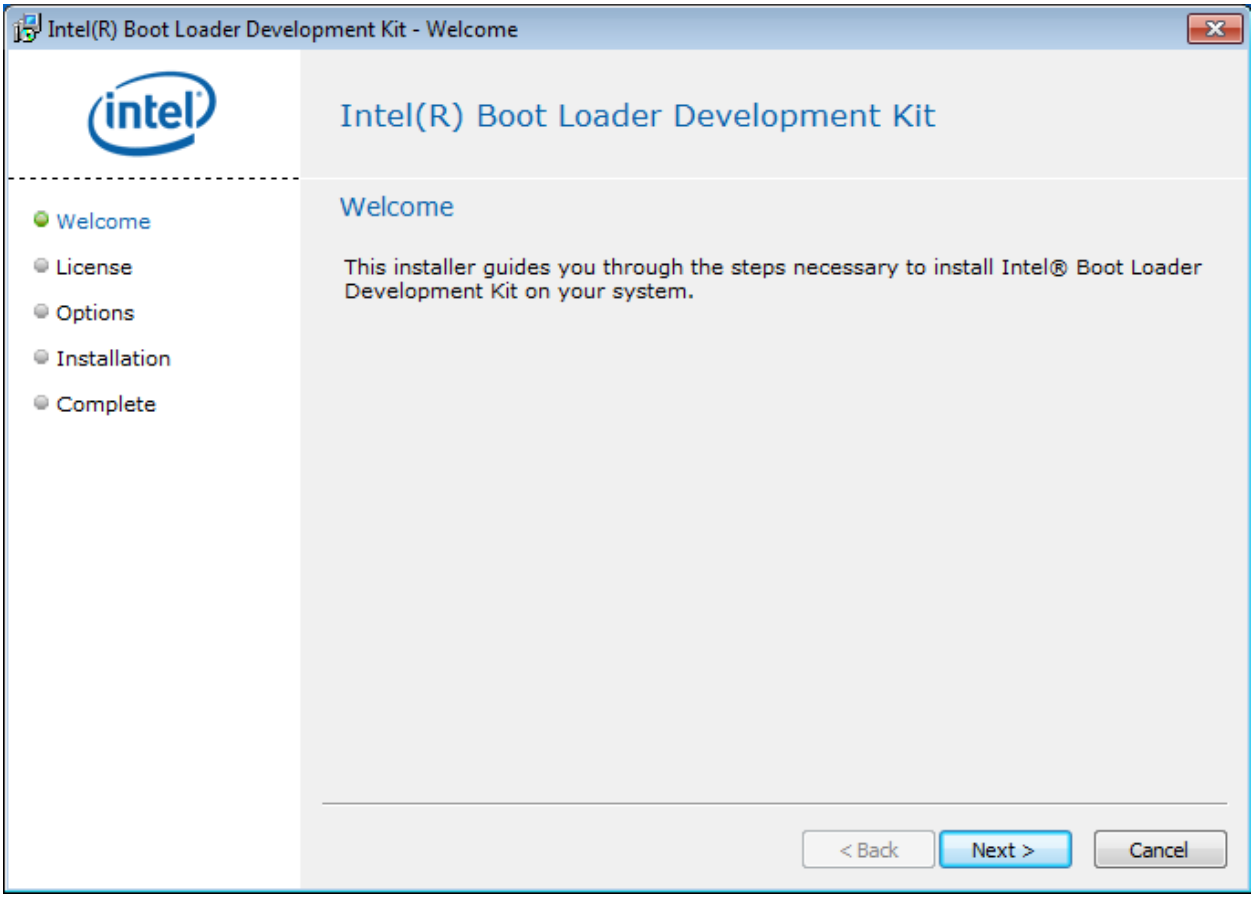



**Figure 3.7 BLDK Installation Wizard**

5. Read the licensing terms. If you agree select accept the terms, and click Next.
6. The next screen asked for Full or Custom installation. Custom only lets you set the installation directory. Select Full and click Next to begin the installation.
7. Click Finish when completed.

## *3.5 Step 5: Installing the Code Base Package*

The following steps are for installing the Crownbay Code Base package.

1. Run the CB-EDKII-PostGold-2.3.6.7.exe
2. The EXE is a self-extracting executable, set the path to c:\Crownbay and click Extract.

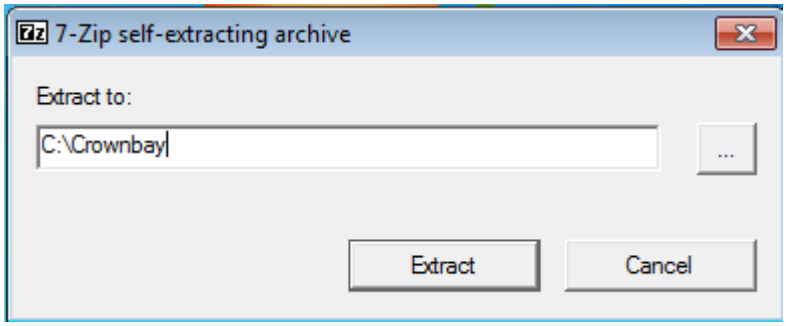



**Figure 3.8Setting up the Crownbay platform package**

The host machine is now setup with all the tools to build the firmware. The Crownbay Code Base contains a pre-built flash file of the firmware. You may want to keep the CB-EDKII-PostGold-2.3.6.7.exe handy to install the tools for a new project or recover the pre-built flash images. A reboot of the host system is always a good last step.

# 4 Building and Customizing the Firmware

Figure 4.1 shows an overview of the BLDK development process. The BLDK Application can patch an existing firmware image with custom settings. The BLDK Application provides a GUI interface to a Boot Setting File (BSF). The BSF is a text metadata file that is part of the Firmware Package. The file contains the settings used to customize the firmware image. Never modify the .bsf file as you could create an image that doesn't boot. When you create a new project a copy of the BSF file will be create (.absf).

The BLDK Application can also be used to build the firmware from source code and then patch the resulting binary with custom settings. Building the firmware from the supplied source code simply rebuilds the binaries that are already included with the BLDK. Therefore, the main motivation for building from source code is if you are customizing the source code. Whether rebuilding the original source code or building customized source code, you still must patch the resulting binary to apply your custom settings.

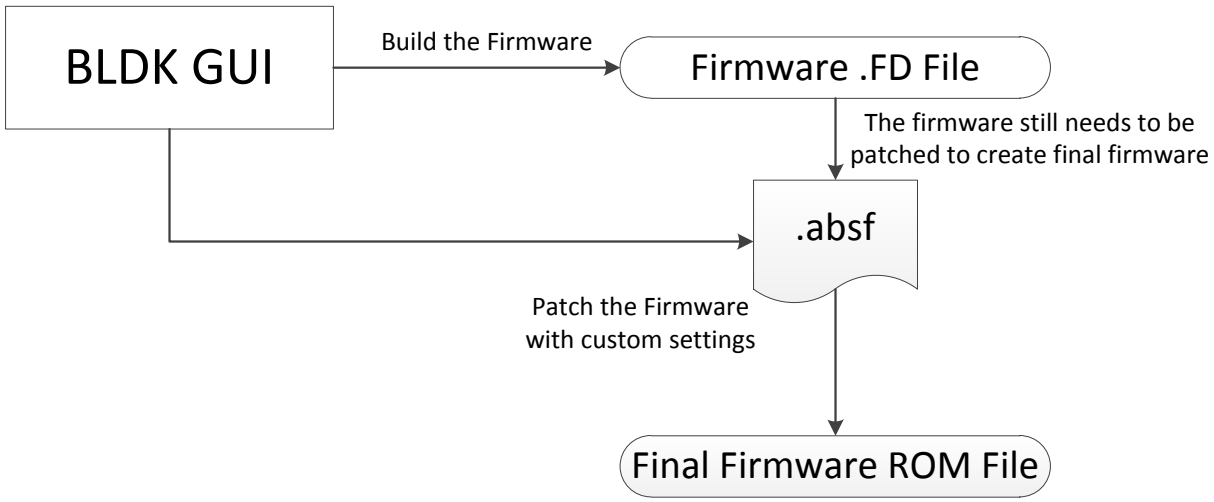



**Figure 4.1 - BLDK Build / Development Process**

The following steps demonstrate the development process using the Crownbay firmware package.

## *4.1 Step 1 Create a new Project*

Once all the tools have been installed, you can open the BLDK Applications and create a project.

