EFI Version 1.10.14.62 Sample Implementation

Product Release Notes

Version 1.0

December 2, 2003

Revision History

Revision	Revision History	Date
1.0	First release.	12/2/03

Contents

Tables

1 Introduction

1.1 Summary

The *EFI Version 1.10.14.62 Sample Implementation* contains the source code and documentation that are required to develop EFI-related software. This EFI-related software includes EFI firmware, the EFI Shell, EFI device drivers, EFI applications, and EFI OS loaders. This release is designed to support the development of EFI drivers that follow the final draft of the *Extensible Firmware Interface Specification*, version 1.10 (hereafter referred to as the "*EFI 1.10 Specification*").

These release notes contain the following information:

- Introduction
- System requirements
- Package contents
- Known limitations for this release
- Installing EFI Version 1.10.14.62
- Building EFI Version 1.10.14.62
- Building EFI Version 1.10.14.62 for Itanium®-based platforms
- Debugging EFI Version 1.10.14.62
- Directory structure for EFI Version 1.10.14.62
- Porting to a new platform
- Using the EFI boot manager
- Driver map
- New features and bug fixes

1.2 System Requirements

The *EFI Version 1.10.14.62 Sample Implementation* requires the following software to be installed on an IA-32-based system.

Software Requirements

- Microsoft* Windows* NT* 4.0, Microsoft Windows 2000, or Microsoft Windows XP operating system
- One of the following versions of Microsoft Visual Studio* or Visual C++*:
  - — Microsoft Visual Studio .NET 2003 or 2002. These versions do not require service packs. It is recommended to use the Visual Studio .NET versions instead of Visual C++ 6.0.
  - — Microsoft Visual C++ 6.0 with the Visual Studio 6.0 Service Pack 5 or later and the Visual C++ Processor Pack Download. These two service packs are available for download from:
    http://msdn.microsoft.com/vstudio/downloads/updates/sp/vs6/sp5/default.aspx
    http://msdn.microsoft.com/vstudio/downloads/tools/ppack/download.aspx
- Microsoft Macro Assembler* Version 6.15

For the Itanium® architecture environment, an Itanium® compiler and linker from the Microsoft Windows Driver Development Kit (DDK) are required.

1.3 Package Contents

The *EFI Version 1.10.14.62 Sample Implementation* contains the following:

- EFI Version 1.10.14.62 core firmware source code.
- EFI Library source code.
- EFI 1.10 sample drivers.
- EFI 1.10 Driver Library.
- EFI 1.10 Shell source code.
- EFI build tools.
- *EFI Version 1.10.14.62 Sample Implementation Release Notes* (this document).
- *EFI Library Specification.*
- *EFI Driver Library Specification.*
- *EFI 1.1 Shell Commands Specification.*
- *EFI 1.1 SCSI Driver Model.*
- *EFI 1.10 Sample Implementation SAL to EFI Handoff State.*
- A text file containing the three-letter ASCII strings for ISO 639-2. ISO 639-2 specifies an association between a language or dialect and a three-character ASCII string.

1.4 New Features

This release of the *EFI Sample Implementation* has the following new features:

- This release is an update to match the final draft of the *EFI 1.10 Specification* plus the *EFI 1.10 Specification Update*, version -001. It also contains a number of bug fixes to many of the modules. There are no new major features in this release.
- This release adds executable binaries from the build points. These binaries are the nonoptimized executables from the build tips and are for evaluation and testing purposes. You can simply copy the executables to the target machine and run them without having to build the source.
- It updates the build files to support the Visual C++ 6.0 and Visual Studio .NET compilers. Table 1-1 lists the shorthands that are used in this document to refer to the different versions of Visual C++ and Visual Studio.

Table 1-1. Shorthands Used in This Document for Visual Studio/Visual C++

Shorthand	Description
VC6	Microsoft Visual Studio C++ 6.0
VC7	Microsoft Visual Studio .NET 2002
VC71	Microsoft Visual Studio .NET 2003
VC	Refers generically to any of the three versions of Visual C++ or Visual Studio listed above

- It adds new binaries for the EFI UNDI driver to support Gigabit Ethernet cards.
- It adds FAT16 and FAT32 hard disk support to EFI 1.10. To boot from a hard disk, a FAT16 or FAT32 partition must exist in the first 1024 cylinders of the hard disk. In either the **Bios32** or **Ia32-Emb** build tips, build the FAT16 or FAT32 boot sector using the following steps:

  ```
  nmake
  nmake fat16 (or “nmake fat32”)
  ```

  Then copy **bin\Efildr16** or **bin\Efildr20** to the root directory of the partition and use **bin\dumpbs.efi** to write the boot sector:

  dumpbs -w -fat16 blk1 bin\BS16.com (**blk1** is a FAT16 partition)

  dumpbs -w -fat32 blk2 bin\BS32.com (**blk2** is a FAT32 partition)

  Activate the FAT16 or FAT32 partition and then you can boot EFI 1.10 from the hard disk.
- To avoid some debug issues, duplicate symbols are removed from this release. Some of them originally were in both the EFI Library and EFI Driver Library. The following four functions in EFI Driver Library have been renamed:
  - **LShiftU64()**
  - **RShiftU64()**
  - **MultU64x32()**
  - **DivU64x32()**

  This renaming may cause backward-compatibility issues but four new macros that have the same name of the original function are introduced in the EFI Driver Library to avoid such a problem.

1.5 Known Limitations in This Release

The following are the known limitations in this release of the *EFI Sample Implementation:*

- The EFI Shell was changed to store many of its environment variables as volatile instead of nonvolatile variables. As a result, if the **Bios32** or **Ia-32Emb** builds are booted on a system that was used with a previous release of the *EFI Sample Implementation*, there may be some extra error messages when the EFI Shell is booted. To eliminate these messages, use the **alias** Shell command to remove all the aliased commands from nonvolatile storage. The next boot should succeed with no error messages.
- The default console selection is a very simple algorithm that may not be suitable for all platforms. This area can be customized as required for specific platforms.
- A few EFI Runtime Services are not fully implemented. These services include the following:
  - **GetWakeupTime()**: This service is platform specific. As a result, a functional sample implementation cannot be provided.
  - **SetWakeupTime()**: This service is platform specific. As a result, a functional sample implementation cannot be provided.

- The NT emulation environment does not support all the EFI Shell commands and all the **EFI_FILE_SYSTEM_PROTOCOL** interfaces when using the emulated file system. The following Shell commands are not functional:
  - **cd**
  - **rmdir**
  - **mkdir**
- The **Bios32** environment has the SNP32_64, BIOSSNP16, and UNDI drivers built in. The UNDI driver has precedence over the BIOSSNP16 driver, so if you wish to change this behavior, the *Version* fields of the Driver Binding Protocols for these drivers must be modified. Higher *Version* field values mean the driver has higher precedence.
- The **Ia-32Emb** environment has the SNP32_64 and UNDI drivers built in. These drivers can be removed to save space by updating the **makefile** and **init.c** for the **Ia-32Emb** environment.
- The **Sal64** environment has the SNP32_64, BIOSSNP16, and UNDI drivers built in. The SNP32_64 driver has precedence over the BIOSSNP16 driver, so if you wish to change this behavior, the *Version* fields of the Driver Binding Protocols for these drivers must be modified. Higher *Version* field values mean the driver has higher precedence.
- The **Nt32** environment does not have any networking support.
- The **DebugPort** driver (**EFI1.1\Edk\Drivers\Debugport**) produces its services by layering on an instance of the Serial I/O Protocol. The **DebugPort** services are an abstraction of the transport between an off-target debugger and a debugger-specific on-target agent. Because the **DebugPort** services are called in interrupt context by the on-target debugger agent, all code that executes as a result of calling the Serial I/O Protocol interfaces **must** be safe to call from interrupt context. It must be re-entrant, must not modify TPL, assert locks, or cause any internal EFI state to change. This reference implementation has been shown to have these properties. However, if this code is integrated into other environments where these conditions are not met, the results will be unpredictable. If the Serial I/O stack is not safe to use, then the **DebugPort** driver must be rewritten to eliminate the hazards.
- For **Bios32** and **Ia-32Emb**, variables are actually stored in the hard disk (**EFI\BootStr**) and so too the boot options. When changes are made to the system (for example, adding one more hard disk or removing a hard disk), the boot options cannot reflect these changes because the EFI 1.10 sample code only uses the data in that file and does not refresh hardware changes.

2 Installing, Building, and Debugging the EFI Sample Implementation

2.1 Installing the EFI Sample Implementation

To install the *EFI Version 1.10.14.62 Sample Implementation* with Microsoft Visual Studio C++ 6.0, perform the following steps:

1. Install Microsoft Visual C++ 6.0.
2. Install Microsoft Macro Assembler 6.11.
3. Install Visual Studio 6.0 Service Pack 5 or later, which you can get from: http://msdn.microsoft.com/vstudio/downloads/updates/sp/vs6/sp5/default.aspx
4. Install Visual C++ Processor Pack Download, which you can get from: http://msdn.microsoft.com/vstudio/downloads/tools/ppack/download.aspx
5. Copy **`ML.EXE`** from the **`BIN`** directory where Visual C++ is installed into the **`BIN`** directory where Macro Assembler 6.11 is installed. This step is typically a copy from **`C:\Program Files\Microsoft Visual Studio\VC98\Bin\ML.EXE`** to the **`C:\MASM611\BIN`** directory. However, paths may differ depending on your installation.
6. Install the Itanium compiler and linker (for development on the Itanium® processor only).
7. Install the EFI Version 1.10.14.62 Sample Implementation.
8. Copy Macro Assembler 6.11 from **`c:\MASM611\BIN`** to **`EFI1.1\Tools\Ia32\MASM6.11\BIN`**.
9. Copy the Itanium compiler and linker from **`\WINDDK\3790\bin\WIN64\x86`** to **`EFI1.1\Tools\Itanium\Win64`**. If you want to use an Intel® Itanium compiler, you should copy it into **`EFI1.1\Tools\Itanium\Intel64`**.
10. If you are building EFI Byte Code (EBC) EFI 1.10 drivers, you will need the Intel® C Compiler for EFI Byte Code. This compiler is available from http://developer.intel.com/software/products/compilers/efibc/.
11. Use the **`cd`** command to build the directory of interest (e.g., **`cd \EFI1.1\build\bios32`**), and then type **`build VC6`** and then **`nmake`**.

To install the *EFI Version 1.10.14.62 Sample implementation* with Microsoft Visual Studio .NET 2003, perform the following steps:

1. Install Microsoft Visual Studio .NET.
2. Install Microsoft Macro Assembler 6.11.
3. Copy **`ML.EXE`** from the **`BIN`** directory where Visual Studio is installed into the **`BIN`** directory where Macro Assembler 6.11 is installed. This step is typically a copy from **`C:\Program Files\Microsoft Visual Studio .NET 2003\VC7\Bin\ML.EXE`** to the **`C:\MASM611\BIN`** directory. However, paths may differ depending on your installation.
4. Install the Itanium compiler and linker (for development on the Itanium processor only).
5. Install the EFI Version 1.10.14.62 Sample Implementation.

6. Copy Macro Assembler 6.11 from **C:\MASM611\BIN** to **\EFI1.1\Tools\Ia32\MASM611\BIN**.
7. Copy the Itanium compiler and linker from **\WINDDK\3790\bin\WIN64\x86** to **\EFI1.1\Tools\Itanium\Win64**. If you want to use an Intel Itanium compiler, you should copy it into **EFI1.1\Tools\Itanium\Intel64**.
8. If you are building EBC EFI 1.10 drivers, you will need the Intel C Compiler for EFI Byte Code. This compiler is available from http://developer.intel.com/software/products/compilers/efibc/
9. Use the **cd** command to build the directory of interest (e.g., **cd \EFI1.1\build\bios32**), then type **build VC71** and then **nmake**.

2.2 Building EFI Version 1.10.14.62

There are six major build points (tips) in the tree, which are listed in Table 2-1.

Table 2-1. Build Points (Tips) in EFI 1.10.14.62 Source Tree

Build Tip	Description
Efi1.1\Build\Nt32	The NT emulation environment.
Efi1.1\Build\Bios32	EFI for an IA-32 system that uses standard BIOS calls for I/O.
Efi1.1\Build\Ia-32emb	EFI for an IA-32 system that does **not** use BIOS calls for I/O and uses native EFI 1.10 drivers.
Efi1.1\Build\Sal64	EFI for an Itanium-based system.
Efi1.1\Build\Ia32Drivers	Used to build EFI 1.10 drivers for IA-32 systems.
Efi1.1\Build\IpfDrivers	Used to build EFI 1.10 drivers for Itanium-based systems.
Efi1.1\Build\EbcDrivers	Used to build EFI Byte Code (EBC) driver executables.

There is a **build.cmd** file in each directory tip. This file has the global environment variables that must match your tree and your tools.

Table 2-2 describes how the build script determines the Visual C++ or Visual Studio version to build the source tree.

Table 2-2. Determining the Visual Studio/Visual C++ Version

Build Script	Description
Build VC6	Use Visual C++ 6.0 to build the source tree.
Build VC7	Use Visual Studio .NET 2002 (VC7) to build the source tree.
Build VC71	Use Visual Studio .NET 2002 (VC71) to build the source tree.
Build Auto	The build script will try to detect the Visual C++/Visual Studio version itself.
Build	The build script uses the environment variable **%EFI_VCVER_DEFAULT%** to determine the version of VC for building. The value of **%EFI_VCVER_DEFAULT%** can be **VC6**, **VC7**, **VC71**, or **AUTO**.

When you run **build.cmd**, the top of the screen will then include the set of **nmake** commands that are available in that build environment. The set of **nmake** commands varies from build tip to build tip. Table 2-3 lists the common **nmake** commands.

Table 2-3. Common nmake Commands

Command	Description
nmake	Incrementally compiles and links the EFI sources.
nmake clean	Removes all **.OBJ**, **.LIB**, **.EFI**, and **.EXE** files from the build tree.
nmake bsc	Creates the browse information file.
nmake floppy	Copies **EFILDR** to the floppy disk (**Bios32** and **Ia-32Emb** only).
nmake createfloppy	Creates a bootable EFI floppy disk (**Bios32** and **Ia-32Emb** only).
nmake floppytools	Copies the EFI Shell to a floppy disk (**Bios32** and **Ia-32Emb** only).
nmake run	Executes EFI (**Nt32** only).
nmake runloop	Executes EFI with simulated resets (**Nt32** only).

When **nmake** is executed, the output files are placed in the following two directories below the build tip:

- **bin**
- **output**

The **output** directory contains a copy of the EFI tree that is automatically generated. It contains all the **.OBJ**, **.LIB**, and other intermediate files that are created during a build. The **bin** directory contains all the file output files including all of the EFI images with an **.EFI** extension. When **nmake clean** is executed, it removes the **bin** and **output** directories below the current build tip.

There is also a **clean.cmd** file in the **\Efi1.1\Build** directory that cleans the entire *EFI 1.10 Sample Implementation*. This command removes all the **.OBJ**, **.LIB**, **.EFI**, and **.EXE** files from all of the build tips.

Table 2-4 describes the files related to the building process.

Table 2-4. Build-Related Files

File	Description
VCVAR32.BAT	A command script generated by VC setup, which would set all the variables needed by the VC compiler for you. It is located in the **Bin** subdirectory under the VC installation directory. Every version of VC provides its own **VCVAR32.BAT** file. Thus, if there were multiple versions of Visual C++ or Visual Studio residing on your disk, you have to choose the correct version of it before you could use the intended VC compiler.

continued

Table 2-4. Build-Related Files (continued)

File	Description
Build.cmd	There is one **build.cmd** file under each build tip directory. Its intention is to set up the compiling environment for a specific tip. It could call **VCVARS32.BAT** for you, as long as the appropriate arguments were given.
Build-common.cmd	Sets **EFI_SOURCE** automatically. It would be called by **Build-common-ia32.cmd** and **Build-common-ia64.cmd**.
Build-common-ia32.cmd	Does all the things common to all **Ia32** build tips. It assumes VC was installed under the **%ProgramFiles%** directory. If it was not installed there, you have to edit this file manually.
Build-common-ia64.cmd	Currently it only calls **Build-common.cmd** and does nothing else.

Table 2-5 describes the environment variables related to the building process.

Table 2-5. Build-Related Environment Variables

Variable	Description	Example
ProgramFiles	A variable defined by the system. It is the default location where your applications will be installed.	C:\Program Files
EFI_SOURCE	The root directory of the EFI 1.10 source tree. It would be set automatically by **Build-common.cmd**; you do not have to set it manually.	C:\Projecs\EFI1.1
EFI_VCVER_DEFAULT	The default argument that you want to provide to the **build.cmd** script to indicate which version of VC you prefer. Possible values including **AUTO**, **VC6**, **VC7**, and **VC71**.	VC7
MSVCDIR	Set by **VCVARS32.BAT** and should be pointed to the folder where VC was installed.	C:\Program Files\Microsoft Visual Studio .NET 2003\VC7
EFI_VCVER	Set by **Build.cmd**, which is the numeric representation of your VC version.	7
EFI_MASMPATH	Macro Assembler path	C:\Projects\EFI1.1\MASM611

You may need to modify **build.cmd** at each build tip to match your system. There are a few environment variables that must be set correctly to build EFI. A sample of these settings is listed in Table 2-6.