1. Open the Intel® Boot Loader Development Kit.
2. The first time the application runs it will show a licensing page. Read the licensing terms. If you agree select accept the terms, and click the button to continue with the BLDK Application.
3. The BLDK Application has four panes. The left pane is the navigation window, the center pain is the main window to change settings, and the right pain provides on-demand help for the items in the main window. The bottom pane is the output window for the build activity.
4. From the menu, select Project->New Project…
5. The main window will show you the project properties. Provide the following:

| Project property | Description | Value |
|---|---|---|
| Project Name | Name of the project | **BLDK1** |
| Project file directory: | The project file (.ews) will be created in this directory. | **C:\Crownbay** |
| Workspace directory: | Location of the files to build the firmware image | **C:\CrownBay\CrownBayPlatformPkg** |
| Image configuration file | The location of the baseline BSF file for the package. | **C:\CrownBay\CrownBayPlatformPkg\FV\tc.bsf** |

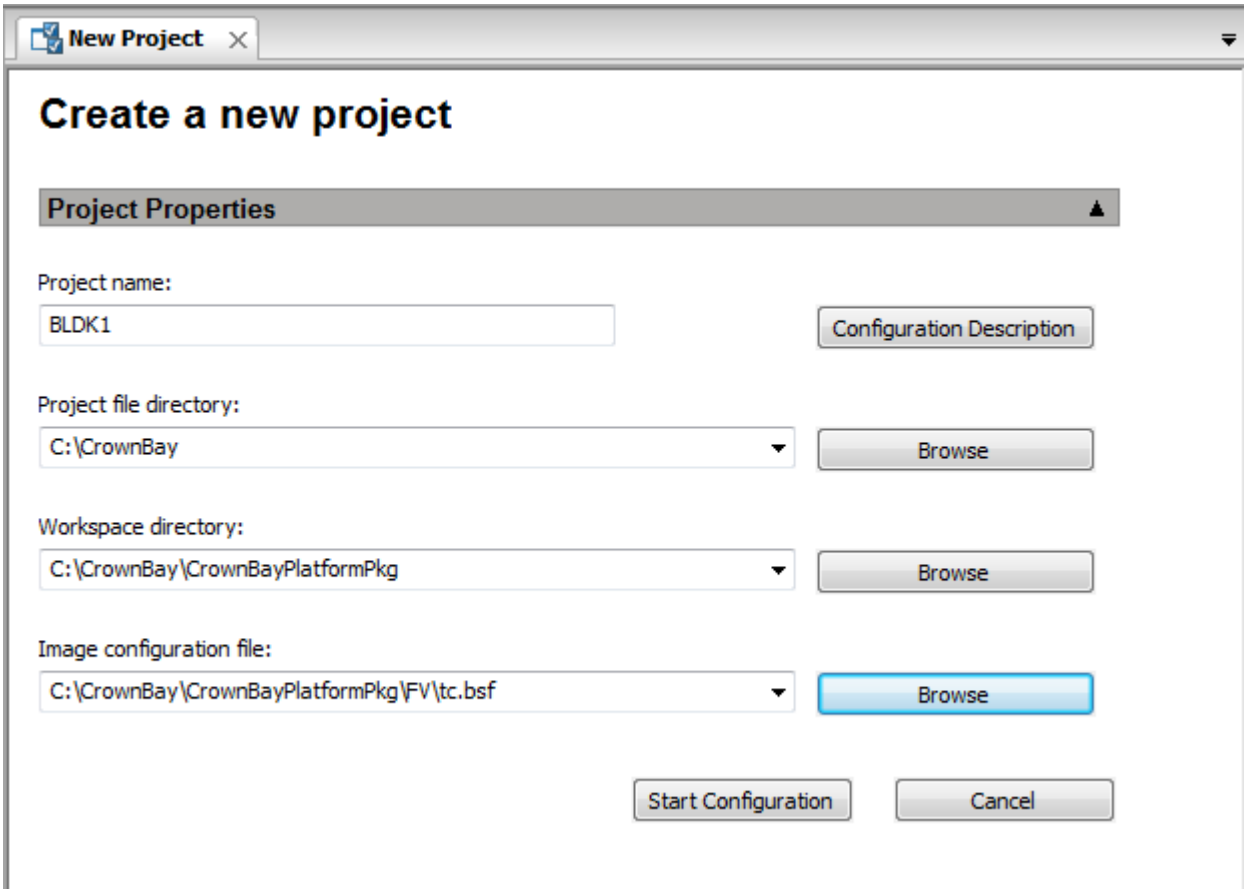



**Figure 4.2 Creating a new project**

6. Click the Start Configuration button.
7. The next screen will ask for a firmware feature regarding source debugging. You can set this option later. Click the Create Project Button

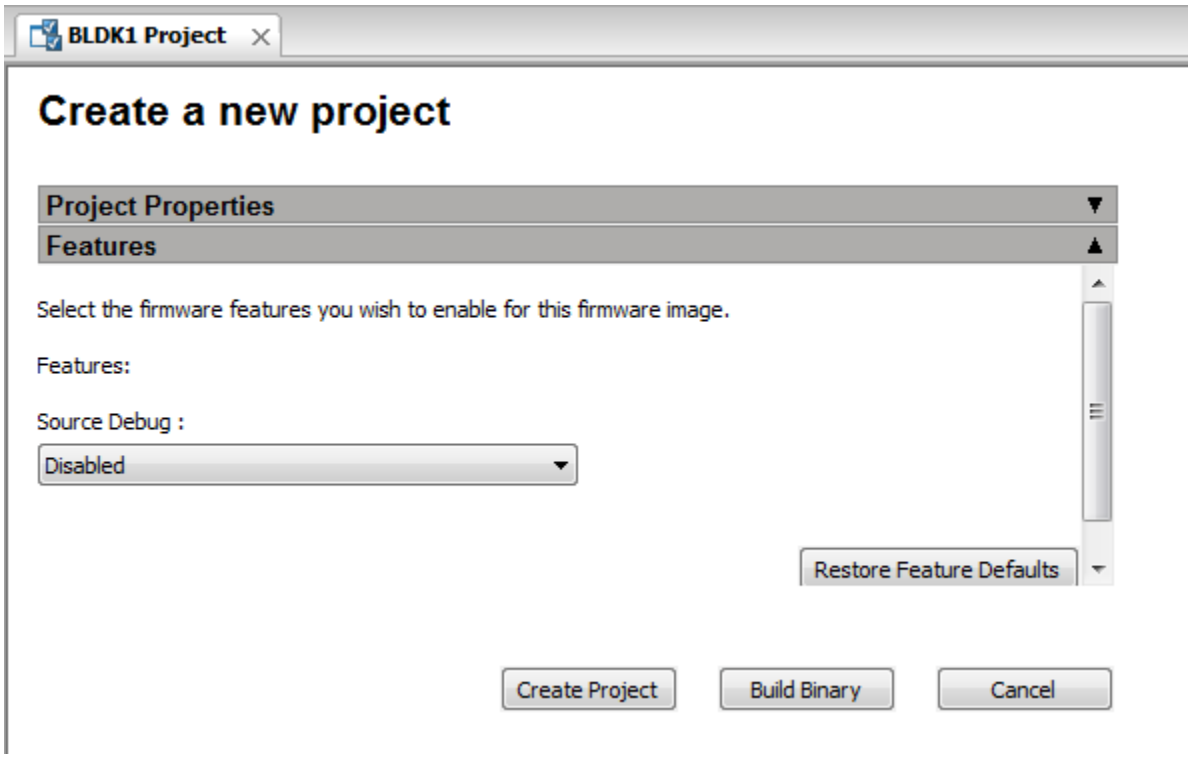



**Figure 4.3 Keep the defaults for the project**

## *4.2 Step 2: Modifying the BSF file*

Once the project has been created, you can customize your firmware to fit your application. Some settings require knowledge about your hardware platform.

1. In the Navigation, click on Boot.
2. Set the Fast Boot to **Enabled**.
3. Change the Firmware banner to **ACME Systems**.
4. Change the Copyright information to **Copyright (C) 1999-2011 ACME Corporation. All rights reserved**.

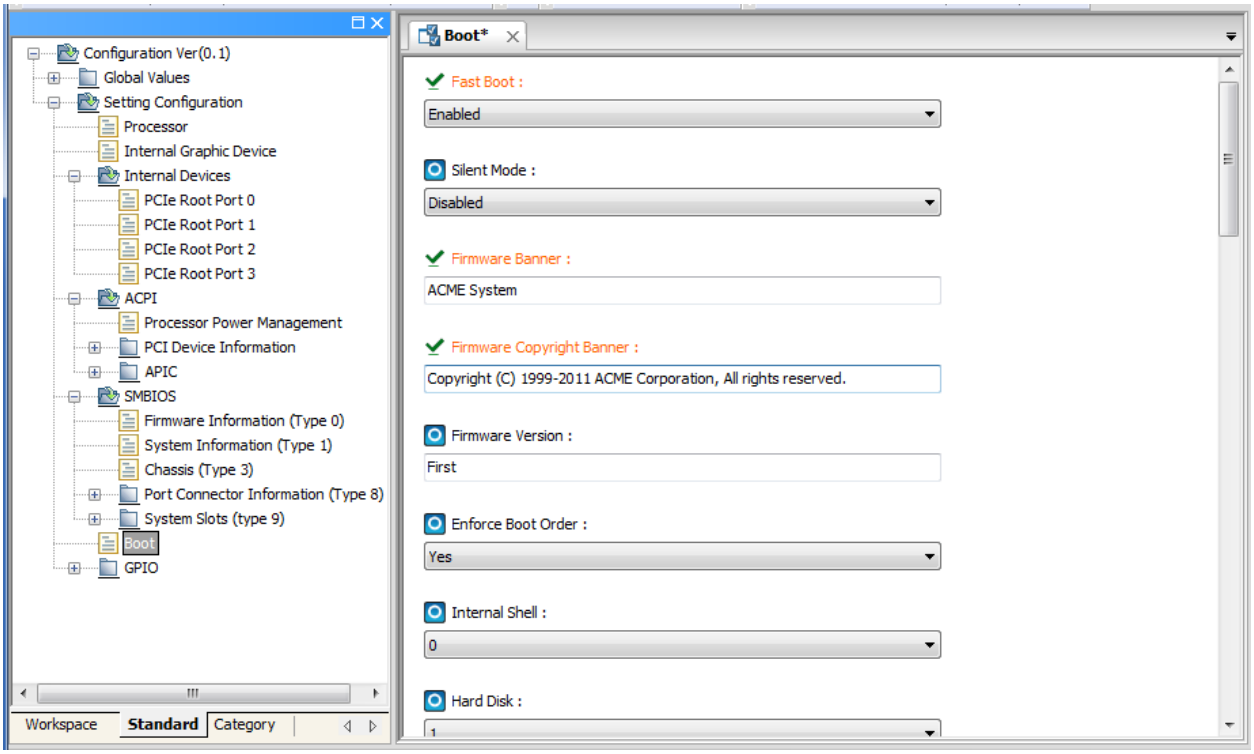



**Figure 4.4 Customizing the Firmware**

5. From the menu, select Project-> Save Configuration. The BLDK1.ews file will be saved and the tc.absf file will be created in the same directory as the original tc.bsf file - C:\CrownBay\CrownBayPlatformPkg\FV.

***Note**: TC means Tunnel Creek, code name for the E6xx processor.*

## *4.3 Step 3 – Build the Firmware*

Based on Figure 4.1, there are two options to build the firmware. The first option is to build the final firmware from an existing binary. The Firmware Package comes with existing binaries already created.

The second option is to build the project from source and libraries. The main reason you would want to build from source code is to make a change to the source code or add a driver. You can either build a release build or a debug build. The debug build will include the debug agent and symbols for source level debug. Once the build process is completed, you can create the final firmware file by patching the firmware with the custom settings defined in the .absf file.

***Note**: BLDK Application version 2.0.1.18265 has some quirks with creating the final image. Our workaround will differ from the BLDK documentation.*

### 4.3.1 Option 1: Create the Final Firmware Based on an Existing Binary

The .asbf file can be used against an existing firmware binary to create a final ROM image. This build method is ideal when you want to make quick changes to the configuration settings for an existing binary without having to rebuild the whole binary.

There is an issue. The BLDK Application wants the binary file (.fd) in the same directory as the .absf file when the project is open. Since the crownbay.fd file is not in the same directory as the tc.absf file, an error will appear if you try to create the final firmware file with the project open.

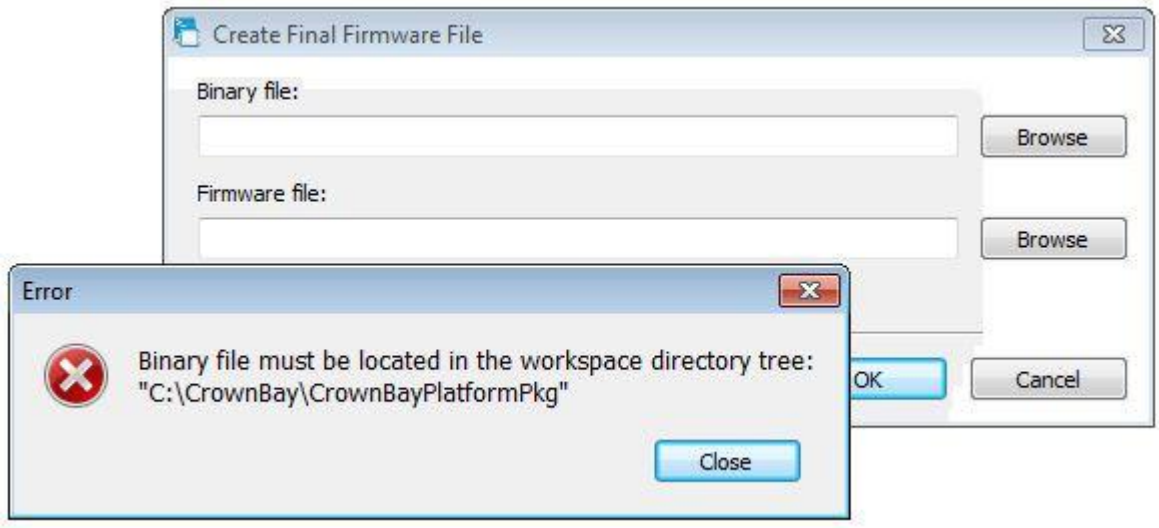



**Figure 4.5 Reference Error in the BLDK application**

If you get this error dialog shown in Figure 4.5, close the Error dialog, and the Cancel the Create Final Firmware File dialog. Then follow these steps that work around the issue.

1. Close the Project. From the menu, select Project->Close Project.
2. A dialog will appear asking to save the configuration, click Yes.

The Code Base project comes with a pre-built binary in the \Build\CrownBayPlatform directory. We will use this binary to create a final ROM file.

3. From the menu, Select Build-> Create Final Firmware Image.
4. The Create Final Firmware File dialog appears. Select the location of the binary. You will have to change file type to All Files. For this example,

   C:\CrownBay\Build\CrownBayPlatform\RELEASE_VS2005x86\FV\CROWNBAY.fd.

Once you select the file the Firmware file will automatically fill in - CROWNBAY.rom.

5. Select the As built File (.absf). For this example,

   C:\CrownBay\CrownBayPlatformPkg\FV\tc.absf

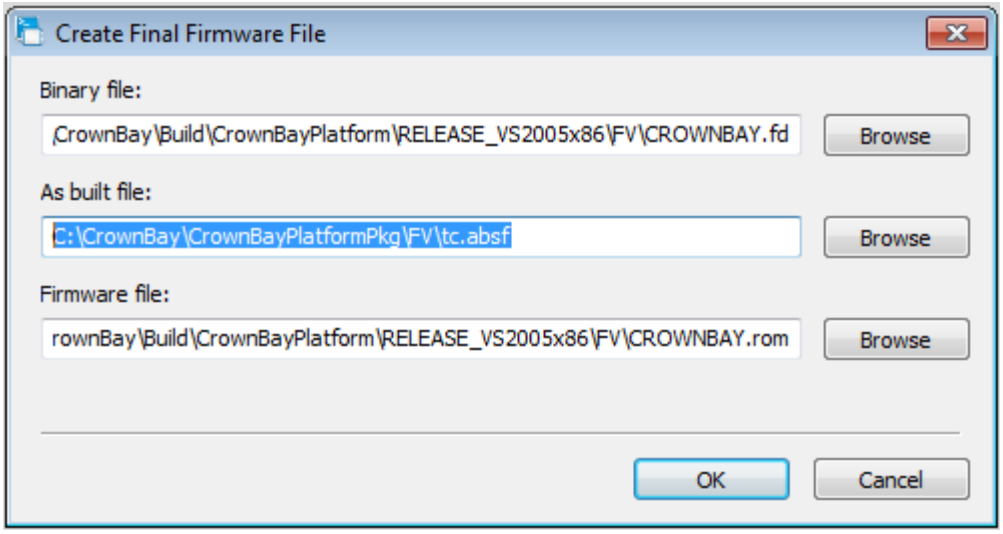



**Figure 4.6 Creating the final firmware**

6. Click OK to build the firmware. A message will appear when the firmware file has been created.

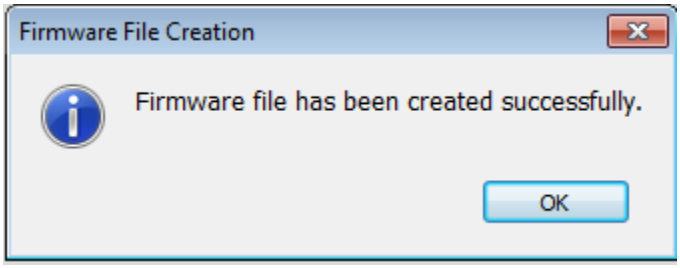



**Figure 4.7 Firmware Competed**

### 4.3.2 Option 2: Build the Project and then create the final Firmware with the Customizations

If you have added or changed source code, or just want to build the image from source code, you can build the image from the BLDK application.