Table 2-6. Sample Environment Variable Settings for Building EFI

Setting	Description
EFI_DEBUG=YES	**EFI_DEBUG** is set to **YES** to enable all debug messages and asserts and set to **NO** to disable all debug messages and asserts. Setting **EFI_DEBUG** to **YES** will increase the size of the EFI core and EFI drivers because the images must carry a print library. By setting **EFI_DEBUG** to **NO**, the EFI core and EFI drivers may be reduced in size.
EFI_DEBUG_CLEAR_MEMORY=YES	**EFI_DEBUG_CLEAR_MEMORY** is set to **YES** to enable setting buffers to a fixed value on EFI Boot Service calls of **FreePages()**, **AllocatePool()**, and **FreePool()**. This setting is intended to be a fine-grained control such that debug messages will be available without the overhead of frequent buffer clears. Set this variable to **NO** to disable the feature.
EFI_BOOTSHELL=YES	**EFI_BOOTSHELL** is set to **YES** to build the EFI Shell and all of the EFI Shell commands into the firmware. When the EFI boots to the EFI boot manager, one of the default boot options will be the EFI Shell. If this environment variable is set to **NO**, then the firmware will not include the EFI Shell. Instead, an output file called **SHELL.EFI** is placed in the **BIN** directory. This file can be executed from the EFI boot manager to boot the EFI Shell from media such as floppy, hard disk, or network.

2.2.1 Building EFI Version 1.10.14.62 for Itanium®-Based Platforms

In addition to the environment variables in the **build.cmd** file, there are additional settings in the **master.env** file. These settings include the **IA64_TOOLS** make variable and the compiler and linker settings.

There are four sets of compiler and linker settings in the **master.env** file. Only one set should be uncommented at a time. The settings with optimizations enabled are available to help reduce the size of EFI 1.10 drivers. The four sets include the following:

- Intel Itanium Compiler 5.01 flags with optimizations off.
- Intel Itanium Compiler 5.01 flags with optimizations on.
- Flags for the Microsoft compiler for Itanium processors (Beta1 or Beta3; available in the Microsoft Windows Server* 2003 DDK, with optimizations off:

```
C_BUILD_FLAGS=/nologo /X /Zi /Zl /Od /W3 /WX /QIPF_fr32
/DNO_INTERFACE_DECL
L_BUILD_FLAGS=/nolog  /SUBSYSTEM:NATIVE /NODEFAULTLIB
/MACHINE:IA64
```

- Flags for the Microsoft compiler for Itanium processors (Beta1 or Beta3; available in the Microsoft Windows Server 2003 DDK), with optimizations on:

```
C_BUILD_FLAGS=/nologo /X /Zl /O1 /Gl /W3 /WX /QIPF_fr32
/DNO_INTERFACE_DECL
L_BUILD_FLAGS=/nolog  /OPT:REF  /LTCG /SUBSYSTEM:NATIVE
/NODEFAULTLIB /MACHINE:IA64
```

If you are using the Microsoft Windows .NET DDK Beta 3, then the **master.env** file needs to be modified to build EFI components with that tool chain:

```
IA64_TOOLS=$(EFI_SOURCE)\Tools\Itanium
CC=CL
LINK=LINK
LIB=LIB
ASM=IAS
```

To set up the build environment and build the **Sal64** build tip using the Microsoft Windows .NET DDK Beta 3, perform the following steps:

```
Start ->
  programs ->
    development kits ->
      Windows DDK 3590 ->
        Build Environments ->
          Windows .NET ->
            Windows .NET Free 64 bit build environment

cd to \efi1.1\build\sal64

build

nmake clean

nmake
```

2.3 Debugging EFI Version 1.10.14.62 with Visual C++/Visual Studio

Source-level debugging is available in the NT Emulation Environment, including source-level debugging of the EFI core, EFI drivers, and any applications and drivers that are loaded using the EFI Shell command **load**. An EFI Shell command **break** was added to force a software breakpoint. When **break** is executed, a pop-up will appear that gives you the option of debugging the application. Select **CANCEL** to debug the application. The source code to the EFI Shell command **break** will be shown. From here, you can set the breakpoints in the code that you wish to debug and select **RUN**. You can also add **BREAKPOINT()** macros to the EFI core or **EFI_BREAKPOINT()** macros to EFI drivers to force software breakpoints. The debugger will need the **.dll** files and **.bsc** files that are created in the build. As an additional step, you must also run the **nmake bsc** command in the build directory to make the **.bsc** file for the **Nt32** build tip.

In addition, you can use the following steps to set up a project in Visual C++ 6.0:

1. Create a project (go to **File→New**, and then select the **Project** tab) and click the **Win32 Application** button.
2. Point the project to your **`Efi1.1\Build\Nt32\Bin`** directory, fill in a dummy project name, and click **OK.**
3. On the **Project** pull-down menu, select the following settings options:
   - On the **Debug** tab, the executable for **Debug** must point to: **`Efi1.1\Build\Nt32\Bin\Nt32.exe`**
   - On the **Debug** tab, set the **Working Directory** to: **`Efi1.1\Build\Nt32\Bin`**
   - On the **Browse Info** tab, set the **browse info** file name to: **`Efi1.1\Build\Nt32\Bin\Nt32.bsc`**
4. On the **Build** pull-down menu, click **Start debugging.** You are then at source-level debugging. You can browse the source code from the **Tools→Source Browser** submenu.

Another way of dumping the EFI executable into the Visual Studio debugger is to execute a break (INT 3) from either the EFI application or embed it in the program that is under test:

1. Start the **`Nt32`** program environment using the command **`cd EFI1.1\build\Nt32\Bin`** and then execute **`nt32.exe`**.
2. Select the Shell.
3. Type **`BREAK`**.

At this point, the system should open a window indicating if you want to enter the debugger. Click **Step out** on the **Debug** tab to exit the break instruction and go back to the calling code.

3 EFI Version 1.10.14.62 Directory Structure

The EFI Version 1.10.14.62 tree has a basic directory structure. All IA-32 code or code specific to the Itanium processor family resides in **Ia32** or **Ipf** subdirectories. If code specific to IA-32 or Itanium architecture is added to a directory, it will force the creation of **Ia32** and **Ipf** subdirectories. Only directories that have nonportable code have **Ia32** and **Ipf** subdirectories.

There are two main components in the EFI 1.10.14.62 directory structure:

- Directories and files that are required for core EFI firmware development
- Files that are required for writing EFI 1.10–compliant drivers

The core EFI firmware will make use of the EFI 1.10–compliant drivers to build a complete firmware stack. However, developers of EFI 1.10 drivers will only need to use a subset of the files and directories that are present in the *EFI Version 1.10.14.62 Sample Implementation.*

3.1 Top Directory Level – Efi1.1

The top directory level of EFI Version 1.10.14.62 is as follows. Table 3-1 describes each subdirectory.

```
Efi1.1\
  Binary\
  Build\
  CoreFw\
  Edk\
  Inc\
  Lib\
  Notes\
  Shell\
```

Table 3-1. Contents of Top Efi1.1 Directory

Directory Name	Description
Binary	Executable binaries from the build points. These binaries are the nonoptimized executables from the build tips and are for evaluation and testing purposes. You can simply copy the executables to the target machine and run them without having to build the source.
Build	Build tips as outlined in section 2.2 (Building EFI Version 1.10.14.62).
CoreFw	The core EFI firmware. This directory includes the core EFI code that follows the *EFI Specification* and the platform-specific code that must be ported for each new platform type.

continued

Table 3-1. Contents of Top Efi1.1 Directory (continued)

Directory Name	Description
Edk	The EFI Driver Toolkit directory. This directory contains all the source code, include files, and libraries required to develop EFI 1.10 drivers that are compliant with the EFI Driver Model in the *EFI 1.10 Specification.* All new EFI driver development should be done in this directory. See section 3.2.6 for details on the files and directories below **Edk**.
Inc	Common include files for the core EFI firmware. EFI drivers that follow the EFI Driver Model must not use this include directory. There is a separate include directory under the **Edk** directory for EFI 1.10–compliant drivers.
Lib	Common library routines that can be used in the core EFI firmware, the EFI Shell, EFI applications, and EFI OS loaders. EFI 1.10–compliant drivers that follow the EFI Driver Model must not use this library. There is a separate EFI Driver Library under the **Edk** directory for EFI 1.10–compliant drivers.
Notes	Documentation for the *EFI Version 1.10.14.62 Sample Implementation.*
Shell	Source code to the new EFI 1.10 Shell and all the new EFI Shell commands. Please see the *EFI 1.10 Shell Commands Specification* for details on all the EFI 1.10 Shell commands and capabilities.

3.2 Subdirectories

The following sections describe some of the subdirectories in the source tree for the *EFI Version 1.10.14.62 Sample Implementation.*

3.2.1 Efi1.1\Binary

The following is the directory structure of the **Efi1.1\Binary** subdirectory:

```
Efi1.1\
  Binary\
    Bios32\
      Bin\
      BIOS32image\
      ShellBios32\
    EBCDriver\
      Bin\
    IA-32EMB\
      Bin\
      IA-32EMBimage\
      ShellIA32emb\
    IA32Drivers\
      Bin\
    IPFDrivers
      Bin\
    NT32\
      Bin\
```

```
Pro1000Undi\
  IA32\
  Itanium\
  RomImage\
    IA32\
    Itanium\
SAL64\
  Bin\
  ShellItanium\
```

Table 3-2 describes the contents of the **Efi1.1\Binary** subdirectory.

Table 3-2. Contents of Efi1.1\Binary Subdirectory

Subdirectory Name	Description
Bios32\bin	This directory contains all the **.efi** executables that would appear from the **\Efi1.1\build\Bios32\bin** directory if you compiled and linked the **Bios32** build tip. These files are built to run on IA-32 using EFI drivers that call down into the legacy system BIOS to touch hardware. This process will rely on standard PC BIOS calls such as INT 10, INT 13, and INT 16 and expect standard PC-AT* hardware (for example, 8259, PS/2* keyboard and mouse, VGA).
Bios32\BIOS32image	This directory contains the files that are necessary to make a **Bios32** 1.44 MB boot floppy image. Any standard PC system with a floppy should be able to boot off this floppy and come up to the EFI boot manager. The purpose of this floppy is for evaluating, testing, and developing EFI applications, the EFI Shell, and EFI drivers and for customizing the boot manager. This floppy should be the most compatible boot floppy and should operate on any standard PC system. The nonvolatile storage for this version of EFI is stored in a file called **Bootstr.efi**, which is hidden and can only be seen with the EFI Shell command **attrib -r**, because there is no flash access for the boot manager. It will be stored on the first hard drive that has a formatted FAT partition. All of the executable images from the **Bios32\bin** directory should run on the system if it is booted from the **Bios32** boot floppy. This version of EFI will not support the real EFI 1.10 device path notation because the legacy BIOS INT 13 calls do not provide this information. A utility **dskimage.exe** is provided that will put the **Bios32** boot image onto a floppy. Type **dskimage –w EFIboot.img**. (**Note:** This utility assumes that a: is your 1.44 MB floppy drive.) **Dskimage.exe** is located in the **\Efi1.1\build\tools\bin** directory. When you are at the **\Efi1.1\build\Bios32** build tip, you can also transfer the EFI boot image to floppy by typing **nmake createfloppy** and then **nmake floppy**.
Bios32\ShellBios32	This directory contains the EFI Shell that is built for IA-32. When used in conjunction with a boot floppy or a system that has EFI in flash, this directory contains the EFI Shell executable. To add the Shell as an application to a boot floppy, copy **nshell.efi** to the floppy or removable device at **\EFI\Boot\Bootia32.efi**.

continued

Table 3-2. Contents of `Efi1.1\Binary` Subdirectory (continued)

Subdirectory Name	Description
EBCDriver\bin	This directory contains two EFI 1.10 drivers that were compiled with the Intel C Compiler for EFI Byte Code. These `.efi` driver executables are portable between Itanium-based systems and IA-32 EFI systems. As an example, the Cirrus Logic* 5430 UGA EBC driver and the IDE ATAPI EBC driver are provided.
IA-32EMB\bin	This directory contains all the `.efi` executables that would appear from the `\Efi1.1\build\Bios32\IA-32EMB` directory if you compiled and linked the `Ia-32Emb` build tip. These files are built to run on IA-32 using native EFI 1.10 drivers. This implementation uses all the native EFI 1.10 drivers to directly touch hardware (with C code). This build tip will not use any BIOS call such as INT 16 or INT 13 once the PC has given control to the boot image on the floppy. The drivers in this build were designed for standard PC-AT hardware (such as ATAPI IDE, 8259, PS/2 keyboard and mouse, and VGA).Developers can use this boot floppy to develop native EFI 1.10 drivers with their add-in cards. The executables and EFI drivers in this directory are for testing and evaluation purposes.
IA-32EMB\ IA-32EMBimage	This directory contains the files that are necessary to make an `IA-32emb` 1.44MB boot floppy image. Any standard PC system with a floppy should be able to boot off this floppy and come up to the EFI boot manager. The purpose of this floppy is for evaluating, testing, and developing EFI applications, the EFI Shell, and EFI drivers and for customizing the boot manager. This floppy should be the most compatible boot floppy and should operate on any standard PC system. The nonvolatile storage for this version of EFI is stored in a file called `Bootstr.efi`, which is hidden and can only be seen with the EFI Shell command `attrib -r`, because there is no flash access for the boot manager. It will be stored on the first hard drive that has a formatted FAT partition. All of the executable images from the `IA-32emb\bin` directory should run on the system if it is booted from the `IA-32emb` boot floppy or an IA-32 system that supports EFI 1.10 from flash. This version of EFI will support the full device path notation for all media and boot devices. A utility `dskimage.exe` is provided that will put the `Ia-32Emb` boot image onto a floppy. Type `dskimage –w EFIboot.img`. (**Note:** This utility assumes that a: is your 1.44 MB floppy drive.). `Dskimage.exe` is located in the `\Efi1.1\build\tools\bin` directory. When you are at the `\Efi1.1\build\Ia-32Emb` build tip, you can also transfer the EFI boot image to floppy by typing `nmake createfloppy` and then `nmake floppy`.
IA-32EMB\ ShellIA32emb	This directory contains the EFI Shell that is built for `IA-32emb`. When used in conjunction with a boot floppy or a system that has EFI in flash, this directory contains the EFI Shell executable. To add the Shell as an application to a boot floppy, copy `nshell.efi` to the floppy or removable device at `\EFI\Boot\Bootia32.efi`.

continued

Table 3-2. Contents of `Efi1.1\Binary` Subdirectory (continued)

Subdirectory Name	Description
IA32Drivers\bin	This directory contains all the IA-32 `.efi` executable drivers that would be built from the `\EFI1.1\build\IA32Drivers\bin` directory. There are about 39 standard PC drivers that are provided as examples of the EFI 1.10 Driver Model. They cover USB (UHCI), IDE, floppy, PS/2 mouse and keyboard, VGA, SCSI passthru, UNDI for Intel® PRO/100 network adapters, PXE, PCI, FAT, ATAPI passthru, and others. All of these drivers are native EFI 1.10 drivers and expect a system to have the full *EFI 1.10 Specification* implemented. These drivers can be used with the EFI `IA-32emb` boot floppy or a system with the support in flash. These drivers will not work on a system that is booted from the `Bios32` boot floppy. These executables match the source found in the `\EFI1.1\EDK\Drivers` directory.
IPFDrivers\bin	This directory contains all the `.efi` executable drivers for Itanium processors that would be built from the `\EFI1.1\build\IPFDrivers\bin` directory. There are about 39 drivers that are provided as examples of the EFI 1.10 Driver Model and that are compiled for an Itanium-based system. They cover USB (UHCI), IDE, floppy, PS/2 mouse and keyboard, VGA, SCSI passthru, UNDI for Intel PRO/100 network adapters, PXE, PCI, FAT, ATAPI passthru, and others. There is no boot floppy for Itanium-based systems and it is required that the full EFI 1.10 is implemented in flash. These executables match the source found in the `\EFI1.1\EDK\Drivers` directory.
NT32\bin	This directory contains all the IA-32 executables from the `\EFI1.1\build\NT32\bin` directory. This version of EFI is meant to run on top of an IA-32 system running Windows XP, 2000, or NT. When copied to the hard drive of a normal Windows system, the executable `nt32.exe` and binaries should allow you to execute an emulated EFI environment. The purpose of this environment is to allow development and debug of the EFI boot manager, EFI Shell, and EFI applications using the standard EFI protocols and services. Because this environment runs on Windows with Visual Studio present, the full IDE debugger from Visual Studio can be used to debug applications and the boot manager source as long as the `.pdb` and `.dll` debug files are provided. The Shell `break` command (or compiling in an `EFI_Breakpoint()` or the macro `breakpoint()` into the source code) will force a breakpoint to enter the debugger. You cannot develop or debug EFI 1.10 drivers in this environment because the operating system will virtualize all access to the hardware in the system. The nonvolatile RAM for the boot manager and Shell is stored in a file on the disk called `NV1file`. This version of EFI creates a virtual memory ramdrive that is mapped as `FS0:` where all the `.efi` executables from the `\Nt32\bin` directory are copied. The `Nt32` emulation environment version can only map file systems that are FAT.

continued

Table 3-2. Contents of `Efi1.1\Binary` Subdirectory (continued)

Subdirectory Name	Description
Pro1000Undi	This directory contains the binary EFI 1.10 drivers for the Intel® PRO/1000 family of Gigabit network adapters. Two native drivers are provided: one for Itanium processors and one for IA-32. These drivers are for incorporation in OEM systems that need to support booting PXE from LAN on the Intel PRO/1000 family of network adapter cards through EFI. The **`RomImage`** directory provides ROM images that can be flashed onto the Intel PRO/1000 family of network adapter cards for Itanium architecture or IA-32. There is no source provided for these drivers. If you have any questions about the drivers, please contact your Intel ICG support person.
SAL64\bin	This directory contains the binaries for Itanium processors that were created from the **`\EFI1.1\build\sal64\bin`** directory. These binaries are meant to be used on Itanium-based systems that already have EFI. It is important to note the version of EFI 1.10 on which they are released; you can determined the EFI 1.10 version by the looking at the EFI boot manager version or by going to the EFI Shell and typing **`ver`**.
SAL64\ShellItanium	This directory contains the EFI Shell binary that is compiled for Itanium-based systems. The source for this binary is found in the **`\EFI1.1\shell`** directory.

3.2.2 Efi1.1\CoreFw\Fw\

The **`Efi1.1\CoreFw\Fw\`** subdirectory contains the following subdirectories:

```
Efi1.1\
  CoreFw\
    Fw\
      Efi\
      Inc\
      Platform\
```

Sections 3.2.3 through 3.2.5 discuss some of these subdirectories in more detail.

3.2.3 Efi1.1\CoreFw\Fw\Efi

The subdirectory **`Efi1.1\CoreFw\Fw\Efi`** contains the EFI code that implements the *EFI Specification*. This code is portable and should not be modified when a port is made to a new platform. This code is the key code that tracks the specification. This code should not be modified, and any bugs found in this code should be reported to the EFI team.

The following directories in **`Efi1.1\CoreFw\Fw\Efi`** are supported:

```
Efi1.1\
  CoreFw\
    Fw\
      Efi\
        Crc\
        Exec\
        Hand\
        Inc\
        Init\
        Loader\
```

```
Mem\
Misc\
Variable\
```

3.2.4 Efi1.1\CoreFw\Fw\Platform

The following is the directory structure of the **Efi1.1\CoreFw\Fw\Platform** subdirectory:

```
Efi1.1\
  CoreFw\
    Fw\
      Platform\
        BootMgr\
        BuildTip\
        Drivers\
        Inc\
        Lib\
        PlDriver\
```

The code in the **Efi1.1\CoreFw\Fw\Platform** subdirectory is the only code that needs to be modified to port EFI to new platforms. Currently there are four supported platforms:

- **Nt32**
- **Bios32**
- **Ia-32Emb**
- **Sal64**

The entry points to each of these platforms can be found in the **Efi1.1\CoreFw\Fw\Platform\BuildTip** subdirectory. Table 3-3 describes the contents of the **Efi1.1\CoreFw\Fw\Platform\BuildTip** subdirectory, which are as follows:

```
Efi1.1\
  CoreFw\
    Fw\
      Platform\
        BuildTip\
            Nt32\
            Bios32\
            Ia-32Emb\
            Sal64\
```

Table 3-3. Contents of Efi1.1\CoreFw\Fw\Platform\BuildTip Subdirectory

Subdirectory Name	Description
Nt32	This directory contains the platform code for the NT emulator.
Bios32	This platform is a generic port of EFI to a PC. The IA-32 code must load, relocate, and then call **MainEntry()**. The **ConsoleIn**, **ConsoleOut**, and **BlockIo** drivers come from the **Drivers\BiosInt** directory. These drivers use the legacy 16-bit BIOS INT calls to build the basic EFI platform drivers.

continued

Table 3-3. Contents of `Efi1.1\CoreFw\Fw\Platform\BuildTip` Subdirectory (continued)

Subdirectory Name	Description
Ia-32Emb	This platform is a generic port of EFI to a PC. The IA-32 code must load, relocate, and then call **`MainEntry()`**. The **`ConsoleIn`**, **`ConsoleOut`**, and **`BlockIo`** drivers come from drivers in the **`Efi1.1\Edk`** directory. These drivers are native drivers that do not depend on any BIOS services. In fact, this implementation does not use any BIOS calls at all.
Sal64	This platform is a generic port of the reference System Abstraction Layer (SAL). There is code in the reference SAL that loads, relocates, and then calls **`MainEntry()`**.The **`ConsoleIn`**, **`ConsoleOut`**, and **`BlockIo`** drivers come from the **`Drivers\BiosInt`** directory. These drivers use the legacy BIOS INT calls to build the basic EFI platform drivers.

Some common code also exists in the **`Efi1.1\CoreFw\Fw\Platform`** directory, in the subdirectories listed in Table 3-4.

Table 3-4. Common Code in the `Efi1.1\CoreFw\Fw\Platform` Subdirectory

Subdirectory Name	Description
BootMgr	Sample Implementation of the EFI boot manager and EFI Boot Maintenance Manager.
Drivers	This directory contains device drivers that use BIOS services to access console and block I/O devices.
PlDriver	This directory contains all the drivers for system-specific devices. These drivers are typically for motherboard devices.

3.2.5 Efi1.1\CoreFw\Fw\Platform\Drivers\BiosInt

The code in the **`Efi1.1\CoreFw\Fw\Platform\Drivers\BiosInt`** subdirectory contains the device drivers that use BIOS services to access boot devices. Even though these drivers use BIOS services, most have been converted to follow the EFI 1.10 Driver Model. All native drivers that follow the EFI 1.10 Driver Model are in the **`\Efi1.1\Edk`** directory.

The contents of the **`Efi1.1\CoreFw\Fw\Platform\Drivers\BiosInt`** subdirectory are as follows. Table 3-5 describes these subdirectories.

```
Efi1.1\
  CoreFw\
    Fw\
      Platform\
        Drivers\
          BiosInt\
            BiosKeyboard\
            BiosSnp16\
            BiosVga\
            BiosVgaMiniPort\
            Disk\
```

Table 3-5. Contents of the `Efi1.1\CoreFw\Fw\Platform\Drivers\BiosInt` Subdirectory

Subdirectory Name	Description
BiosKeyboard	Driver that uses BIOS INT 0x16 services to produce the Simple Input Protocol for the keyboard. This driver follows the EFI Driver Model, so it can be disconnected and replaced with a native keyboard driver.
BiosSnp16	Driver that uses 16-bit FAR calls into a legacy UNDI to produce the Simple Network Protocol. This driver follows the EFI Driver Model, so it can be disconnected and replaced with a native UNDI driver.
BiosVga	Driver that uses BIOS INT 0x10 services to produce the Simple Text Output Protocol. This driver is typically used with VGA class devices. This driver follows the EFI Driver Model, so it can be disconnected and replaced with a native video driver.
BiosVgaMiniPort	Driver that uses BIOS INT 0x10 services to produce the VGA Mini Port Protocol. This protocol is used by the **`VgaClass`** driver to produce the Simple Text Output Protocol. This driver is typically used with VGA class devices. A system will either include the **`BiosVga`** driver or the combination of the **`BiosVgaMiniPort`** and the **`VgaClass`** driver to produce a console. This driver follows the EFI Driver Model, so it can be disconnected and replaced with a native video driver.
Disk	Driver that uses BIOS INT 0x13 services to produce the Block I/O Protocol for disk devices including floppy devices, hard disks, and CD-ROMs. This driver is the only one that does not follow the EFI Driver Model. Instead, it produces the handles for the disk devices, and those handles can never be stopped. This is done to prevent a native EFI 1.10 driver from managing a disk controller at the same time that a BIOS INT 0x13 driver is also managing the disk controller. To guarantee that there are no resource conflicts, platforms that use this driver must have INT 0x13 services that follow the *BIOS Enhanced Disk Drive Specification,* version 3.0.

3.2.6 Efi1.1\Edk

The EFI Driver Toolkit directory, **`Efi1.1\Edk`**, contains all the components that are required to develop EFI 1.10 drivers. EFI 1.10 drivers must not use include files or libraries from the **`Efi1.1\Inc`** or **`Efi1.1\Lib`** directories. There were naming inconsistencies in the *EFI 1.02 Sample Implementation*, and these inconsistencies have been resolved for EFI 1.10 driver development by placing new versions of the include files, libraries, GUIDs, and protocols in the **`Efi1.1\Edk`** directory. These new versions also have the added benefit of reducing the size of EFI 1.10 driver images.

The contents of the **`Efi1.1\Edk`** directory are as follows:

```
Efi1.1\
  Edk\
    Drivers\
    Guid\
    Protocol\
    Include\
    Lib\
```

Table 3-6 describes these subdirectories.

Table 3-6. Contents of the Efi1.1\Edk Subdirectory

Subdirectory Name	Description
Drivers	This directory contains source code to all the EFI 1.10–compliant drivers.
Guid	This directory contains source code for all the GUIDs defined in the *EFI Specification.* This directory is used to generate a library called **GUID.LIB** that contains all the GUIDs that are defined in this directory. If an EFI 1.10 driver requires a GUID from this directory, the driver needs to link against **GUID.LIB**. The **.h** files in this directory contain the definition of the GUID and the declaration of any data structures that are associated with the GUID.
Protocol	This directory contains source code for all the protocols defined in the *EFI Specification.* This directory is used to generate a library called **PROTOCOL.LIB** that contains all the GUIDs that are associated with the protocol definitions defined in this directory. If an EFI 1.10 driver consumes or produces a protocol from this directory, the driver needs to link against **PROTOCOL.LIB**. The **.h** files in this directory contain the definition of the GUID and the protocol interface data structure.
Include	This directory contains the include files for the definitions from the *EFI Specification* and other industry standard specifications (e.g., PCI, ACPI, and PXE).
Lib	This directory contains a lightweight library that is used for EFI 1.10 drivers. EFI 1.10 drivers should use this library to keep the driver image sizes small. There are three libraries in the directory: • EFI Driver Library • Print Library • String Library The functions in these directories are described in the *EFI 1.10 Driver Library Specification.* The Print Library is typically used only for debug messages by using the **DEBUG()** macro. When **EFI_DEBUG** is set to **NO**, the **DEBUG()** macros are removed and the driver size is reduced because the Print Library is no longer required. EFI 1.10 drivers should not directly use the services of the Print Library unless they absolutely require them.

3.2.7 Efi1.1\Edk\Guid

The following is the list of GUIDs in the **Efi1.1\Edk\Guid** subdirectory that are available to EFI 1.10 drivers.

```
Efi1.1\
  Edk\
    Guid\
      Bmp\
      ConsoleInDevice\
      ConsoleOutDevice\
      DebugImageInfoTable\
      GlobalVariable\
      Gpt\
      HotPlugDevice\
      Pcansi\
      PciOptionRomTable\
```

```
PrimaryConsoleInDevice\
PrimaryConsoleOutDevice\
PrimaryStandardErrorDevice\
SalSystemTable\
SmBios\
StandardErrorDevice\
```

3.2.8 Efi1.1\Edk\Protocol

The following is the list of protocol interfaces in the **Efi1.1\Edk\Protocol** subdirectory that are available to EFI 1.10 drivers. See the *EFI 1.10 Specification* for descriptions of these protocols.

```
Efi1.1\
  Edk\
    Guid\
      Bis\
      BlockIo\
      BusSpecificDriverOverride\
      ComponentName\
      DebugPort\
      DebugSupport\
      Decompress\
      DeviceIo\
      DevicePath\
      DiskIO\
      DriverBinding\
      DriverConfiguration\
      DriverDiagnostics\
      Ebc\
      EfiNetworkInterfaceIdentifier\
      IsaAcpi\
      IsaIo\
      LegacyBoot\
      LoadedImage\
      LoadFile\
      PciIo\
      PciRootBridgeIo\
      PlatformDriverOverride\
      PxeBaseCode\
      PxeBaseCodeCallBack\
      PxeDhcp4\
      PxeDhcp4Callback\
      ScsiIo\
      ScsiPassThru\
      SerialIO\
      SimpleFileSystem\
      SimpleNetwork\
      SimplePointer\
      SimpleTextIn\
      SimpleTextOut\
```

Tcp
UgaDraw
UgaIo
UgaSplash
UnicodeCollation
UsbAtapi
UsbHostController
UsbIo
VgaMiniPort
WinNtIo
WinNtThunk

3.2.9 Efi1.1\Edk\Drivers

Table 3-7 lists the sample drivers in the **Efi1.1\Edk\Drivers** subdirectory that are either service drivers, root bridge drivers, or drivers that follow the EFI Driver Model.

Table 3-7. Sample Drivers in the Efi1.1\Edk\Drivers Subdirectory

Driver Name	Description
AtapiPassThru	A sample driver that produces the EFI 1.10 SCSI Pass Thru Protocol from the *EFI 1.10 Specification* for ATAPI devices on an IDE channel. This protocol allows SCSI command packets to be sent directly to an ATAPI device.
Bis	A sample driver that is an implementation of the Boot Integrity Services Protocol from the *EFI 1.10 Specification.*
CirrusLogic5430	This driver is a sample implementation of the UGA Draw Protocol for the Cirrus Logic 5430 family of PCI video controllers. This driver is usable only in the EFI preboot environment. This sample is intended to show how the UGA Draw Protocol is able to function. The UGA I/O Protocol is not implemented in this sample. A fully compliant EFI UGA driver requires both the UGA Draw Protocol and the UGA I/O Protocol. See Microsoft's documentation on UGA for details on writing a UGA driver that is able to function both in the EFI preboot environment and from the OS runtime.
Console\ConPlatform	This driver controls the platform policy for selecting console devices including the Console Input devices, Console Output devices, and Standard Error devices. The default policy is to use the *ConIn*, *ConOut*, and *StdErr* environment variables to decide which console should be connected when the system is booted. If the *ConIn*, *ConOut*, and *StdErr* environment variables do not exist, then all consoles are connected and available when the system is booted.
Console\ConSplitter	This driver performs multiplexing/demultiplexing between the consoles that were selected by the **ConPlatform** driver. It produces a virtual handle for the primary console input device, the primary console output device, and the primary standard error device. The EFI boot manager connects to these primary consoles if they are present in the system. If a primary console device cannot be found, then the EFI boot manager connects to any console devices it can find.

continued

Table 3-7. Sample Drivers in the `Efi1.1\Edk\Drivers` Subdirectory (continued)

Driver Name	Description
Console\GraphicsConsole	This driver produces the Simple Text Output Protocol on a UGA display device. It uses the services of the UGA Draw Protocol along with a font database to draw characters onto the UGA display device.
Console\Terminal	This driver produces console devices for PC ANSI, VT-100, VT-00 Plus, and VT-UTF8 serial terminals.
DebugPort	A sample driver that implements the Debug Port Protocol from the *EFI 1.10 Specification*.
DebugSupport	A sample driver that implements the Debug Support Protocol from the *EFI 1.10 Specification* for IA-32 and Itanium-based systems.
Decompress	A sample driver that implements the Decompress Protocol from the *EFI 1.10 Specification*. This driver provides the basic services that are required to decompress an image that was compressed using the EfiCompress build tool, or the `EfiCompress` Shell command.
DiskIo	A sample driver that consumes the Block I/O Protocol and produces the Disk I/O Protocol on the same device handle.
Ebc	A sample driver that implements the EBC interpreter, the EBC Protocol, and the Debug Support Protocol from the *EFI 1.10 Specification*.
FileSystem\Fat	A sample driver that consumes the Block I/O Protocol and Disk I/O Protocol and produces the File System Protocol if the disk or partition is formatted with the FAT.
Ide	A sample PCI driver for PCI-based IDE controllers. It consumes the services of the PCI I/O Protocol on the device handle for a PCI IDE controller and produces child handles with the Block I/O Protocol for the IDE and ATAPI disk devices that are present on the IDE channels. This driver is the only sample driver that produces all of the EFI Driver Model–related protocols. These protocols include the Driver Binding Protocol, Component Name Protocol, Driver Configuration Protocol, and Driver Diagnostics Protocol.
IsaBus	A sample driver that consumes the PCI I/O Protocol for a PCI-to-ISA bridge device and the ISA ACPI Protocol to produce child handles for the ISA devices present in a system.
IsaFloppy	A sample driver that consumes the ISA I/O Protocol for a legacy floppy controller and produces a Block I/O Protocol.
IsaSerial	A sample driver that consumes the ISA I/O Protocol for a 16550 UART controller and produces the Serial I/O Protocol.
Partition	A sample driver that consumes the Block I/O Protocol and the Disk I/O Protocol and produces a Block I/O Protocol for sub-block devices that are detected on the media. The sub-block devices that this driver supports are partitions on disk devices that conform to the Master Boot Record (MBR), El Torito, and Guided Partition Table (GPT) partitioning schemes.

continued

Table 3-7. Sample Drivers in the `Efi1.1\Edk\Drivers` Subdirectory (continued)