1. If you closed the project, open the project (.ews) file. You will be asked for the .absf file.
2. To build the image, from the menu, select Build->Build.
3. The build process will take a few minutes. A dialog will appear when the build is completed. Click Ok.

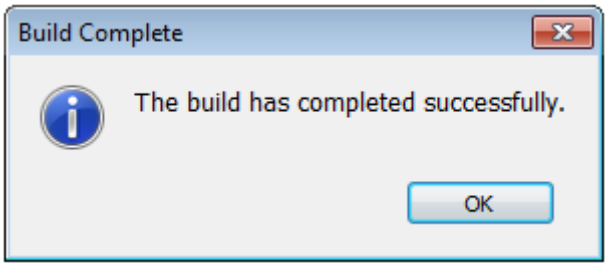



**Figure 4.8 Source code build completed**

The resulting binary file will be found here:

**C:\CrownBay\Build\CrownBayPlatform\RELEASE_VS2008x86\FV\ CROWNBAY.fd**

The RELEASE_VS2008x86 has a specific meaning. RELEASE means release build. VS2008 indicates which build tools were used; in this case Visual Studio 2008.

Building the image doesn't include the customization setting defined by the GUI. The customization settings only get applied when you patch the firmware. The last step is to create the ROM file. If your try to convert the binary to a final ROM file with the project open, you will get an error.

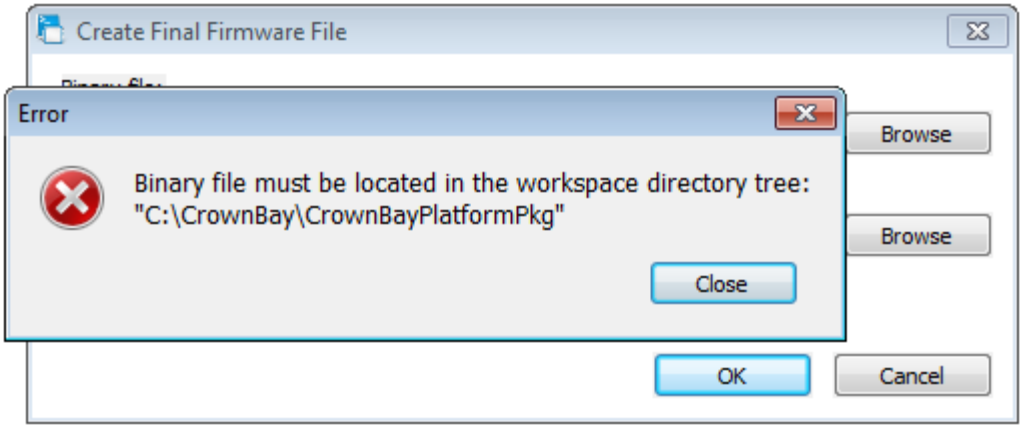



**Figure 4.9 Reference Error in the BLDK application**

You can either move the CROWNBAY.fd to the location of the .absf file, or close the project and do the convert per section 4.3.1 to create the newly built binary.

# 5 Deployment and Other Tests

There are two options to deploying the firmware to the platform. The first option is to use a flash programmer tool to program the CROWNBAY.ROM file directly to the flash chip. The second option is to boot to the UEFI shell and use the SpiUpdate.efi utility to program the CROWNBAY.ROM into flash.

A USB flash disk, a NULL modem serial cable, and a serial terminal application are needed to complete the steps in this section.

## *5.1 Serial Flash Programmer*

The example provided here uses the DediProg SF600 programmer with SOP8 socket adapter. The Inforce SYS9400 doesn't have SPI port available to program the flash on board, but the SST25VF016B flash device is in a socket so it can be removed for programming. Note that some flash programmers can program a flash device on-board, and some require that the flash device be socketed and removable. Pick the right programmer for your particular hardware situation. A SOP8-207 socket module (Model: 127-SOP8-207) will be used to program the flash.

1. Make sure that you have installed the DediProg Software per the installation instructions and that you have installed the USB driver correctly.
2. Take the SST 25VF016B flash chip out of the SYS9400 platform, and place the SST 25VF016B in the first socket. Make sure that pin 1 is aligned with pin 1 of the socket.
3. Plug the USB connector into the development machine.
4. Start the DediProg Engineering program. The application will automatically detect the flash device.

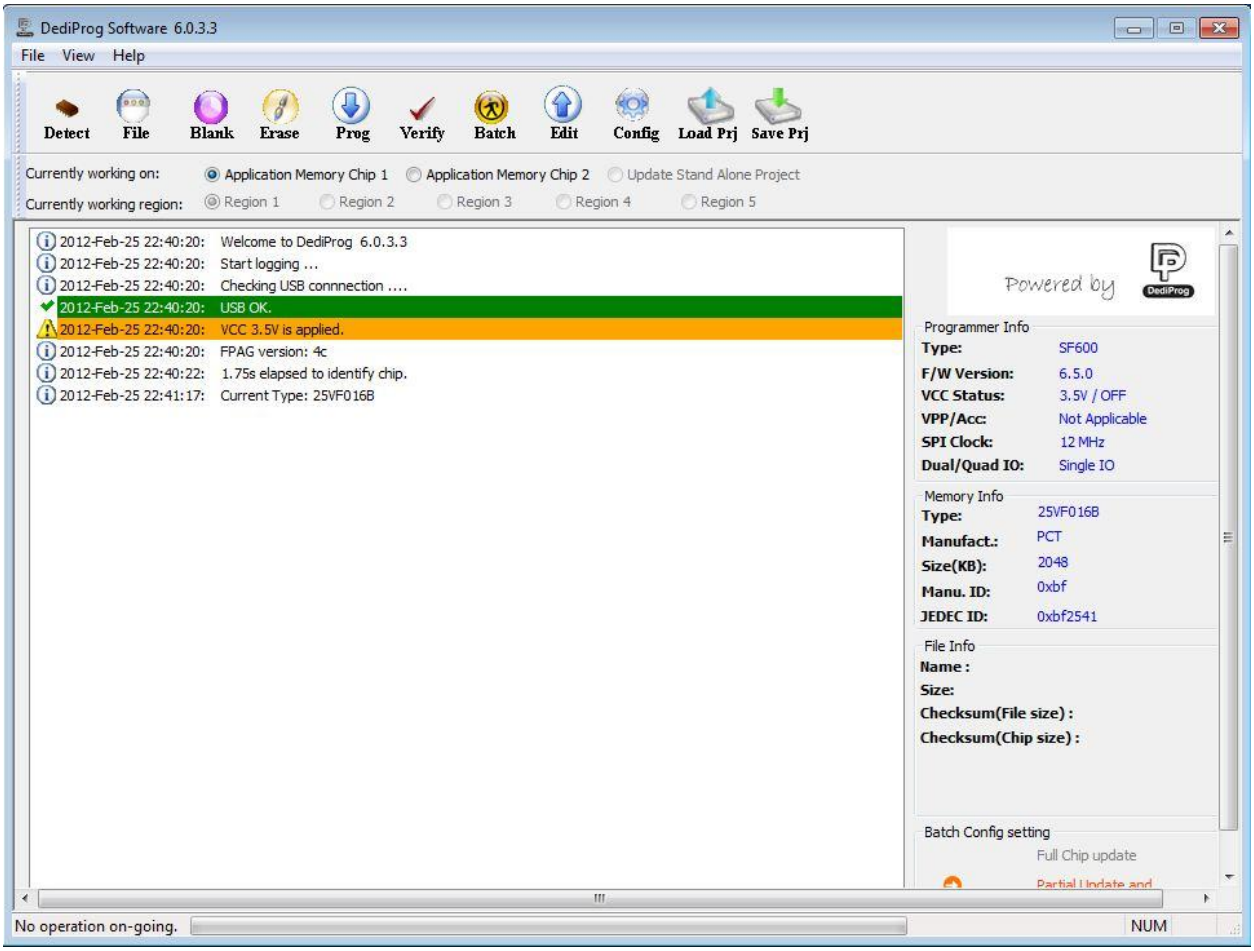



**Figure 5.1 DediProg Application**

5. The next step is to configure the location within the flash to program the device. Since the image is 1MB, the starting location needs to be set for 0x100000. Click on the Configure button.
6. The program provides flexibility in how you program the device. You can either run operations like erase, verify, and program independently or you can run a batch process, which does all the operations to program the flash. The first settings that appear are for the batch process. Select the **Update memory only on sector locations with content difference** radio button. Select the **Update start from address (hex)** radio button and set the value to 100000.