Driver Name	Description
PcatIsaAcpi	A sample driver that produces the ISA ACPI Protocol. This protocol is a platform-specific protocol that describes the ISA devices that are present in a system. This driver is used with the **`Ia-32Emb`** build tip. It describes a platform with two ISA serial ports, a PS/2 keyboard controller, a PS/2 mouse controller, and a legacy floppy disk controller with two floppy drives.
PcatIsaAcpiBios	A sample driver that produces the ISA ACPI Protocol. This protocol is a platform-specific protocol that describes the ISA devices that are present in a system. This driver is used with the **`Bios32`** and **`Sal64`** build tips. It describes a platform with two ISA serial ports and a PS/2 keyboard controller.
PcatPciRootBridge	A sample driver that produces a PCI Root Bridge I/O Protocol from the *EFI 1.10 Specification* for each PCI root bridge that is detected in the system. This driver assumes a PC-AT-like hardware architecture to perform this detection. If a platform has a different hardware architecture, then a different version of this driver will have to be written for that specific platform.
PciBus	A sample driver that consumes the PCI Root Bridge I/O Protocol and produces child handles for PCI devices behind that PCI root bridge. An instance of the PCI I/O Protocol from the *EFI 1.10 Specification* is placed on each child handle that the PCI bus driver detects.
PciVgaMiniPort	A sample driver that consumes the services of the PCI I/O Protocol for a PCI VGA class device and produces the VGA MiniPort Protocol. The VGA MiniPort Protocol is consumed by the **`VgaClass`** driver to produce the Simple Text Output Protocol for the PCI VGA device.
Ps2Keyboard	A sample driver that consumes the ISA I/O Protocol for an 8042 keyboard controller and produces the Simple Input Protocol.
Ps2Mouse	A sample driver that consumes the ISA I/O Protocol for an 8042 keyboard controller and produces the Simple Pointer Protocol if a PS/2 mouse is detected.
PxeBc	A sample driver that consumes the Simple Network Protocol and produces the PXE Base Code Protocol and Load File Protocol for a network device. The Load File Protocol that is produced by the driver will show up in the EFI boot manager, so the user can boot from a PXE server.
PxeDhcp4	A sample driver that provides the ability to perform DHCP from a standard DHCP server.
ScsiBus	A sample driver that consumes the SCSI Pass Thru Protocol and produces child handles for the SCSI devices on a SCSI channel. An instance of the SCSI I/O Protocol is placed on each child handle that the SCSI bus driver detects.

continued

Table 3-7. Sample Drivers in the `Efi1.1\Edk\Drivers` Subdirectory (continued)

Driver Name	Description
ScsiDisk	A sample driver that consumes the SCSI I/O Protocol and produces the Block I/O Protocol if the device is a SCSI disk device.
SerialMouse	A sample driver that consumes the Serial I/O Protocol and produces the Simple Pointer Protocol from the *EFI 1.10 Specification*. This driver specifically supports Microsoft serial mice. If a Microsoft serial mouse is detected, then the Simple Pointer Protocol is produced. If a Microsoft serial mouse is not detected, then no new protocols are produced and the serial port is restored to its original configuration. The Simple Pointer Protocol can be tested from the EFI Shell commands **`edit`** and **`hexedit`**. These EFI Shell commands automatically detect the presence of the Simple Pointer Protocol and will use it if it is present.
Snp32_64	A sample driver that consumes the services of a 32/64-bit UNDI and produces the Simple Network Protocol.
Undi	A sample PCI driver for a PCI Network Interface Controller (NIC). It consumes the services of the PCI I/O Protocol on the device handle for the PCI NIC and produces the Network Interface Identifier Protocol and registers an UNDI in the EFI System Table. This driver is specifically for the Intel® E100B class of NICs.
Usb\Uhci	A driver that consumes the PCI I/O Protocol for a USB host controller and produces the USB Host Controller Protocol. The USB Host Controller Protocol is used by the USB bus driver to produce handles of all the USB devices attached to the USB host controller.
Usb\UsbBot	A driver that consumes the USB I/O Protocol for a block-oriented transfer (BOT) device. This driver produces the USB ATAPI Protocol for all the USB BOT devices it discovers.
Usb\UsbBus	A driver that consumes the USB Host Controller Protocol and produces child handles for every USB device attached to the USB host controller. Because USB devices can be hot plugged into the system, this driver uses a timer service to continually see if USB devices have been attached or detached. When these events occur, child handles for the USB devices are either created or destroyed.
Usb\UsbCbi	A driver that consumes the USB I/O Protocol and produces the USB ATAPI Protocol for all the USB BOT devices that it discovers.
Usb\UsbKb	A driver that consumes the USB I/O Protocol and produces the Simple Input Protocol for every USB keyboard device that it discovers.
Usb\UsbMassStorage	A driver that consumes the USB ATAPI Protocol and produces the Block I/O Protocol. This driver works with the **`UsbBot`** and **`UsbCbi`** drivers to provide access to disk devices on a USB bus.
Usb\UsbMouse	A driver that consumes the USB I/O Protocol and produces the Simple Pointer Protocol for every USB mouse device it discovers.

continued

Table 3-7. Sample Drivers in the `Efi1.1\Edk\Drivers` Subdirectory (continued)

Driver Name	Description
VgaClass	A driver that consumes the VGA Mini Port Protocol and produces the Simple Text Output Protocol. This driver works with either the **`PciVgaMiniPort`** driver or the **`BiosVgaMiniPort`** driver to produce a console for a VGA device.
WinNtThunk\BlockIo	A driver that consumes the WinNt I/O Protocol and produces the Block I/O Protocol for disk devices in the NT emulation environment. These devices can be floppy drives, hard drives, CD-ROM drives, or virtual disk drives that are mapped as files. See **`EFI1.1\Build\Nt32\SYSTEM.CMD`** for a description of the environment variables that must be set to mount different disk devices.
WinNtThunk\Console	A driver that consumes the WinNt I/O Protocol and produces the Simple Text Output Protocol and Simple Input Protocol. This driver is responsible for producing the basic console services for a command window in the NT emulation environment. See **`EFI1.1\Build\Nt32\SYSTEM.CMD`** for a description of the environment variables that must be set to use the command window as a console.
WinNtThunk\SerialIo	A driver that consumes the WinNt I/O Protocol and produces the Serial I/O Protocol for serial ports that are available in the NT emulation environment. See **`EFI1.1\Build\Nt32\SYSTEM.CMD`** for a description of the environment variables that must be set to use serial ports.
WinNtThunk\SimpleFileSystem	A driver that consumes the WinNt I/O Protocol and produces the Simple File System Protocol for directories mounted in the NT emulation environment. See **`EFI1.1\Build\Nt32\SYSTEM.CMD`** for a description of the environment variables that must be set to mount directories.
WinNtThunk\Uga	A driver that consumes the WinNt I/O Protocol and produces the UGA Draw and UGA I/O Protocols for windows in the NT emulation environment. This sample is intended to show how the UGA Draw Protocol is able to function. The UGA I/O Protocol is not implemented in this sample. A fully compliant EFI UGA driver requires both the UGA Draw and UGA I/O Protocols. See Microsoft's documentation on UGA for details on writing a UGA driver that is able to function both in the EFI preboot environment and from the OS runtime. See **`EFI1.1\Build\Nt32\SYSTEM.CMD`** for a description of the environment variables that must be set to produce one or more UGA windows.

continued

Table 3-7. Sample Drivers in the `Efi1.1\Edk\Drivers` Subdirectory (continued)

Driver Name	Description
WinNtThunk\WinNtBusDriver	A driver that consumes the WinNt Thunk Protocol and produces child handles with the WinNt I/O Protocol based on environment variable settings that describe the set of devices to produce in the NT emulation environment. The WinNt Thunk Protocol is produced when the NT emulation environment is initialized. The WinNt Thunk Protocol implementation is in **`EFI1.1\Corefw\Fw\Platform\BuildTip\Nt32`**. See **`EFI1.1\Build\Nt32\SYSTEM.CMD`** for full description of all the environment variable settings that are used to describe devices that are available from the NT emulation environment.
WinNtThunk\WinNtPciRootBridge	A sample driver that produces a PCI Root Bridge I/O Protocol from the *EFI 1.10 Specification*. This PCI root bridge is simulated in the NT emulation environment. This driver can be extended to produce a set of simulated PCI devices so some PCI driver development work can be performed in the NT emulation environment.

4
Developing EFI 1.10 PCI Device Drivers

4.1 EFI Driver Model Protocols

The IDEBUS driver is a sample driver that implements all the EFI Driver Model protocols including the Driver Binding Protocol, Component Name Protocol, Driver Configuration Protocol, and Driver Diagnostics Protocol. Please use this driver a reference for how these protocols are implemented.

4.2 PCI Video Adapters

This release includes two sample UGA drivers. One is for a Cirrus Logic 5430/5446 PCI video adapter, and the other is for the NT emulation environment. These samples are intended to show how the UGA Draw Protocol is able to function. The UGA I/O Protocol is not implemented in these samples. A fully compliant EFI UGA driver requires both the UGA Draw and UGA I/O Protocols. See Microsoft's documentation on UGA for details on how to write a UGA driver that is able to function both in the EFI preboot environment and from the OS runtime. Please join the EFI Mailing List from the EFI Web site to receive the latest information on the UGA specification and sample code.

Once a UGA Draw Protocol is implemented, there are several ways to test the implementation:

- The first is a new EFI Shell command called **`LoadBmp`**. This command allows standard **`.BMP`** files to be loaded from disk and displayed on a UGA display device.
- Another way to test the UGA Draw Protocol is with the **`GraphicsConsole`** driver. This driver produces the Simple Text Output Protocol on top of a UGA display device. It contains a standard font, and it uses the UGA Draw services to copy the bitmaps of the font characters onto the UGA display device.

The text mode VGA drivers have a Driver Binding Version value of 0x00000000. This value gives them the lowest priority of all drivers in the system. If a UGA driver has a Driver Binding Protocol *`Version`* value greater than 0x00000000, then it can replace the text mode VGA driver. To replace this driver, soft load the UGA driver from the EFI Shell with the **`load -nc`** command, and then execute the **`reconnect -r`** command. The **`reconnect`** command should disconnect the text-mode VGA driver from the video adapter and then connect the UGA driver to the video adapter. The **`GraphicsConsole`** driver will automatically bind to the UGA driver and produce a graphical console device.

Once a UGA driver is fully debugged, the **`EFIROM`** tool can be used to produce a PCI option ROM image. The PCI option ROM image can be tested with the EFI Shell command **`LoadPciRom`**. It can also be tested by programming the image into the adapter's option ROM. When a PCI option ROM is discovered by the PCI bus driver, all the EFI drivers in that option ROM are automatically loaded and executed.

4.3 PCI Network Adapters

PCI network adapters need to produce a 32/64-bit UNDI. The **Undi** driver is a sample driver for an Intel E100B PCI network adapter. This driver can be used as a reference for developing EFI 1.10 drivers for PCI network adapters. The **Snp32_64** and **PxeBc** drivers should automatically bind to the UNDI driver when it is initialized. These drivers will produce the Simple Network Protocol, PXE Base Code Protocol, and Load File Protocol. The Load File Protocol will show up from the EFI boot manager. Performing a PXE boot from a PXE server will test the UNDI driver.

An UNDI driver can be tested by loading it from the EFI Shell with the **load** command. Once the network stack is connected, a new boot option will be available from the EFI Boot Maintenance Manager in the “Boot from a File” selection. Selecting the new boot option will use the services of the UNDI driver to perform a PXE boot from a PXE server.

Once an UNDI driver is fully debugged, the **EFIROM** tool can be used to produce a PCI option ROM image. The PCI option ROM image can be tested with the EFI Shell command **LoadPciRom**. It can also be tested by programming the image into the adapter’s option ROM. When a PCI option ROM is discovered by the PCI bus driver, all the EFI drivers in that option ROM are automatically loaded and executed.

4.4 PCI SCSI Adapters

PCI SCSI adapters need to produce a SCSI Pass Thru Protocol for each SCSI channel on the SCSI adapter. The AtapiPassThru driver is a sample driver for a PCI IDE controller. This driver can be used as a reference for developing EFI 1.10 drivers for PCI SCSI adapters. The EFI driver will also have to produce a Block I/O Protocol for each disk device that is present on the SCSI channel. The implementation of the Block I/O Protocol should use the services of the SCSI Pass Thru Protocol to access the disk devices.

The Disk I/O and File System Protocols will bind to the Block I/O Protocols. The disk devices that are discovered will be available from the EFI Shell. The EFI Shell commands **dblk** and **hexedit** can be used to view raw blocks from a Block I/O Protocol. If the disk device is formatted FAT, it will also show up as a mountable drive from the EFI Shell. Then, the file commands in the EFI Shell can be used to test the new driver.

Once an SCSI driver is fully debugged, the **EFIROM** tool can be used to produce a PCI option ROM image. The PCI option ROM image can be tested with the EFI Shell command **LoadPciRom**. It can also be tested by programming the image into the adapter’s option ROM. When a PCI option ROM is discovered by the PCI bus driver, all the EFI drivers in that option ROM are automatically loaded and executed.

4.5 Adding a Device Driver to the Build Environment

To develop a new device driver, add a new directory under the **Efi1.1\Edk\Drivers** directory. Place your source code and a **MAKE.INF** file in this new directory. Look at the other drivers for examples on how to build **MAKE.INF** files.

The **Ia32Drivers** and **IpfDrivers** build tips contain a **makefile** that lists all the drivers to build. To add a driver to these build environments, you need to add a new line to this **makefile**. The **.EFI** driver executable will be in the **bin** directory below the build tip.

4.6 Loading PCI Device Drivers

Once a device driver has been written and compiled, it can be loaded with the EFI Shell command **`load -nc`**. An EFI 1.1 device driver only adds the **`EFI_DRIVER_BINDING_PROTOCOL`** to the image handle of the device driver and exits. As a result, no action is taken by the driver when the **`load -nc`** command is executed. The EFI Shell commands **`connect`** and **`disconnect`** can be used to connect the driver to a device or disconnect the drivers from a device. When a driver is connected the **`Supported()`** and **`Start()`** functions of the Driver Binding Protocol are executed. When a driver is disconnected, the **`Stop()`** function of the Driver Binding Protocol is executed. The **`connect -r`** command will connect all drivers to all devices recursively.

Once a driver is fully debugged, it can be placed on a hard drive or floppy, and the EFI Shell command **`bcfg`** can be used to configure the system to automatically load and start a driver each time the EFI boot manager is executed.

4.7 Debugging PCI Device Drivers

There are several EFI Shell commands that are available to help debug EFI drivers:

- **`connect`**
- **`disconnect`**
- **`reconnect`**
- **`OpenInfo`**
- **`drivers`**
- **`devices`**
- **`devTree`**
- **`drvcfg`**
- **`drvdiag`**
- A new version of the **`dh`** command that now has a **`-d`** option.

See the *EFI 1.10 Shell Command Specification* for details on these commands.

The **`drivers`**, **`devices`**, **`devtree`**, and **`dh`** commands show the drivers that are following the EFI Driver Model and the devices that those drivers are currently managing. Each command shows this information from a slightly different perspective.

The **`drvcfg`** command provides an easy way to test a driver's Driver Configuration Protocol implementation, and the **`drvdiag`** command provides an easy way to test a driver's Driver Diagnostic Protocol implementation.

The EFI Shell commands **`dh`** and **`OpenInfo`** can be used to view image handles and device handles. The **`OpenInfo`** command is useful because it shows the list of agents that currently have each of the protocol interfaces on a handle open.

The **`connect`** and **`disconnect`** commands allow drivers to be dynamically connected and disconnected from their devices that they manage. The **`disconnect`** command can be dangerous, because you may accidentally disconnect all the console devices. The EFI Shell will detect this case and automatically reconnect the console devices.

The **reconnect** command is basically a combination of a **disconnect** followed by a **connect** command. The **reconnect** command is a very good test to make sure a driver has implemented its **Start()** and **Stop()** functions correctly. The command **reconnect –r** will disconnect all drivers from all the devices in the system and then reconnect them. This command will test to see if all the drivers are following the interoperability rules of the EFI Driver Model. All of the drivers in the *EFI Version 1.10.14.62 Sample Implementation* pass the **reconnect –r** test. Any new EFI drivers should use this test to make sure they are also following the EFI Driver Model interoperability rules.

4.8 Building and Loading PCI Option ROMs

A build utility called **EFIROM** is shipped as part of the *EFI Version 1.10.14.62 Sample Implementation*, and it can be used to generate a binary file that may contain a legacy PCI option ROM image and EFI driver images. The EFI Shell command **LoadPciRom** can be used to load a file that was generated by the **EFIROM** utility to load all the EFI drivers from the PCI option ROM image. This command simulates the image loading that would be performed by the PCI bus driver if the option ROM on the adapter were programmed with the binary image that was generated by the **EFIROM** utility. Once testing with **LoadPciRom** is complete, the image can be programmed into the PCI option ROM on a PCI adapter, and the PCI bus driver will load and execute the EFI drivers from the PCI option ROM.