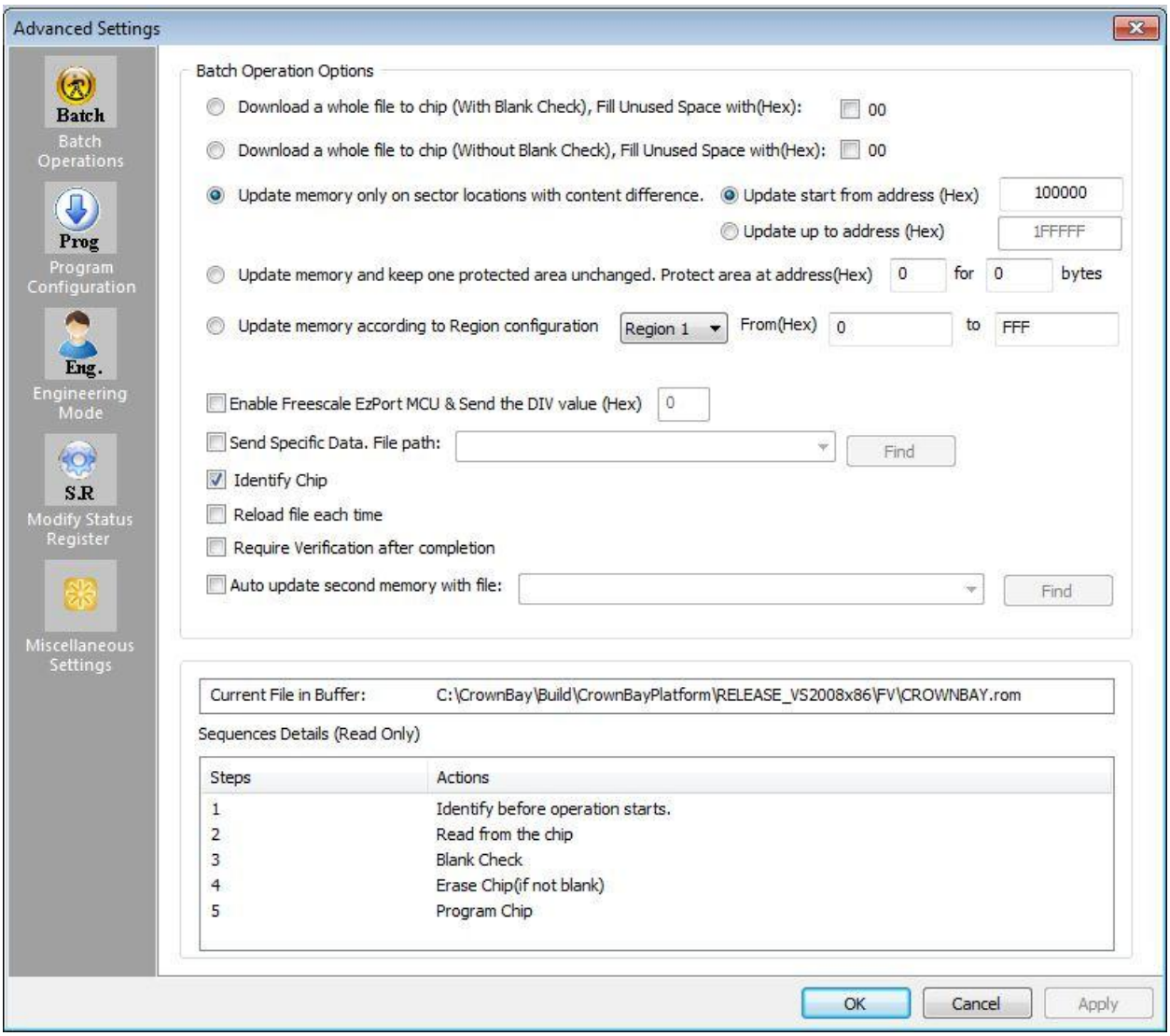
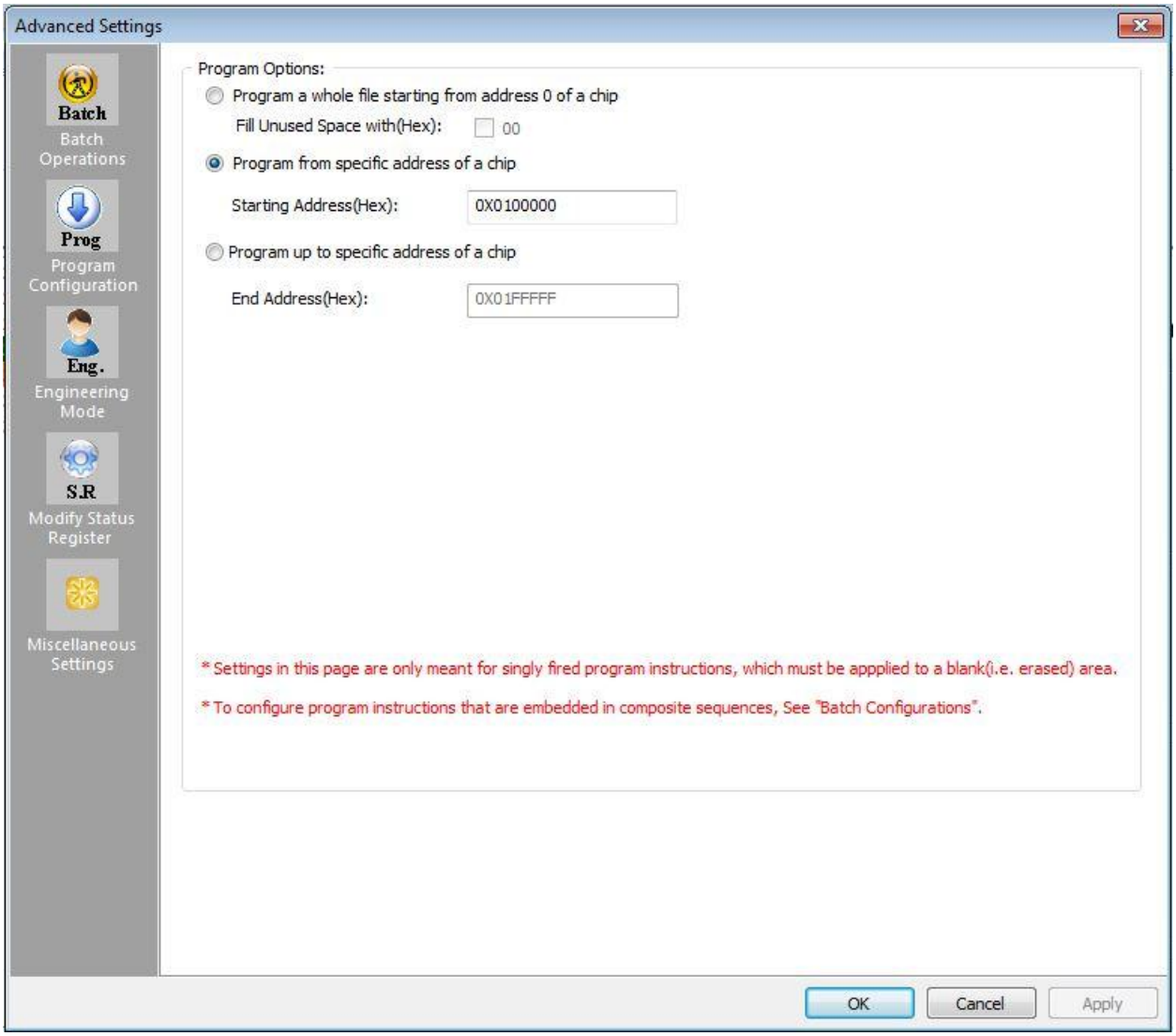



**Figure 5.2 Set the Batch Process starting address**

7. If you prefer to perform the individual operations, click on the Program Configuration button.
8. Select **Program from specific address of a chip – Starting Address (Hex**) and set the value to 0x0100000.