5
EFI 1.10 Build Tools

Table 5-1 lists the build tools that are shipped with the *EFI Version 1.10.14.62 Sample Implementation.*

Table 5-1. Build Tools in EFI 1.10.14.62 Sample Implementation

Build Tool	Description
Col	A utility that converts tabs to spaces on source files.
DskImage	A utility that transfers a disk image file to a floppy disk.
EfiCompress	A utility that compresses a file using the Compression Algorithm in the *EFI 1.10 Specification.* A file that is compressed using this utility can be decompressed using the services of the Decompress Protocol or the EFI Shell command **`EfiDecompress`**.
EfiLdrImage	A utility that is used to build a bootable image for the **`Bios32`** and **`Ia-32Emb`** environments. The build environment uses this tool internally.
EfiRom	A utility that creates a binary file that is a PCI option ROM image that conforms to the *PCI 2.2 Specification.* It takes as input either raw images, EFI images, or any combination of the two. By default all EFI images are compressed, and the images are placed in the PCI option ROM in the order in which they are passed to this tool. The EFI Shell command **`LoadPciRom`** can be used to load uncompressed and compressed EFI drivers from the file that this tool generates. The binary file that this tool generates is also the same image that can be directly programmed into a PCI option ROM.
Futil	A utility that reprograms the flash on an Intel® NIC.
FwImage	A utility that adjusts the PE/COFF header to mark a file as an EFI application, EFI Boot Services Driver, or EFI runtime driver. The build environment uses this tool internally.
GenHelp	A utility that converts a Unicode text file into a **`.C`** file with an array of **`CHAR16`** values. This tool is used to convert the EFI Shell help information in **`\EFI1.1\Shell\HelpData\HelpData.Src`** into **`HelpData.c`** during the build process.
GenMake	A utility that converts **`MAKE.INF`** files to **`makefile`**s. The build environment uses this tool internally.
SplitFile	A utility that is used to build a bootable image for the **`Bios32`** and **`Ia-32Emb`** environments. The build environment uses this tool internally.

6
Porting the Core EFI Firmware to a New Platform

Each platform will need its own copy of **`init.c`**. This file will contain **`MainEntry()`**, which is the code that the platform will call to pass control to EFI. The platform is responsible for loading and relocating the EFI code and calling its entry point.

Code in the **`Platform`** tree should never call EFI code directly. All core EFI code must be called via **`FW (`***`pointer to`*** **`EFI_FIRMWARE_TABLE)`**. The exception to this rule is that the platform code may call EFI Library functions. EFI Library functions can be called only after the platform's memory map has been initialized and a call to **`InitializeLib()`** has been made. The set of drivers that a platform loads may need to be adjusted on a platform-specific basis. In addition, new drivers may have to be written for platform/chipset-specific functions. Examples of these specific functions include Real Time Clock access, EFI Variable Services, and PCI Root Bridge I/O Protocol services.

The main flow of the EFI code is controlled from the **`Platform`** code, and the calls back into the core firmware must be made in sequence. The following is the process to initialize EFI:

1. The platform-specific code loads and relocates EFI code.
2. The platform-specific code calls **`MainEntry()`**. The source tree starts running.
3. The platform-specific code initializes tables:
   a. It gets the initial EFI System Table.
   b. It patches in any Runtime or Boot Services functions that not supported by the EFI core.
4. The platform-specific code calls **`PlInstallMemoryMap()`** to set up the EFI memory map. First, call back to firmware to add a memory descriptor. The parameter that is passed in must be conventional memory, which will then be used to build the EFI memory map.
5. The firmware is called with an initialized memory map.
6. Install the system configuration tables:
   - ACPI
   - SMBIOS
   - MPS
7. Initialize the EFI Library. Library calls can now be made.
8. Load all EFI 1.10 drivers that are linked into the EFI core firmware.
9. Install base devices:
   a. Call the firmware to install base I/O devices
   b. Initialize internal NVRAM.
10. The firmware is called to install the NVRAM store. On **`Bios32`** systems where variables are stored in a file on a fixed disk, the NVRAM store is in the system volumes.
11. Load all EFI 1.10 drivers that are linked into the EFI core firmware that require the EFI Variables Services.
12. Install the legacy console drivers:
    a. Add the BIOS keyboard driver.
    b. Add the BIOS VGA driver.

13. Connect all EFI 1.10 drivers to all controllers. This point is the first time that a console might be available.
14. The firmware is called with a working console.
15. Install system volumes. Add system block I/O drivers and BIOS block I/O drivers.
16. The firmware is called with the system volumes that were installed.
17. Install other devices and whatever extra you need. The NT emulator adds an emulated or raw floppy device. It also adds bogus NT file system access at this point.
18. Call the firmware boot manager.
19. Call the platform boot manager.

7
Using the EFI Boot Manager

The EFI Boot Maintenance Manager allows the user to do the following:

- Add boot options.
- Delete boot options.
- Launch an EFI application.
- Set the auto boot timeout value.

If there is no boot option in the system (and no integrated Shell), the Boot Maintenance Menu is presented. If boot options are available, then the set of available boot options is displayed, and the user can select one or choose to go to the Boot Maintenance Menu. If the timeout period is not zero, then the system will automatically boot the first boot selection after the timeout has expired. If the timeout period is zero, then the EFI boot manager will wait for the user to select an option.

Table 7-1 lists the options that are available on the Boot Maintenance Menu.

Table 7-1. Boot Maintenance Menu Options

No.	Menu Option	Description
1	Boot from a file	This option searches all the EFI system partitions in the system. For each partition, it looks for an **`EFI`** directory. If the **`EFI`** directory is found, then it looks in each of the subdirectories below **`EFI`**. In each of those subdirectories, it looks for the first file that is an executable EFI application. Each of the EFI applications that meets these criteria is automatically added as a boot option. In addition, legacy boot options for **`A:`** and **`C:`** are also added if those devices are present. This option allows the user to launch an application without adding it as a boot option. The EFI boot manager will search the root directories and the **`\EFI\TOOLS`** directories of all of the EFI system partitions that are present in the system for the specified EFI application.
2	Add a boot option	This option allows the user to specify the name of the EFI application to add as a boot option. The EFI boot manager searches the same partitions and directories as described in 1), until it finds an EFI application with the same name that the user specified. This menu also allows the user to provide either ASCII or UNICODE arguments to the option that will be launched.
3	Delete boot options	This option provides the opportunity to delete any single boot options or all boot options.
4	Change boot order	This option gives the user control over the relative order in which the EFI boot manager will attempt to boot the options. There is a help menu to describe the control key sequences to use.

continued

Table 7-1. Boot Maintenance Menu Options (continued)

No.	Menu Option	Description
5	Manage BootNext setting	This option gives the user the ability to prescribe the first boot option to be tried on a subsequent boot.
6	Set auto boot time out	This option allows the user to set the auto boot timeout value in seconds. If it is set to zero, then the auto boot feature is disabled.
7	Select active console output devices	This option displays the list of available console output devices, as contained in the *ConOutDev* list of volatile variables. The console output devices that have been selected to be active consoles are annotated as such. The user can select or deselect additional output consoles from this menu. The Boot Maintenance Manager will perform logic checking to ensure that a legal ensemble of devices is chosen (i.e., you cannot choose two different messaging devices, such as both PC-ANSI and VT-100, to be active consoles on a given UART). The system should choose a set of defaults in an implementation-specific fashion if the Console Out variable is empty; this case could be expected for the first boot of a given system.
8	Select active console input devices	This option is the same as option 7 above, but it treats the *ConInDev* list of devices and the subset detailed in the *ConIn* variable.
9	Select active error devices	This option is the same as option 7, but it treats the *ErrOutDev* list of devices and the subset detailed in the *ErrOut* variable. The active error devices are essentially a type of console output devices whose only traffic includes error messaging.
10	Cold Reset	This option will perform a platform-specific cold reset of the system. This action has traditionally meant a full platform reset.
11	Exit	Return to the boot manager main menu. This option will display the active boot devices, including a possible integrated Shell (if available).

Appendix A
Driver Map

The following table is a map of all the drivers in the *EFI Version 1.10.14.62 Sample Implementation* and the build tips that the drivers are built and used.

Table A-1. Map of Drivers in EFI Version 1.10.14.62 Sample Implementation

Driver	BIOS32	IA-32Emb	SAL64	NT32	Ia32Drivers	IpfDrivers	EbcDrivers
AtapiPassThru		X			X	X	
Bis	X	X		X	X	X	
CirrusLogic5430	X	X			X	X	X
Console\ConPlatform	X	X	X	X	X	X	
Console\ConSplitter	X	X	X	X	X	X	
Console\GraphicsConsole	X	X	X	X	X	X	
Console\Terminal	X	X	X	X	X	X	
DebugPort					X	X	
DebugSupport					X	X	
Decompress	X	X	X	X	X	X	
DiskIo	X	X	X	X	X	X	
Ebc	X	X	X	X	X	X	
FileSystem\Fat	X	X	X	X	X	X	
Ide		X			X	X	X
IsaBus	X	X	X		X	X	
IsaFloppy		X			X	X	
IsaSerial	X	X	X		X	X	
Partition	X	X	X	X	X	X	
PcatIsaAcpi		X			X	X	
PcatIsaAcpiBios	X		X		X	X	

continued

Table A-1. Map of Drivers in EFI Version 1.10.14.62 Sample Implementation (continued)

Driver	BIOS32	IA-32Emb	SAL64	NT32	Ia32Drivers	IpfDrivers	EbcDrivers
PcatPciRootBridge	X	X	X		X	X	
PciBus	X	X	X	X	X	X	
PciVgaMiniPort		X			X	X	
Ps2Keyboard		X			X	X	
Ps2Mouse		X			X	X	
PxeBc	X	X	X		X	X	
PxeDhcp4	X	X	X		X	X	
ScsiBus					X	X	
ScsiDisk					X	X	
SerialMouse	X	X	X	X	X	X	
Snp32_64	X	X	X		X	X	
Undi	X	X	X		X	X	
Usb\Uhci		X			X	X	
Usb\UsbBot		X			X	X	
Usb\UsbBus		X			X	X	
Usb\UsbCbi		X			X	X	
Usb\UsbKb		X			X	X	
Usb\UsbMassStorage		X			X	X	
Usb\UsbMouse		X			X	X	
VgaClass	X	X	X		X	X	
WinNtThunk\BlockIo				X			
WinNtThunk\Console				X			
WinNtThunk\SerialIo				X			
WinNtThunk\SimpleFileSystem				X			
WinNtThunk\Uga				X			
WinNtThunk\WinNtBusDriver				X			

continued

Table A-1. Map of Drivers in EFI Version 1.10.14.62 Sample Implementation (continued)

Driver	BIOS32	IA-32Emb	SAL64	NT32	Ia32Drivers	IpfDrivers	EbcDrivers
WinNtThunk\WinNtPciRootBridge				X			
BiosInt\BiosKeyboard	X		X				
BiosInt\BiosSnp16	X		X				
BiosInt\BiosVga			X				
BiosInt\BiosVgaMiniPort	X		X				
BiosInt\Disk	X		X				

Appendix B
New Features and Bug Fixes

This section lists the bug fixes, new features, and files that were added, changed, or deleted in the different released versions of the *EFI Sample Implementation.*

B.1 EFI 1.10.14.62

B.1.1 Core Bug Fixes

1. Updated version to EFI 1.10.14.62.
2. Updated the BIOS32 boot floppy image to EFI 1.10.14.62.
3. Updated the release notes to EFI 1.10.14.62.
4. Remove all the duplicate symbols from the source code. Changed the duplicate name so that the debugger can find the correct symbol.
5. Updated device detection algorithm. IdeBus driver can now properly detect slow ATAPI devices such as LS-120 or ZIP that could not be detected by the old version.
6. Added more parameter checking to DeviceIo.PciDevicePath().
7. Added more parameter checking to CalculateCrc32().
8. Added more parameter checking to PciRootBridgeIo.FreeBuffer().
9. Made SetAttributes() of SerialIO more consistent with *EFI 1.10 Specification.*
10. Added more parameter checking to PciRootBridgeIo.Unmap().
11. Added flow control bit supported to SetControl() service of SerialIo.
12. Added more parameter checking to SetTime().
13. Added more parameter checking to SetControl() service of SerialIo.
14. Added more parameter checking to PciRootBridgeIo.GetAttributes().
15. Added more parameter checking to PciRootBridgeIo Mem.Read/Write and Io.Read/Write.
16. Added more parameter checking to SetAttributes() of SerialIo.
17. Added more parameter checking to PciRootBridgeIo.PollMem() and PollIo().
18. Updated to return correct status when calling DeviceIo.Map() with Address > 4 GB.
19. Fixed a bug in SerialIo that actual BaudRate will exceed the giving BaudRate setting.
20. Added more parameter checking to DeviceIo.Map().
21. Added more parameter checking to DeviceIo.FreeBuffer().
22. Updated bcfg to use HEX for both list index and remove index.
23. Updated IDE I/O to decode 16 bits instead of 10 bits.
24. Updated to allow 16-bit decode of VGA I/O resources.
25. Fixed a bug in the FAT that returned an unexpected status code for file access after media is removed.
26. Fixed a bug in the newer edit.efi on Intel® Server Platform SR870BH2 with Unicode files.
27. Fixed a bug in the type command for large files on SCSI on Intel® Server Platform SR870BN4–based systems.
28. Fixed a memory leak bug in the PS/2 mouse driver.

29. Fixed a bug that PCI.efi report devsel time incorrectly.
30. Updated all serial console drivers to use ESC [= 3 h.
31. Fixed a bug that EFI EBC interpreter causes alignment faults on the Itanium 2 processor.
32. Updated new device IDs for the silicon in Intel PRO/100 network adapters.
33. Fix an IA-32 floating point bug in EFI applications. The optimized CopyMem() and ZeroMem() functions use registers for Intel® MMX™ technology (the MMX registers). Any function that uses MMX registers must have an EMMS instruction at the end of the function.
34. Added more parameter checking to Supported() function of partition driver.
35. Added more parameter checking to CreateEvent().
36. Updated to case-insensitive implement for UNICODE_COLLATION.MetaiMatch().
37. Fixed bug for UNICODE_COLLATION.StrToFat().
38. Added more parameter checking to Partition:BlockIo:ReadBlocks().
39. Added more parameter checking to ConsoleSplitter:UgaDraw->GetMode().
40. Fixed a bug that ConsoleSplitter:UgaDraw->Blt uses wrong Pixel to fill video.
41. Fixed a bug that ConSplitter:UgaDraw operate wrong with Delta of zero.
42. Added more parameter checking to VgaClass, SimpleTextOut->SetAttribute().
43. Added more status checking to SimplePointerProtocol->GetState().
44. Fixed a bug that PCI command causes alignment fault on the Itanium processor family for 64-bit BARs.
45. Updated to check the BlockIo that buffer size should be a multiple of block size.
46. Fixed a bug that PXE_BC.Start() does not zero-fill the Icmperror and Tftperror structures.
47. Fixed a bug for Virtual Memory Services: ConvertPointer() success for NULL.
48. Added more parameter checking to pciio functions Mem.read() and Mem.write().
49. Added more parameter checking to PCI_IO_PROTOCOL Map() and PCI_IO_PROTOCOL UnMap().
50. Added more parameter checking to USB_HC.SyncInterruptTransfer.
51. Fixed a bug that DebugPort.Write() does not time out.
52. Updated EFILDR for BIOS32 and IA-32EMB to enable source-level debugging of the PE/COFF loader.
53. Fixed PCI I/O bar bugs with 64-bit addresses.
54. Fixed USB UHCI bug and USB bus driver reset bug.
55. Fixed a bug in which an IBM TrackPoint* USB keyboard caused an EFI 1.1 Intel Server Platform SR870BN4–based system to hang.
56. Fixed a USB bug for Maxpacketlength, which now allows USB 1.44 floppies to function.
57. Improved search method in VarStoreAddBank().
58. Fixed IA-32 FreePoolPages() bug for mem between 2 GB and 4 GB.
59. Fixed Boot Maintenance Manager Bug in “Add a Boot Option” and “Delete Boot Option(s).” After editing existing boot option or making a new entry or deleting a boot option, if we select not to save changes to NVRAM, the changes are still saved.
60. Fixed Boot Maintenance Manager bug in “Select Active Console.” Can select a Uart as PcAnsi in “Select Active Console Output Devices,” then can select the same Uart as Vt100+ in “Select Active Console Input Devices.”
61. Fixed Boot Maintenance Manager bug in “Add a Boot Option.” Only the first entry can be edited in a volume. We cannot select to edit other entries.

62. Added more parameter checking to BS.InstallProtocolInterface().
63. Added more parameter checking to PciIo.GetBarAttributes().
64. Added more parameter checking to PciIo.FreeBuffer().
65. Added more parameter checking to GetVariable().
66. Added more parameter checking to PciIo.SetBarAttributes().
67. Added more parameter checking to FreePool().
68. Added more parameter checking to GetNextMonotonicCount().
69. Added more parameter checking to PciIo.CopyMem().
70. Fixed SetControl () of SerialIo to clean Software Loopback bit.
71. Added more parameter checking to PciIo.PollMem()/PollIo().

B.1.2 List of Deleted Files

EFI1.1\COREFW\fw\platform\PlDriver\Defio\Pci\IA32\

EFI1.1\COREFW\fw\platform\PlDriver\Defio\Pci\Ipf\

EFI1.1\EDK\Drivers\ebc\Ebc\

EFI1.1\NOTES\EFI1.1ShellCommands.DOC

EFI1.1\NOTES\EFI_LIB.DOC

EFI1.1\NOTES\EfiDriverLib.Doc

EFI1.1\NOTES\RELNOTE.DOC

EFI1.1\NOTES\SalEfiHandOffState.Doc

EFI1.1\NOTES\ScsiDriverModel.doc

B.1.3 List of New Files

EFI1.1\Binary\BIOS32\Bin\bis.efi

EFI1.1\Binary\BIOS32\Bin\bmaint.efi

EFI1.1\Binary\BIOS32\Bin\bootmgr.efi

EFI1.1\Binary\BIOS32\Bin\CirrusLogic5430Uga.efi

EFI1.1\Binary\BIOS32\Bin\ConPlatform.efi

EFI1.1\Binary\BIOS32\Bin\ConSplitter.efi

EFI1.1\Binary\BIOS32\Bin\DebugPort.efi

EFI1.1\Binary\BIOS32\Bin\DebugSupport.efi

EFI1.1\Binary\BIOS32\Bin\Decompress.efi

EFI1.1\Binary\BIOS32\Bin\DiskIo.efi

EFI1.1\Binary\BIOS32\Bin\EBC.efi

EFI1.1\Binary\BIOS32\Bin\Fat.efi

EFI1.1\Binary\BIOS32\Bin\GraphicsConsole.efi

EFI1.1\Binary\BIOS32\Bin\IsaBus.efi

EFI1.1\Binary\BIOS32\Bin\IsaSerial.efi

EFI1.1\Binary\BIOS32\Bin\Partition.efi
EFI1.1\Binary\BIOS32\Bin\PcatIsaAcpi.efi
EFI1.1\Binary\BIOS32\Bin\PcatPciRootBridge.efi
EFI1.1\Binary\BIOS32\Bin\PciBus.efi
EFI1.1\Binary\BIOS32\Bin\PxeBc.efi
EFI1.1\Binary\BIOS32\Bin\PxeDhcp4.efi
EFI1.1\Binary\BIOS32\Bin\SerialMouse.efi
EFI1.1\Binary\BIOS32\Bin\Snp3264.efi
EFI1.1\Binary\BIOS32\Bin\Terminal.efi
EFI1.1\Binary\BIOS32\Bin\Undi.efi
EFI1.1\Binary\BIOS32\Bin\VgaClass.efi
EFI1.1\Binary\BIOS32\BIOS32image\BIOS32.IMG
EFI1.1\Binary\BIOS32\SHELLBios32\Shell.efi
EFI1.1\Binary\EBCDriver\BIN\CirrusLogic5430Uga.efi
EFI1.1\Binary\EBCDriver\BIN\IdeBus.efi
EFI1.1\Binary\IA-32EMB\BIN\bis.efi
EFI1.1\Binary\IA-32EMB\BIN\bmaint.efi
EFI1.1\Binary\IA-32EMB\BIN\bootmgr.efi
EFI1.1\Binary\IA-32EMB\BIN\CirrusLogic5430Uga.efi
EFI1.1\Binary\IA-32EMB\BIN\ConPlatform.efi
EFI1.1\Binary\IA-32EMB\BIN\ConSplitter.efi
EFI1.1\Binary\IA-32EMB\BIN\Decompress.efi
EFI1.1\Binary\IA-32EMB\BIN\DiskIo.efi
EFI1.1\Binary\IA-32EMB\BIN\EBC.efi
EFI1.1\Binary\IA-32EMB\BIN\Fat.efi
EFI1.1\Binary\IA-32EMB\BIN\GraphicsConsole.efi
EFI1.1\Binary\IA-32EMB\BIN\IdeBus.efi
EFI1.1\Binary\IA-32EMB\BIN\IsaBus.efi
EFI1.1\Binary\IA-32EMB\BIN\IsaFloppy.efi
EFI1.1\Binary\IA-32EMB\BIN\IsaSerial.efi
EFI1.1\Binary\IA-32EMB\BIN\Partition.efi
EFI1.1\Binary\IA-32EMB\BIN\PcatIsaAcpi.efi
EFI1.1\Binary\IA-32EMB\BIN\PcatPciRootBridge.efi
EFI1.1\Binary\IA-32EMB\BIN\PciBus.efi
EFI1.1\Binary\IA-32EMB\BIN\PciVgaMiniPort.efi
EFI1.1\Binary\IA-32EMB\BIN\Ps2Keyboard.efi

EFI1.1\Binary\IA-32EMB\BIN\Ps2Mouse.efi
EFI1.1\Binary\IA-32EMB\BIN\PxeBc.efi
EFI1.1\Binary\IA-32EMB\BIN\PxeDhcp4.efi
EFI1.1\Binary\IA-32EMB\BIN\SerialMouse.efi
EFI1.1\Binary\IA-32EMB\BIN\Snp3264.efi
EFI1.1\Binary\IA-32EMB\BIN\Terminal.efi
EFI1.1\Binary\IA-32EMB\BIN\Undi.efi
EFI1.1\Binary\IA-32EMB\BIN\UsbBot.efi
EFI1.1\Binary\IA-32EMB\BIN\UsbBus.efi
EFI1.1\Binary\IA-32EMB\BIN\UsbCbi0.efi
EFI1.1\Binary\IA-32EMB\BIN\UsbCbi1.efi
EFI1.1\Binary\IA-32EMB\BIN\UsbKeyboard.efi
EFI1.1\Binary\IA-32EMB\BIN\UsbMassStorage.efi
EFI1.1\Binary\IA-32EMB\BIN\UsbMouse.efi
EFI1.1\Binary\IA-32EMB\BIN\UsbUhci.efi
EFI1.1\Binary\IA-32EMB\BIN\VgaClass.efi
EFI1.1\Binary\IA-32EMB\IA-32EMBimage\IA-32EMB.IMG
EFI1.1\Binary\IA-32EMB\SHELLIA32emb\Shell.efi
EFI1.1\Binary\IA32Drivers\BIN\AtapiPassThru.efi
EFI1.1\Binary\IA32Drivers\BIN\bis.efi
EFI1.1\Binary\IA32Drivers\BIN\CirrusLogic5430Uga.efi
EFI1.1\Binary\IA32Drivers\BIN\ConPlatform.efi
EFI1.1\Binary\IA32Drivers\BIN\ConSplitter.efi
EFI1.1\Binary\IA32Drivers\BIN\DebugPort.efi
EFI1.1\Binary\IA32Drivers\BIN\DebugSupport.efi
EFI1.1\Binary\IA32Drivers\BIN\Decompress.efi
EFI1.1\Binary\IA32Drivers\BIN\DiskIo.efi
EFI1.1\Binary\IA32Drivers\BIN\EBC.efi
EFI1.1\Binary\IA32Drivers\BIN\Fat.efi
EFI1.1\Binary\IA32Drivers\BIN\GraphicsConsole.efi
EFI1.1\Binary\IA32Drivers\BIN\IdeBus.efi
EFI1.1\Binary\IA32Drivers\BIN\IsaBus.efi
EFI1.1\Binary\IA32Drivers\BIN\IsaFloppy.efi
EFI1.1\Binary\IA32Drivers\BIN\IsaSerial.efi
EFI1.1\Binary\IA32Drivers\BIN\Partition.efi
EFI1.1\Binary\IA32Drivers\BIN\PcatIsaAcpi.efi

EFI1.1\Binary\IA32Drivers\BIN\PcatPciRootBridge.efi
EFI1.1\Binary\IA32Drivers\BIN\PciBus.efi
EFI1.1\Binary\IA32Drivers\BIN\PciVgaMiniPort.efi
EFI1.1\Binary\IA32Drivers\BIN\Ps2Keyboard.efi
EFI1.1\Binary\IA32Drivers\BIN\Ps2Mouse.efi
EFI1.1\Binary\IA32Drivers\BIN\PxeBc.efi
EFI1.1\Binary\IA32Drivers\BIN\PxeDhcp4.efi
EFI1.1\Binary\IA32Drivers\BIN\ScsiBus.efi
EFI1.1\Binary\IA32Drivers\BIN\ScsiDisk.efi
EFI1.1\Binary\IA32Drivers\BIN\SerialMouse.efi
EFI1.1\Binary\IA32Drivers\BIN\Snp3264.efi
EFI1.1\Binary\IA32Drivers\BIN\Terminal.efi
EFI1.1\Binary\IA32Drivers\BIN\Undi.efi
EFI1.1\Binary\IA32Drivers\BIN\UsbBot.efi
EFI1.1\Binary\IA32Drivers\BIN\UsbBus.efi
EFI1.1\Binary\IA32Drivers\BIN\UsbCbi1.efi
EFI1.1\Binary\IA32Drivers\BIN\UsbKeyboard.efi
EFI1.1\Binary\IA32Drivers\BIN\UsbMassStorage.efi
EFI1.1\Binary\IA32Drivers\BIN\UsbMouse.efi
EFI1.1\Binary\IA32Drivers\BIN\UsbUhci.efi
EFI1.1\Binary\IA32Drivers\BIN\VgaClass.efi
EFI1.1\Binary\IPFDrivers\BIN\AtapiPassThru.efi
EFI1.1\Binary\IPFDrivers\BIN\bis.efi
EFI1.1\Binary\IPFDrivers\BIN\CirrusLogic5430Uga.efi
EFI1.1\Binary\IPFDrivers\BIN\ConPlatform.efi
EFI1.1\Binary\IPFDrivers\BIN\ConSplitter.efi
EFI1.1\Binary\IPFDrivers\BIN\DebugPort.efi
EFI1.1\Binary\IPFDrivers\BIN\DebugSupport.efi
EFI1.1\Binary\IPFDrivers\BIN\Decompress.efi
EFI1.1\Binary\IPFDrivers\BIN\DiskIo.efi
EFI1.1\Binary\IPFDrivers\BIN\EBC.efi
EFI1.1\Binary\IPFDrivers\BIN\Fat.efi
EFI1.1\Binary\IPFDrivers\BIN\GraphicsConsole.efi
EFI1.1\Binary\IPFDrivers\BIN\IdeBus.efi
EFI1.1\Binary\IPFDrivers\BIN\IsaBus.efi
EFI1.1\Binary\IPFDrivers\BIN\IsaFloppy.efi

EFI1.1\Binary\IPFDrivers\BIN\IsaSerial.efi
EFI1.1\Binary\IPFDrivers\BIN\Partition.efi
EFI1.1\Binary\IPFDrivers\BIN\PcatIsaAcpi.efi
EFI1.1\Binary\IPFDrivers\BIN\PcatPciRootBridge.efi
EFI1.1\Binary\IPFDrivers\BIN\PciBus.efi
EFI1.1\Binary\IPFDrivers\BIN\PciVgaMiniPort.efi
EFI1.1\Binary\IPFDrivers\BIN\Ps2Keyboard.efi
EFI1.1\Binary\IPFDrivers\BIN\Ps2Mouse.efi
EFI1.1\Binary\IPFDrivers\BIN\PxeBc.efi
EFI1.1\Binary\IPFDrivers\BIN\PxeDhcp4.efi
EFI1.1\Binary\IPFDrivers\BIN\ScsiBus.efi
EFI1.1\Binary\IPFDrivers\BIN\ScsiDisk.efi
EFI1.1\Binary\IPFDrivers\BIN\SerialMouse.efi
EFI1.1\Binary\IPFDrivers\BIN\Snp3264.efi
EFI1.1\Binary\IPFDrivers\BIN\Terminal.efi
EFI1.1\Binary\IPFDrivers\BIN\Undi.efi
EFI1.1\Binary\IPFDrivers\BIN\UsbBot.efi
EFI1.1\Binary\IPFDrivers\BIN\UsbBus.efi
EFI1.1\Binary\IPFDrivers\BIN\UsbCbi1.efi
EFI1.1\Binary\IPFDrivers\BIN\UsbKeyboard.efi
EFI1.1\Binary\IPFDrivers\BIN\UsbMassStorage.efi
EFI1.1\Binary\IPFDrivers\BIN\UsbMouse.efi
EFI1.1\Binary\IPFDrivers\BIN\UsbUhci.efi
EFI1.1\Binary\IPFDrivers\BIN\VgaClass.efi
EFI1.1\Binary\NT32\BIN\AtapiPassThru.dll
EFI1.1\Binary\NT32\BIN\AtapiPassThru.efi
EFI1.1\Binary\NT32\BIN\attrib.dll
EFI1.1\Binary\NT32\BIN\attrib.efi
EFI1.1\Binary\NT32\BIN\bcfg.dll
EFI1.1\Binary\NT32\BIN\bcfg.efi
EFI1.1\Binary\NT32\BIN\bis.dll
EFI1.1\Binary\NT32\BIN\bis.efi
EFI1.1\Binary\NT32\BIN\bmaint.dll
EFI1.1\Binary\NT32\BIN\bmaint.efi
EFI1.1\Binary\NT32\BIN\bootmgr.dll
EFI1.1\Binary\NT32\BIN\bootmgr.efi

EFI1.1\Binary\NT32\BIN\cls.dll
EFI1.1\Binary\NT32\BIN\cls.efi
EFI1.1\Binary\NT32\BIN\comp.dll
EFI1.1\Binary\NT32\BIN\comp.efi
EFI1.1\Binary\NT32\BIN\ConPlatform.dll
EFI1.1\Binary\NT32\BIN\ConPlatform.efi
EFI1.1\Binary\NT32\BIN\ConSplitter.dll
EFI1.1\Binary\NT32\BIN\ConSplitter.efi
EFI1.1\Binary\NT32\BIN\corefile
EFI1.1\Binary\NT32\BIN\cp.dll
EFI1.1\Binary\NT32\BIN\cp.efi
EFI1.1\Binary\NT32\BIN\date.dll
EFI1.1\Binary\NT32\BIN\date.efi
EFI1.1\Binary\NT32\BIN\dblk.dll
EFI1.1\Binary\NT32\BIN\dblk.efi
EFI1.1\Binary\NT32\BIN\DebugPort.dll
EFI1.1\Binary\NT32\BIN\DebugPort.efi
EFI1.1\Binary\NT32\BIN\DebugSupport.dll
EFI1.1\Binary\NT32\BIN\DebugSupport.efi
EFI1.1\Binary\NT32\BIN\Decompress.dll
EFI1.1\Binary\NT32\BIN\Decompress.efi
EFI1.1\Binary\NT32\BIN\DiskIo.dll
EFI1.1\Binary\NT32\BIN\DiskIo.efi
EFI1.1\Binary\NT32\BIN\dmem.dll
EFI1.1\Binary\NT32\BIN\dmem.efi
EFI1.1\Binary\NT32\BIN\dmpstore.dll
EFI1.1\Binary\NT32\BIN\dmpstore.efi
EFI1.1\Binary\NT32\BIN\dumpbs.dll
EFI1.1\Binary\NT32\BIN\dumpbs.efi
EFI1.1\Binary\NT32\BIN\EBC.dll
EFI1.1\Binary\NT32\BIN\EBC.efi
EFI1.1\Binary\NT32\BIN\edit.dll
EFI1.1\Binary\NT32\BIN\edit.efi
EFI1.1\Binary\NT32\BIN\EfiCompress.dll
EFI1.1\Binary\NT32\BIN\EfiCompress.efi
EFI1.1\Binary\NT32\BIN\EfiDecompress.dll

EFI1.1\Binary\NT32\BIN\EfiDecompress.efi
EFI1.1\Binary\NT32\BIN\err.dll
EFI1.1\Binary\NT32\BIN\err.efi
EFI1.1\Binary\NT32\BIN\Fat.dll
EFI1.1\Binary\NT32\BIN\Fat.efi
EFI1.1\Binary\NT32\BIN\getmtc.dll
EFI1.1\Binary\NT32\BIN\getmtc.efi
EFI1.1\Binary\NT32\BIN\GraphicsConsole.dll
EFI1.1\Binary\NT32\BIN\GraphicsConsole.efi
EFI1.1\Binary\NT32\BIN\hexedit.dll
EFI1.1\Binary\NT32\BIN\hexedit.efi
EFI1.1\Binary\NT32\BIN\load.dll
EFI1.1\Binary\NT32\BIN\load.efi
EFI1.1\Binary\NT32\BIN\LoadBmp.dll
EFI1.1\Binary\NT32\BIN\LoadBmp.efi
EFI1.1\Binary\NT32\BIN\LoadPciRom.dll
EFI1.1\Binary\NT32\BIN\LoadPciRom.efi
EFI1.1\Binary\NT32\BIN\ls.dll
EFI1.1\Binary\NT32\BIN\ls.efi
EFI1.1\Binary\NT32\BIN\memmap.dll
EFI1.1\Binary\NT32\BIN\memmap.efi
EFI1.1\Binary\NT32\BIN\mkdir.dll
EFI1.1\Binary\NT32\BIN\mkdir.efi
EFI1.1\Binary\NT32\BIN\mm.dll
EFI1.1\Binary\NT32\BIN\mm.efi
EFI1.1\Binary\NT32\BIN\mode.dll
EFI1.1\Binary\NT32\BIN\mode.efi
EFI1.1\Binary\NT32\BIN\mv.dll
EFI1.1\Binary\NT32\BIN\mv.efi
EFI1.1\Binary\NT32\BIN\nshell.dll
EFI1.1\Binary\NT32\BIN\nshell.efi
EFI1.1\Binary\NT32\BIN\nt32.exe
EFI1.1\Binary\NT32\BIN\NV1File
EFI1.1\Binary\NT32\BIN\Partition.dll
EFI1.1\Binary\NT32\BIN\Partition.efi
EFI1.1\Binary\NT32\BIN\pci.dll