9. Click Ok.
10. Click on the File button, and open the CROWNBAY.rom file.

***Warning**: The current BLDK documentation calls out the .FD file to program into the flash. This will work, but the custom modification defined in the BLDK GUI will not be present. The customization settings only get applied when you patch the firmware. You must use the final firmware .ROM file.*

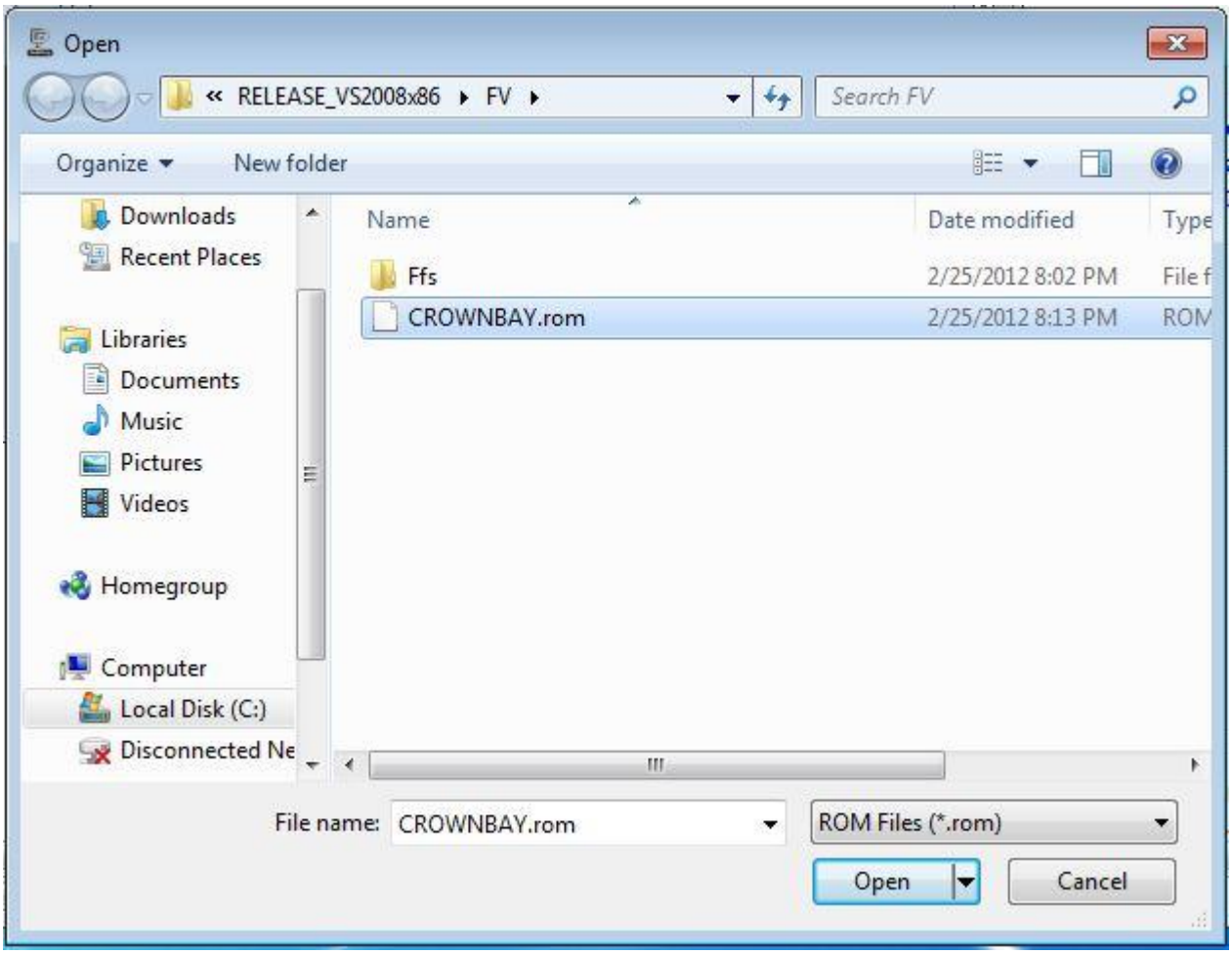
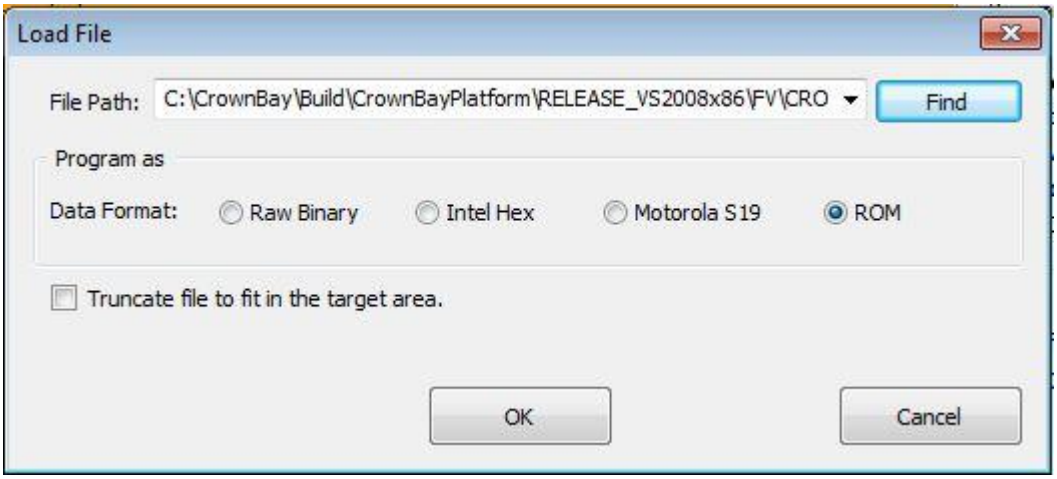
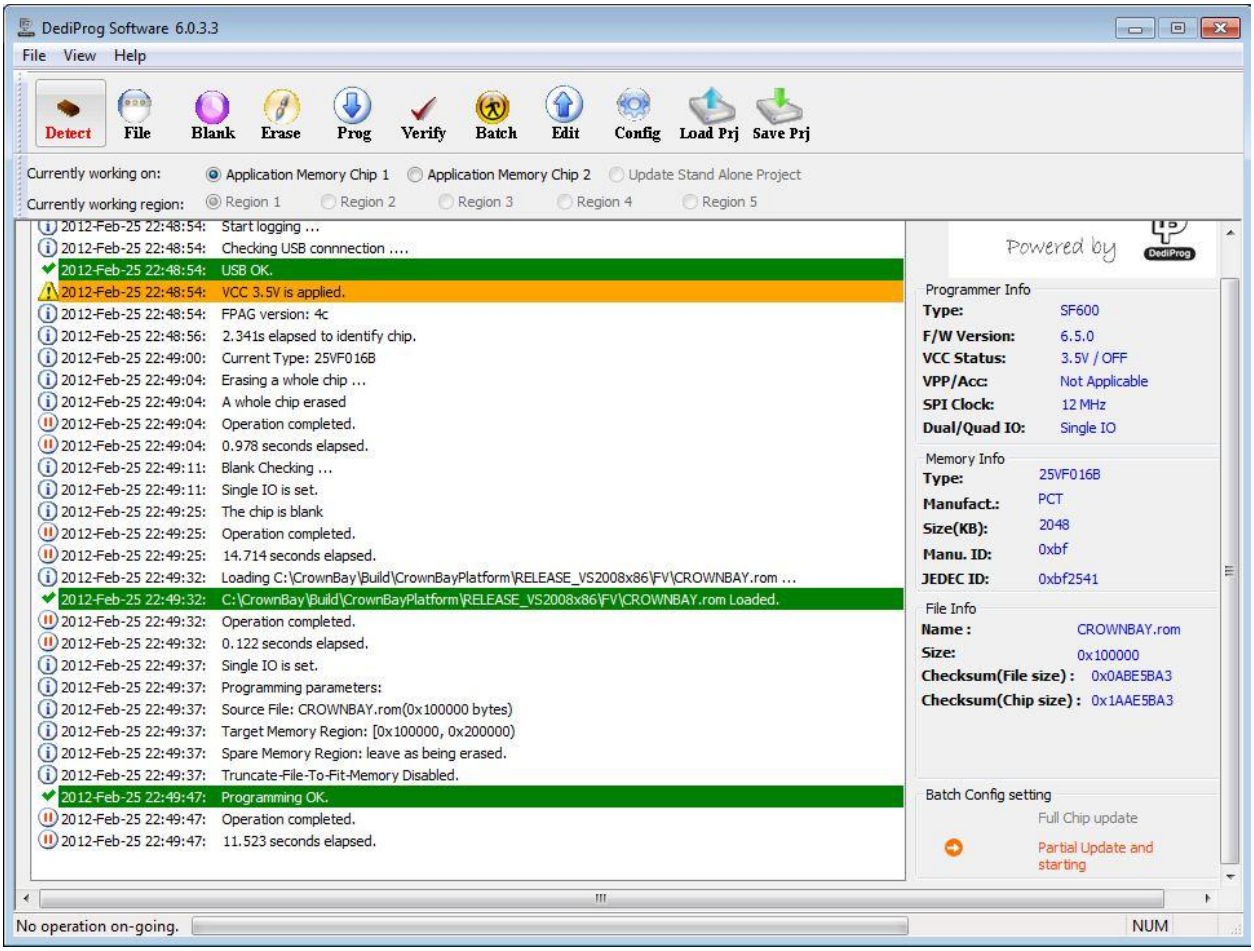