EFI1.1\Binary\NT32\BIN\pci.efi
EFI1.1\Binary\NT32\BIN\PciBus.dll
EFI1.1\Binary\NT32\BIN\PciBus.efi
EFI1.1\Binary\NT32\BIN\PxeBc.dll
EFI1.1\Binary\NT32\BIN\PxeBc.efi
EFI1.1\Binary\NT32\BIN\reset.dll
EFI1.1\Binary\NT32\BIN\reset.efi
EFI1.1\Binary\NT32\BIN\rm.dll
EFI1.1\Binary\NT32\BIN\rm.efi
EFI1.1\Binary\NT32\BIN\SerialMouse.dll
EFI1.1\Binary\NT32\BIN\SerialMouse.efi
EFI1.1\Binary\NT32\BIN\setsize.dll
EFI1.1\Binary\NT32\BIN\setsize.efi
EFI1.1\Binary\NT32\BIN\shellenv.dll
EFI1.1\Binary\NT32\BIN\shellenv.efi
EFI1.1\Binary\NT32\BIN\Snp3264.dll
EFI1.1\Binary\NT32\BIN\Snp3264.efi
EFI1.1\Binary\NT32\BIN\stall.dll
EFI1.1\Binary\NT32\BIN\stall.efi
EFI1.1\Binary\NT32\BIN\Terminal.dll
EFI1.1\Binary\NT32\BIN\Terminal.efi
EFI1.1\Binary\NT32\BIN\time.dll
EFI1.1\Binary\NT32\BIN\time.efi
EFI1.1\Binary\NT32\BIN\touch.dll
EFI1.1\Binary\NT32\BIN\touch.efi
EFI1.1\Binary\NT32\BIN\type.dll
EFI1.1\Binary\NT32\BIN\type.efi
EFI1.1\Binary\NT32\BIN\Undi.dll
EFI1.1\Binary\NT32\BIN\Undi.efi
EFI1.1\Binary\NT32\BIN\ver.dll
EFI1.1\Binary\NT32\BIN\ver.efi
EFI1.1\Binary\NT32\BIN\vol.dll
EFI1.1\Binary\NT32\BIN\vol.efi
EFI1.1\Binary\NT32\BIN\WinNtBlockIo.dll
EFI1.1\Binary\NT32\BIN\WinNtBlockIo.efi
EFI1.1\Binary\NT32\BIN\WinNtBusDriver.dll

EFI1.1\Binary\NT32\BIN\WinNtBusDriver.efi
EFI1.1\Binary\NT32\BIN\WinNtConsole.dll
EFI1.1\Binary\NT32\BIN\WinNtConsole.efi
EFI1.1\Binary\NT32\BIN\WinNtPciRootBridge.dll
EFI1.1\Binary\NT32\BIN\WinNtPciRootBridge.efi
EFI1.1\Binary\NT32\BIN\WinNtSerialIo.dll
EFI1.1\Binary\NT32\BIN\WinNtSerialIo.efi
EFI1.1\Binary\NT32\BIN\WinNtSimpleFileSystem.dll
EFI1.1\Binary\NT32\BIN\WinNtSimpleFileSystem.efi
EFI1.1\Binary\NT32\BIN\WinNtUga.dll
EFI1.1\Binary\NT32\BIN\WinNtUga.efi
EFI1.1\Binary\Pro1000Undi\readme.txt
EFI1.1\Binary\Pro1000Undi\IA32\e216e2.efi
EFI1.1\Binary\Pro1000Undi\Itanium\e216i2.efi
EFI1.1\Binary\Pro1000Undi\RomImage\IA32\BA216E2.NIC
EFI1.1\Binary\Pro1000Undi\RomImage\Itanium\BA216I2.NIC
EFI1.1\Binary\SAL64\BIN\bis.efi
EFI1.1\Binary\SAL64\BIN\bmaint.efi
EFI1.1\Binary\SAL64\BIN\bootmgr.efi
EFI1.1\Binary\SAL64\BIN\ConPlatform.efi
EFI1.1\Binary\SAL64\BIN\ConSplitter.efi
EFI1.1\Binary\SAL64\BIN\DebugPort.efi
EFI1.1\Binary\SAL64\BIN\DebugSupport.efi
EFI1.1\Binary\SAL64\BIN\Decompress.efi
EFI1.1\Binary\SAL64\BIN\DiskIo.efi
EFI1.1\Binary\SAL64\BIN\EBC.efi
EFI1.1\Binary\SAL64\BIN\Fat.efi
EFI1.1\Binary\SAL64\BIN\GraphicsConsole.efi
EFI1.1\Binary\SAL64\BIN\IsaBus.efi
EFI1.1\Binary\SAL64\BIN\IsaSerial.efi
EFI1.1\Binary\SAL64\BIN\Partition.efi
EFI1.1\Binary\SAL64\BIN\PcatIsaAcpi.efi
EFI1.1\Binary\SAL64\BIN\PcatPciRootBridge.efi
EFI1.1\Binary\SAL64\BIN\PciBus.efi
EFI1.1\Binary\SAL64\BIN\PxeBc.efi
EFI1.1\Binary\SAL64\BIN\PxeDhcp4.efi

EFI1.1\Binary\SAL64\BIN\SerialMouse.efi
EFI1.1\Binary\SAL64\BIN\Snp3264.efi
EFI1.1\Binary\SAL64\BIN\Terminal.efi
EFI1.1\Binary\SAL64\BIN\Undi.efi
EFI1.1\Binary\SAL64\BIN\VgaClass.efi
EFI1.1\Binary\SAL64\SHELLItanium\Shell.efi
EFI1.1\BUILD\nt32\system.cmd
EFI1.1\BUILD\build-common.cmd
EFI1.1\BUILD\build-common-ia32.cmd
EFI1.1\BUILD\build-common-ia64.cmd
EFI1.1\BUILD\makefile.common
EFI1.1\BUILD\tools\src\dskimage\dskimage.c
EFI1.1\COREFW\fw\platform\BuildTip\BootSector\bs16.asm
EFI1.1\COREFW\fw\platform\BuildTip\BootSector\bs32.asm
EFI1.1\COREFW\fw\platform\BuildTip\BootSector\start16.asm
EFI1.1\COREFW\fw\platform\BuildTip\BootSector\start32.asm
EFI1.1\EDK\Protocol\Ps2Policy\Ps2Policy.c
EFI1.1\EDK\Protocol\Ps2Policy\Ps2Policy.h
EFI1.1\NOTES\EFI1.1ShellCommands.pdf
EFI1.1\NOTES\EFI_LIB.pdf
EFI1.1\NOTES\EfiDriverLib.pdf
EFI1.1\NOTES\RELNOTE.pdf
EFI1.1\NOTES\SalEfiHandOffState.pdf
EFI1.1\NOTES\ScsiDriverModel.pdf

B.1.4 List of Modified Files

EFI1.1\BUILD\bios32\build.cmd
EFI1.1\BUILD\bios32\makefile
EFI1.1\BUILD\bios32\master.env
EFI1.1\BUILD\EbcDrivers\build.cmd
EFI1.1\BUILD\EbcDrivers\makefile
EFI1.1\BUILD\EbcDrivers\master.env
EFI1.1\BUILD\Ia32Drivers\build.cmd
EFI1.1\BUILD\Ia32Drivers\makefile
EFI1.1\BUILD\Ia32Drivers\master.env
EFI1.1\BUILD\ia-32emb\build.cmd

EFI1.1\BUILD\ia-32emb\makefile
EFI1.1\BUILD\ia-32emb\master.env
EFI1.1\BUILD\IpfDrivers\build.cmd
EFI1.1\BUILD\IpfDrivers\makefile
EFI1.1\BUILD\IpfDrivers\master.env
EFI1.1\BUILD\nt32\build.cmd
EFI1.1\BUILD\nt32\makefile
EFI1.1\BUILD\nt32\master.env
EFI1.1\BUILD\sal64\build.cmd
EFI1.1\BUILD\sal64\makefile
EFI1.1\BUILD\sal64\master.env
EFI1.1\BUILD\tools\bin\col.exe
EFI1.1\BUILD\tools\bin\EFIBOOT.IMG
EFI1.1\BUILD\tools\bin\eficompress.exe
EFI1.1\BUILD\tools\bin\efildrimage.exe
EFI1.1\BUILD\tools\bin\efirom.exe
EFI1.1\BUILD\tools\bin\fwimage.exe
EFI1.1\BUILD\tools\bin\genhelp.exe
EFI1.1\BUILD\tools\bin\genmake.exe
EFI1.1\BUILD\tools\bin\splitfile.exe
EFI1.1\BUILD\tools\bin\vccheck.exe
EFI1.1\BUILD\tools\src\genmake\genmake.c
EFI1.1\BUILD\tools\src\makefile
EFI1.1\BUILD\clean.cmd
EFI1.1\COREFW\fw\efi\crc\crc.c
EFI1.1\COREFW\fw\efi\exec\event.c
EFI1.1\COREFW\fw\efi\exec\mtc.c
EFI1.1\COREFW\fw\efi\hand\handle.c
EFI1.1\COREFW\fw\efi\hand\locate.c
EFI1.1\COREFW\fw\efi\loader\load.c
EFI1.1\COREFW\fw\efi\loader\pe.c
EFI1.1\COREFW\fw\efi\mem\page.c
EFI1.1\COREFW\fw\efi\mem\pool.c
EFI1.1\COREFW\fw\efi\mem\virt.c
EFI1.1\COREFW\fw\platform\bootmgr\Maint\boot.c
EFI1.1\COREFW\fw\platform\bootmgr\Maint\console.c

EFI1.1\COREFW\fw\platform\bootmgr\Maint\menu.h
EFI1.1\COREFW\fw\platform\BuildTip\bios32\efildr.c
EFI1.1\COREFW\fw\platform\BuildTip\bios32\init.c
EFI1.1\COREFW\fw\platform\BuildTip\BootSector\bootsect.asm
EFI1.1\COREFW\fw\platform\BuildTip\IA-32Emb\efildr.c
EFI1.1\COREFW\fw\platform\BuildTip\IA-32Emb\init.c
EFI1.1\COREFW\fw\platform\BuildTip\Inc\Drivers.h
EFI1.1\COREFW\fw\platform\BuildTip\nt32\init.c
EFI1.1\COREFW\fw\platform\BuildTip\nt32\misc.c
EFI1.1\COREFW\fw\platform\BuildTip\nt32\ntemul.h
EFI1.1\COREFW\fw\platform\BuildTip\Sal64\init.c
EFI1.1\COREFW\fw\platform\drivers\biosint\BiosSnp16\BiosSnp16.c
EFI1.1\COREFW\fw\platform\drivers\biosint\BiosSnp16\misc.c
EFI1.1\COREFW\fw\platform\drivers\biosint\BiosVga\BiosVga.c
EFI1.1\COREFW\fw\platform\drivers\biosint\disk\biosint13.c
EFI1.1\COREFW\fw\platform\PlDriver\Defio\IA32\definit.c
EFI1.1\COREFW\fw\platform\PlDriver\Defio\Ipf\definit.c
EFI1.1\COREFW\fw\platform\PlDriver\Defio\Pci\pci.c
EFI1.1\COREFW\fw\platform\PlDriver\efildr\PlEfiLdr.c
EFI1.1\COREFW\fw\platform\PlDriver\NullDefio\IA32\definit.c
EFI1.1\COREFW\fw\platform\PlDriver\NullDefio\Pci\pci.c
EFI1.1\COREFW\fw\platform\PlDriver\Time\DALLAS1287\time.c
EFI1.1\COREFW\fw\platform\PlDriver\Unicode\English\unicode.c
EFI1.1\EDK\Drivers\AtapiPassThru\AtapiPassThru.c
EFI1.1\EDK\Drivers\CirrusLogic5430\CirrusLogic5430UgaDraw.c
EFI1.1\EDK\Drivers\Console\ConSplitter\ConSplitter.c
EFI1.1\EDK\Drivers\Console\ConSplitter\ConSplitter.h
EFI1.1\EDK\Drivers\Console\ConSplitter\ConSplitterGraphics.c
EFI1.1\EDK\Drivers\Console\GraphicsConsole\GraphicsConsole.c
EFI1.1\EDK\Drivers\Console\Terminal\TerminalConIn.c
EFI1.1\EDK\Drivers\Console\Terminal\TerminalConOut.c
EFI1.1\EDK\Drivers\DebugPort\DebugPort.c
EFI1.1\EDK\Drivers\DiskIo\diskio.c
EFI1.1\EDK\Drivers\DiskIo\diskio.h
EFI1.1\EDK\Drivers\ebc\Ipf\EbcLowLevel.s
EFI1.1\EDK\Drivers\ebc\Ipf\EbcSupport.c

EFI1.1\EDK\Drivers\ebc\EbcExecute.c
EFI1.1\EDK\Drivers\ebc\EbcInt.c
EFI1.1\EDK\Drivers\FileSystem\Fat\cache.c
EFI1.1\EDK\Drivers\FileSystem\Fat\ComponentName.c
EFI1.1\EDK\Drivers\FileSystem\Fat\data.c
EFI1.1\EDK\Drivers\FileSystem\Fat\delete.c
EFI1.1\EDK\Drivers\FileSystem\Fat\dirent.c
EFI1.1\EDK\Drivers\FileSystem\Fat\dirsup.c
EFI1.1\EDK\Drivers\FileSystem\Fat\Fat.c
EFI1.1\EDK\Drivers\FileSystem\Fat\fat.h
EFI1.1\EDK\Drivers\FileSystem\Fat\flush.c
EFI1.1\EDK\Drivers\FileSystem\Fat\fname.c
EFI1.1\EDK\Drivers\FileSystem\Fat\fspace.c
EFI1.1\EDK\Drivers\FileSystem\Fat\hash.c
EFI1.1\EDK\Drivers\FileSystem\Fat\info.c
EFI1.1\EDK\Drivers\FileSystem\Fat\init.c
EFI1.1\EDK\Drivers\FileSystem\Fat\misc.c
EFI1.1\EDK\Drivers\FileSystem\Fat\open.c
EFI1.1\EDK\Drivers\FileSystem\Fat\openvol.c
EFI1.1\EDK\Drivers\FileSystem\Fat\rw.c
EFI1.1\EDK\Drivers\ide\ata.c
EFI1.1\EDK\Drivers\ide\atapi.c
EFI1.1\EDK\Drivers\ide\DriverConfiguration.c
EFI1.1\EDK\Drivers\ide\DriverDiagnostics.c
EFI1.1\EDK\Drivers\ide\ide.c
EFI1.1\EDK\Drivers\ide\idebus.c
EFI1.1\EDK\Drivers\ide\make.inf
EFI1.1\EDK\Drivers\IsaBus\IsaBus.c
EFI1.1\EDK\Drivers\IsaBus\IsaIo.c
EFI1.1\EDK\Drivers\IsaFloppy\IsaFloppy.c
EFI1.1\EDK\Drivers\IsaFloppy\IsaFloppyBlock.c
EFI1.1\EDK\Drivers\IsaFloppy\IsaFloppyCtrl.c
EFI1.1\EDK\Drivers\Isaserial\serial.c
EFI1.1\EDK\Drivers\Isaserial\serial.h
EFI1.1\EDK\Drivers\Partition\ElTorito.c
EFI1.1\EDK\Drivers\Partition\ElTorito.h

EFI1.1\EDK\Drivers\Partition\Gpt.c
EFI1.1\EDK\Drivers\Partition\Mbr.c
EFI1.1\EDK\Drivers\Partition\Partition.c
EFI1.1\EDK\Drivers\PcatIsaAcpi\IsaAcpi.c
EFI1.1\EDK\Drivers\PcatIsaAcpi\PcatIsaAcpi.h
EFI1.1\EDK\Drivers\PcatPciRootBridge\Ia32\PcatIo.c
EFI1.1\EDK\Drivers\PcatPciRootBridge\Ipf\PcatIo.c
EFI1.1\EDK\Drivers\PcatPciRootBridge\PcatPciRootBridge.c
EFI1.1\EDK\Drivers\PcatPciRootBridge\PcatPciRootBridge.h
EFI1.1\EDK\Drivers\PcatPciRootBridge\PcatPciRootBridgeIo.c
EFI1.1\EDK\Drivers\pcibus\pcibus.h
EFI1.1\EDK\Drivers\pcibus\PciDriverOverride.c
EFI1.1\EDK\Drivers\pcibus\PciEnumeratorSupport.c
EFI1.1\EDK\Drivers\pcibus\PciIo.c
EFI1.1\EDK\Drivers\Ps2Keyboard\Ps2KbdCtrller.c
EFI1.1\EDK\Drivers\Ps2Keyboard\Ps2KbdTextIn.c
EFI1.1\EDK\Drivers\Ps2Keyboard\Ps2Keyboard.c
EFI1.1\EDK\Drivers\Ps2Keyboard\Ps2Keyboard.h
EFI1.1\EDK\Drivers\Ps2Mouse\CommPs2.c
EFI1.1\EDK\Drivers\Ps2Mouse\Ps2Mouse.c
EFI1.1\EDK\Drivers\Ps2Mouse\Ps2Mouse.h
EFI1.1\EDK\Drivers\PxeBc\bc.c
EFI1.1\EDK\Drivers\PxeBc\bc.h
EFI1.1\EDK\Drivers\PxeBc\ip.h
EFI1.1\EDK\Drivers\PxeBc\pxe_bc_dhcp.c
EFI1.1\EDK\Drivers\PxeBc\pxe_bc_mtftp.c
EFI1.1\EDK\Drivers\PxeBc\pxe_bc_tcp.c
EFI1.1\EDK\Drivers\PxeBc\pxe_bc_udp.c
EFI1.1\EDK\Drivers\ScsiDisk\ScsiDisk.c
EFI1.1\EDK\Drivers\SerialMouse\SerialMouse.c
EFI1.1\EDK\Drivers\SerialMouse\SerialMouse.h
EFI1.1\EDK\Drivers\Snp32_64\callback.c
EFI1.1\EDK\Drivers\Snp32_64\statistics.c
EFI1.1\EDK\Drivers\Undi\E100B.H
EFI1.1\EDK\Drivers\Undi\INIT.C
EFI1.1\EDK\Drivers\Usb\Uhci\uhci.c