**Figure 5.3 Open the ROM file to be programmed into the flash**

11. The program will set the Data Format type. Click OK to continue.




12. Click on the Batch button and the programmer will begin the process to check the flash, erase if necessary, and then program the firmware.




13. Remove the flash device from the socket, and place it back into the SYS9400 socket. Make sure that pin 1 is aligned with pin 1 of the socket.
14. Connect the NULL Modem Cable between the SYS9400 Reference Platform and the host PC.
15. Make sure no other drives are connected and power on the target.
16. In the terminal program, you will see the system boot to the UEFI Shell.

## *5.2 UEFI Shell + SpiUpdate*

These are the steps to update the firmware using SpiUpdate:

1. Connect the NULL Modem Cable between the SYS9400 Reference Platform and the host PC.
2. Format the USB flash disk with FAT or FAT32 file system.
3. The C:\CrownBay\CrownBayPlatformPkg\Application\SpiUpdate folder has the SpiUpdate.efi application. Copy the SpiUpdate.efi application to the USB flash disk.
4. The C:\CrownBay\Build\CrownBayPlatform\RELEASE_VS2008x86\FV folder has the CROWNBAY.ROM file. Copy the file to the USB flash disk.
5. Eject the USB flash disk.
6. Connect the USB flash disk to the target platform.
7. Start the terminal application program.
8. Make sure no other drives are connected and power on the target.
9. In the terminal program, you will see the system boot to the UEFI Shell.

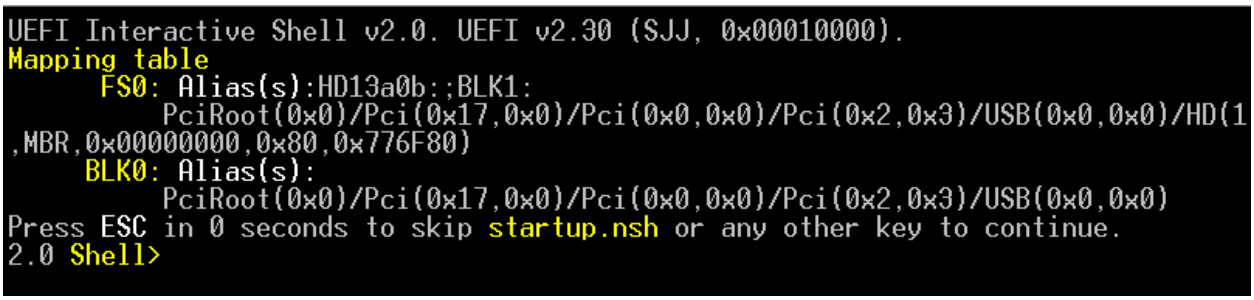



**Figure 5.4 Terminal output from UEFI Shell**

10. The above picture shows the mapping table for the block devices. In this case, UEFI sees the flash disk.
11. In the terminal program, type **fs0:** and hit enter.
12. You can type ls or dir at the prompt to get a directory listing of the flash.

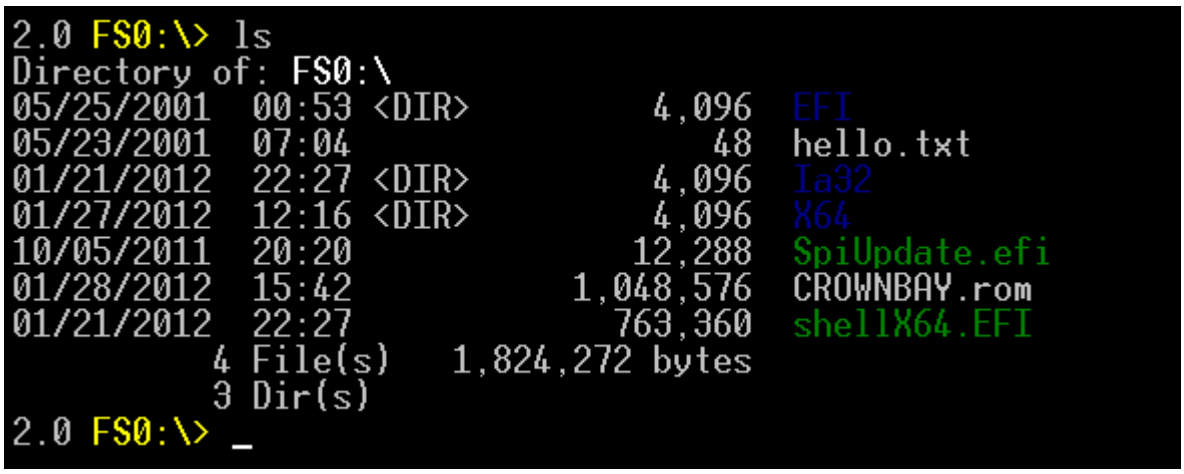



**Figure 5.5 Listing files in the flash drive**

13. To update the flash enter the following at the prompt and hit enter:

    Spiupdate.efi CROWNBAY.rom

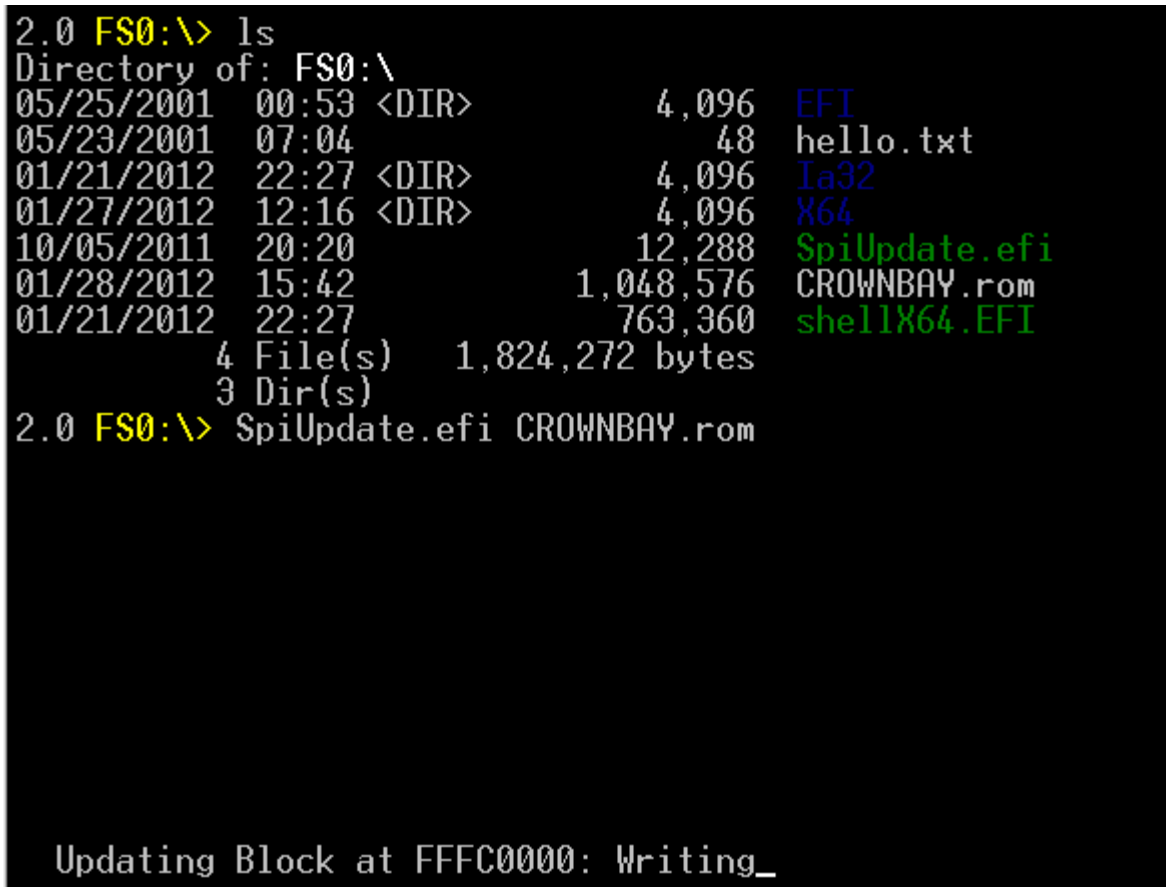



**Figure 5.6 Updating the firmware**

14. The BIOS flash will be updated and the system will then reboot.
15. You will still see the sign-on banner, but the UEFI shell will not enumerate the device mappings.

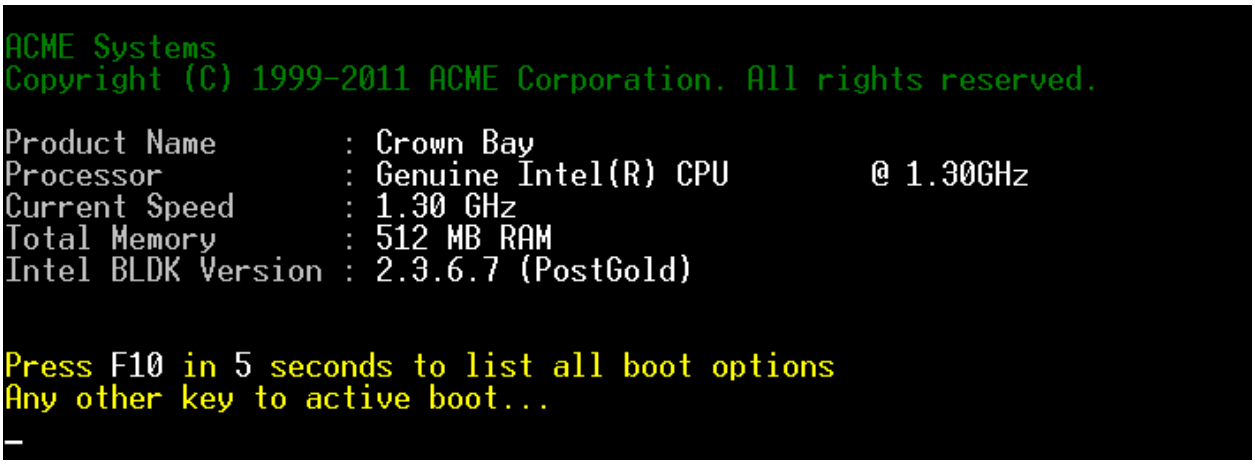



**Figure 5.7 Sign-on banner shows changes**

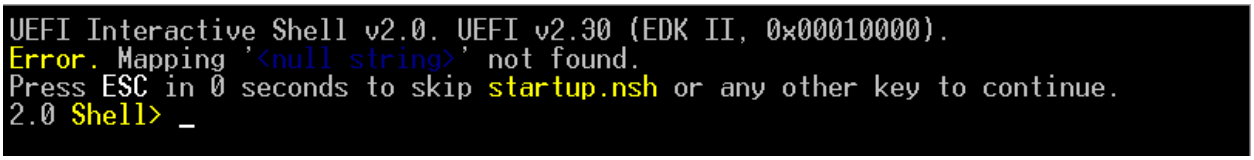



**Figure 5.8 Boot to shell shows mapping not found**

16. Power off the target
17. Remove the USB flash disk

## *5.3 Deployment with Fast Boot Enabled*

To make the system book faster, we can disable the sign-on banner. Since Fast boot is enabled there is an extra step to do the flash programming.

1. Open the BLDK application.
2. Open the BLDK1 project.
3. Go to the Boot options in the configuration tree.
4. Change the Silent Mode to **Enabled**.
5. Save the configuration.
6. Close the project.
7. From the Build menu select Create Final Firmware Image.
8. Select CROWNBAY.FD as the binary file, and tc.absf for the As built file.
9. Copy the CROWNBAY.ROM file to the USB flash disk, and overwrite the existing file.
10. Eject the USB flash disk and plug it into the target.
11. Boot the target system.
12. If you try to run the map command or access the USB flash disk, you will not get very far since the shell didn't enumerate the devices.
13. Type Exit and hit enter. The terminal will show some boot options.


**Figure 5.9 Boot Options**

14. Type the 1 key to enter back into the EFI Internal Shell. This time the devices are enumerated and the shell displays the mapping.

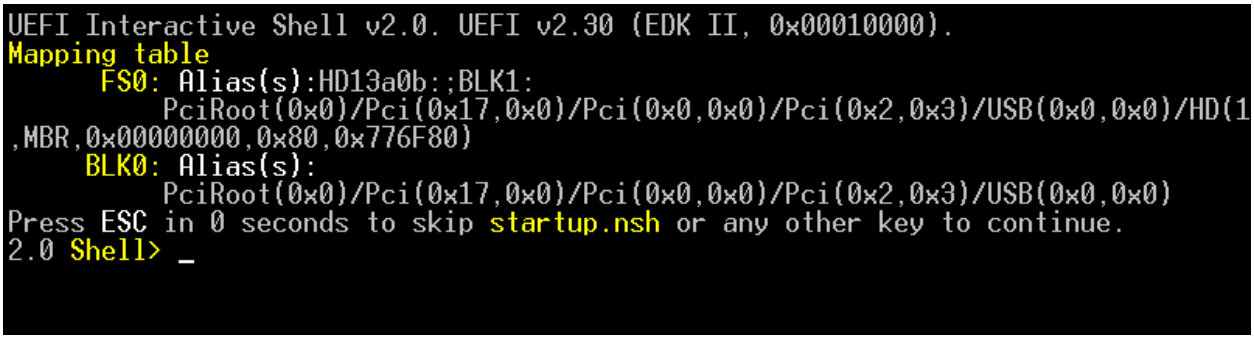



**Figure 5.10 Devices Enumerated and Mapped**

15. Access the USB flash disk, and run the SpiUpdate program again to update the flash with the new changes. The system will reboot and the sign-on banner will not appear. If you reboot the system, you should notice the system booting faster than before.

## *5.4 Booting to the UEFI Shell*

As previously discussed, the BLDK can boot the UEFI shell from a boot disk. The BLDK shell is slightly different than the shell that is part of the UDK2010. The UDK2010 shell includes two editor programs.

1. Change the firmware boot options so the hard drive boots first, USB disk second, and the Internal Shell third.
2. Create the new firmware and update the board. Shut down the target and remove the flash drive when completed.
3. Download the UDK2010 to your development PC, and extract the ZIP file contents.
4. On the USB flash disk, create a directory called EFI.
5. In the EFI folder, create a folder called BOOT.
6. Copy the 32-bit Shell_full.efi found under the UDK path (\UDK2010.SR1.Complete.MyWorkSpace\UDK2010.SR1.MyWorkSpace\MyWorkSpace\EdkShellBinPkg\FullShell\Ia32) to the BOOT folder on the USB flash disk.
7. Rename the Shell_full.efi to BOOTIA32.efi.

***Note**: If the system was 64-bit, the 64-bit Shell_full.efi would be renamed BOOTX64.efi.*

8. Eject the USB flash disk.
9. Put the USB flash disk in the target and power on the target. The system should boot to the shell on the USB flash disk. If you type help, you will see edit and hexedit as available command resources.

# 6 Bibliography

The paper draws from a few resources.

## *6.1 Articles*

*Product Brief: Intel® Boot Loader Development Kit*, Intel, 2011

*Intel® Boot Loader Development Kit (Intel® BLDK) Getting Started Guide*, Intel, Version 2.0, October 2011

*Intel® Boot Loader Development Kit (Intel® BLDK) User Guide*, Intel, Version 2.0, December 2011

*BLDK II with Inforce Tunnel Creek Board for Linux Kernel Version 2.6.29-10 and 2.6.29-12 User Guide*, Inforce Computing, Inc., Version 1 October, 2011.

## *6.2 Websites*

Intel Embedded Design Center – http://edc.intel.com