EFI1.1\EDK\Drivers\Usb\UsbMassStorage\UsbMassStorage.c
EFI1.1\EDK\Drivers\Usb\UsbMouse\mousehid.c
EFI1.1\EDK\Drivers\Usb\UsbMouse\usbmouse.c
EFI1.1\EDK\Drivers\VgaClass\VgaClass.c
EFI1.1\EDK\Drivers\VgaClass\VgaClass.h
EFI1.1\EDK\Drivers\WinNtThunk\BlockIo\WinNtBlockIo.c
EFI1.1\EDK\Drivers\WinNtThunk\Console\ConsoleOut.c
EFI1.1\EDK\Drivers\WinNtThunk\SimpleFileSystem\WinNtSimpleFileSystem.c
EFI1.1\EDK\Drivers\WinNtThunk\Uga\WinNtUgaScreen.c
EFI1.1\EDK\Drivers\WinNtThunk\WinNtPciRootBridge\WinNtPciRootBridgeIo.c
EFI1.1\EDK\Include\efi.h
EFI1.1\EDK\Include\EfiDevicePath.h
EFI1.1\EDK\Include\pci22.h
EFI1.1\EDK\Lib\EfiCommonLib\Ebc\EbcMath.c
EFI1.1\EDK\Lib\EfiCommonLib\Ia32\EfiCopyMem.asm
EFI1.1\EDK\Lib\EfiCommonLib\Ia32\EfiSetMem.asm
EFI1.1\EDK\Lib\EfiCommonLib\Ia32\EfiZeroMem.asm
EFI1.1\EDK\Lib\EfiCommonLib\Ia32\Math.c
EFI1.1\EDK\Lib\EfiCommonLib\Ipf\math.c
EFI1.1\EDK\Lib\EfiCommonLib\String.c
EFI1.1\EDK\Lib\Include\EfiCommonLib.h
EFI1.1\EDK\Lib\Include\EfiDriverLib.h
EFI1.1\EDK\Lib\Include\EfiPrintLib.h
EFI1.1\EDK\Lib\Print\BoxDraw.c
EFI1.1\EDK\Protocol\ConsoleControl\ConsoleControl.h
EFI1.1\EDK\Protocol\LoadedImage\LoadedImage.h
EFI1.1\EDK\Protocol\make.inf
EFI1.1\INC\efi.h
EFI1.1\INC\efiapi.h
EFI1.1\INC\efilib.h
EFI1.1\LIB\systable.c
EFI1.1\NOTES\ISO-639-2.TXT
EFI1.1\SHELL\edit\libEditor.c
EFI1.1\SHELL\edit\libFileBuffer.c
EFI1.1\SHELL\hexedit\libEditor.c
EFI1.1\SHELL\hexedit\libfileimage.c

EFI1.1\SHELL\hexedit\main.c

EFI1.1\SHELL\iomod\make.inf

EFI1.1\SHELL\lib\init.c

EFI1.1\SHELL\mem\make.inf

EFI1.1\SHELL\pci\pci.c

EFI1.1\SHELL\shellenv\exec.c

EFI1.1\SHELL\shellenv\marg.c

EFI1.1\SHELL\shellenv\protid.c

EFI1.1\SHELL\stall\stall.c

EFI1.1\SHELL\ver\ver.c

B.2 EFI 1.10.14.61

B.2.1 Core Bug Fixes

1. Updated version to EFI 1.10.14.61.
2. Updated the BIOS32 boot floppy image to EFI 1.10.14.61.
3. Updated the release notes to EFI 1.10.14.61.
4. Updated the EFI Memory Manager to support the memory types in the range 0x80000000-0xffffffff that are available to OS loader. This change is required to follow the final draft of the EFI 1.10 Specification.
5. Updated the EFI Memory Manager for the Itanium processor family to require the memory of types EfiRuntimeServicesData, EfiRuntimeServicesCode, EfiAcpiNVS, and EfiAcpiReclaim to have their base address and lengths 8 KB aligned. This change is required by the final draft of the EFI 1.10 Specification.
6. Updated ExitBootServices() to verify that the EFI Memory Map follows the construction rules defined in the final draft of the EFI 1.10 Specification.
7. Updated the SAL64 build tip to make sure the memory map passed up from the SAL follows the construction rules defined in the final draft of the EFI 1.10 Specification. If it doesn't, then a best attempt is made to fix the memory map before it is handed to the EFI Memory Manager.
8. Updated the EFI Boot Service CalculateCrc32() to follow the final draft of the EFI 1.10 Specification.
9. Fixed bug in Event Services that now allows events to be created with a TPL level of TPL_HIGH_LEVEL.
10. Added additional parameter checking to the EFI Runtime Service GetNextHighMonotonicCount()
11. Reduced the size of the EFI Boot Service ConnectController().
12. Fixed a bug in the EFI Boot Service ConnectController() that would overrun a handle buffer if a duplicate handle was present in the context override handle list, the Platform Driver Override handle list, or the Bus Specific Driver Override handle list.
13. Fixed a bug in EFI Boot Service ConnectController(). If a recursive connect was being performed, it was returning the status of the connect operation on the last child device instead of the status of the connect operation that was performed in the parent controller.

14. Made sure that ChildHandle is a valid EFI_HANDLE in the EFI Boot Service DisconnectController() if ChildHandle is not NULL.
15. Updated the field names of type EFI_OPEN_PROTOCOL_INFORMATION_ENTRY to match the final draft of the EFI 1.10 Specification.
16. Added a Key field to the internal handle database data structures to track the when a handle was created or most recently modified. This feature is used by StartImage() to automatically connect and handles that were created or modified when an EFI image is executed. This change is required to follow the final draft of the EFI 1.10 Specification.
17. Added more parameter checking to the EFI Boot Service InstallProtocolInterface().
18. Fixed a bug in the EFI Boot Service OpenProtocol() that would return EFI_SUCESS if the same agent opened the same controller exclusively more than once. It should return EFI_ACCESS_DENIED if this is attempted more than once.
19. Removed all references to the EFI Boot Service PCHandleProtocol() to match the final draft of the EFI 1.10 Specification.
20. Fixed a bug in the EFI Boot Service OpenProtocolInformation(). It used to return EFI_NOT_FOUND if no agents had opened a protocol on a handle. Instead, it should return EFI_SUCESS and an EntryCount of 0.
21. Fixed a bug in the EFI Boot Service InstallConfigurationTable() that was no managing the value ST->NumberOfTableEntries correctly of this service ran out of system memory when a new table was allocated.
22. Fixed a bug in the EFI Boot Service LoadImage() that was not correctly checking the parameter ParentImageHandle.
23. Simplified the logic in the EFI Boot Service UnloadImage() and fixed a bug that would incorrectly free the thunks of an EBC image even though the unload operation fails.
24. Added more parameter checking to the EFI Boot Service GetMemoryMap().
25. Added more parameter checking to the EFI Runtime Service ConvertPointer().
26. Added more parameter checking to the EFI Runtime Service GetVariable().
27. Added more parameter checking to the EFI Runtime Service GetTime().
28. Added more parameter checking to the EFI Runtime Service GetWakeupTime().
29. Updated parameter checking in the ReadBlocks() and WriteBlocks() services of the Block I/O drivers.
30. Fixed a bug with the scope of the retry counters in the BIOS INT 13 Block I/O driver.
31. Added MEMORY_FENCE() operations to the Device I/O Protocol in the Map(), UnMap(), and Flush() services.
32. Added a Simple Pointer splitter to the Console Splitter driver.
33. Implemented a more intelligent mode intersection algorithm in the Console Splitter driver.
34. Updated the Graphics Console driver to support up to 3 text modes. 80x25, 80x50, and max rows and columns for the given resolution.
35. Updated the Graphics Console driver to always clear the entire display when the text display is cleared.
36. Updated the Terminal driver to always support the input key mapping for PCANSI, VT-100, VT-100+, and VTUTF8. None of the input key mappings from these terminal types overlap, so they can all be active at the same time.

37. Updated the Terminal driver to allow manual entry of ESC sequences. This is required to follow the specification “Standardizing Out-of-Band Management Console Output and Terminal Emulation (VT-UTF8 and VT100+).”
38. Added more parameter checking to the ReadDisk() and WriteDisk() services of the Disk I/O Protocol.
39. Added MEMORY_FENCE() operations to the EBC interpreter to guarantee that all memory operations performed by the EBC interpreter are strongly ordered. This is required to follow the final draft of the EFI 1.10 Specification.
40. Fixed a bug in the Delete() service of the FAT driver. On an error, it was not returning EFI_WARN_DELETE_FAILURE.
41. Fixed buffer overrun bug in the GetInfo() service of the FAT driver.
42. Fixed a specification violation in the FAT driver with numeric-tail generation algorithm.
43. Added resource range checking to the ISA I/O Protocol
44. Added more parameter checking to the Map() service of the ISA I/O Protocol.
45. Reduced the size of the ISA Floppy driver.
46. Fixed logic bug in the GetControl() service of the ISA Serial driver.
47. Added more parameter checking to the PCI Root Bridge I/O Protocol
48. Added MEMORY_FENCE() operations to the Map(), UnMap(), and Flush() service of the PCI Root Bridge I/O Protocol.
49. Added support for DAC capable device and 64-bit DMA to the PCI Root Bridge I/O Protocol.
50. Updated the PCI Option ROM extraction code to preserve all the I/O and BM command register bits while the Option ROMs are retrieved.
51. Added support for DAC capable devices and 64-bit DMA to the PCI Bus driver.
52. Fixed error return codes from several PCI I/O Protocol services.
53. Updated the GetBarAttributes() service of the PCI I/O Protocol to return the ACPI resource descriptor for the requested BAR.
54. Updated PXE Base Code driver to use the same coding conventions as the other drivers.
55. Updated the SCSI Bus Driver to only bind to logical SCSI Pass Thru interfaces.
56. Many bug fixes in all of the USB drivers.
57. Fixed a hang in the USB Bus driver when a USB speaker was attached.
58. Fixed bugs in the Write(), SetInfo(), and Delete() services of the WinNT Simple File System driver.
59. Cleaned up threads and handles when the WinNT UGA driver is stopped.
60. Updated the InfiniBand* Device Path node to follow final draft of the EFI 1.10 Specification.
61. Added support for Visual Studio .NET 2002 (VC7) debug directory entries.
62. Updated all Shell command to use a centralized manager for page breaks.
63. Changed BCFG to always use HEX parameters.
64. Updated the EFI Shell command DH to display PCI BARs when a PCI I/O Protocol is dumped.
65. Updated the EFI Shell command DH to display the PDB file name for a loaded image.
66. Changed the EFI Shell command MAP to use volatile variables.
67. Fixed MAP command to show consistent drive mappings when removable media is removed.
68. Updated the EFI Shell command TIME to display correct ranges for house, minutes, and seconds.

69. Updated the EFI Shell command DrvCfg to allow the driver to set the options for all the devices the driver is managing at once.
70. Made minor corrections to the EFI Shell help files.
71. Added EFI_PNP_ID() macro to match final draft of the EFI 1.10 Specification.
72. Updated the declaration of the Unicode Collation Protocol to match the final draft of the EFI 1.10 Specification.

B.2.2 List of Deleted Files

efi1.1\edk\drivers\console\terminal\pcansi.c

efi1.1\edk\drivers\console\terminal\vt100.c

efi1.1\edk\drivers\console\terminal\vt100plus.c

efi1.1\edk\drivers\pxebc\efibis.h

B.2.3 List of New Files

efi1.1\edk\drivers\console\terminal\ansi.c

efi1.1\edk\drivers\snp32_64\waitforpacket.c

efi1.1\edk\protocol\ebcdebughelper\ebcdebughelper.c

efi1.1\edk\protocol\ebcdebughelper\ebcdebughelper.h

efi1.1\edk\protocol\simplefilesystem\fileinfo.c

efi1.1\edk\protocol\simplefilesystem\fileinfoid.c

efi1.1\edk\protocol\simplefilesystem\filesystemvolumelabelinfo.c

B.2.4 List of Modified Files

efi1.1\build\ia32drivers\makefile

efi1.1\build\ipfdrivers\makefile

efi1.1\build\ipfdrivers\master.env

efi1.1\build\sal64\master.env

efi1.1\build\tools\bin\efiboot.img

efi1.1\build\tools\bin\vccheck.exe

efi1.1\corefw\fw\efi\crc\crc.c

efi1.1\corefw\fw\efi\debugimageinfo\debugimageinfo.c

efi1.1\corefw\fw\efi\exec\event.c

efi1.1\corefw\fw\efi\exec\mtc.c

efi1.1\corefw\fw\efi\hand\driversupport.c

efi1.1\corefw\fw\efi\hand\hand.h

efi1.1\corefw\fw\efi\hand\handle.c

efi1.1\corefw\fw\efi\hand\notify.c

efi1.1\corefw\fw\efi\inc\efifw.h

efi1.1\corefw\fw\efi\init\bsdata.c
efi1.1\corefw\fw\efi\init\init.c
efi1.1\corefw\fw\efi\loader\load.c
efi1.1\corefw\fw\efi\loader\pe.c
efi1.1\corefw\fw\efi\mem\imem.h
efi1.1\corefw\fw\efi\mem\page.c
efi1.1\corefw\fw\efi\mem\pool.c
efi1.1\corefw\fw\efi\mem\virt.c
efi1.1\corefw\fw\efi\variable\varapi.c
efi1.1\corefw\fw\platform\bootmgr\maint\init.c
efi1.1\corefw\fw\platform\buildtip\nt32\misc.c
efi1.1\corefw\fw\platform\buildtip\nt32\winntthunk.c
efi1.1\corefw\fw\platform\buildtip\sal64\init.c
efi1.1\corefw\fw\platform\drivers\biosint\disk\biosint13.c
efi1.1\corefw\fw\platform\pldriver\defio\ia32\definit.c
efi1.1\corefw\fw\platform\pldriver\nulldefio\ia32\definit.c
efi1.1\corefw\fw\platform\pldriver\time\dallas1287\time.c
efi1.1\edk\drivers\bis\basecode\util\efibis_persist.c
efi1.1\edk\drivers\bis\inc\bis_priv.h
efi1.1\edk\drivers\bis\inc\buildnbr.h
efi1.1\edk\drivers\bis\inc\defaultmemmgr.h
efi1.1\edk\drivers\bis\oasis\inc\codefile.h
efi1.1\edk\drivers\bis\oasis\inc\cssm_codefile.h
efi1.1\edk\drivers\bis\oasis\inc\sei.h
efi1.1\edk\drivers\bis\oasis\src\fwk\util\ber_der\r1_1\inc\ber_internal.h
efi1.1\edk\drivers\bis\oasis\src\integrity\inc\isl_parse.h
efi1.1\edk\drivers\bis\oasis\src\integrity\inc\isl_tags.h
efi1.1\edk\drivers\bis\oasis\src\integrity\inc\muttags.h
efi1.1\edk\drivers\bis\oasis\src\integrity\inc\pkapi.h
efi1.1\edk\drivers\bis\oasis\src\integrity\inc\pkdef.h
efi1.1\edk\drivers\bis\oasis\src\integrity\src\eisl\pk.c
efi1.1\edk\drivers\bis\oasis\src\integrity\src\isl\isl_memory.c
efi1.1\edk\drivers\console\consplitter\componentname.c
efi1.1\edk\drivers\console\consplitter\consplitter.c
efi1.1\edk\drivers\console\consplitter\consplitter.h
efi1.1\edk\drivers\console\consplitter\consplittergraphics.c

efi1.1\edk\drivers\console\graphicsconsole\graphicsconsole.c
efi1.1\edk\drivers\console\graphicsconsole\graphicsconsole.h
efi1.1\edk\drivers\console\terminal\make.inf
efi1.1\edk\drivers\console\terminal\terminal.c
efi1.1\edk\drivers\console\terminal\terminal.h
efi1.1\edk\drivers\console\terminal\terminalconin.c
efi1.1\edk\drivers\console\terminal\terminalconout.c
efi1.1\edk\drivers\console\terminal\vtutf8.c
efi1.1\edk\drivers\diskio\diskio.c
efi1.1\edk\drivers\ebc\ebcexecute.c
efi1.1\edk\drivers\ebc\ebcint.c
efi1.1\edk\drivers\filesystem\fat\delete.c
efi1.1\edk\drivers\filesystem\fat\fat.h
efi1.1\edk\drivers\filesystem\fat\fname.c
efi1.1\edk\drivers\filesystem\fat\info.c
efi1.1\edk\drivers\filesystem\fat\misc.c
efi1.1\edk\drivers\filesystem\fat\open.c
efi1.1\edk\drivers\ide\ata.c
efi1.1\edk\drivers\ide\atapi.c
efi1.1\edk\drivers\isabus\isabus.c
efi1.1\edk\drivers\isabus\isabus.h
efi1.1\edk\drivers\isabus\isaio.c
efi1.1\edk\drivers\isafloppy\isafloppy.c
efi1.1\edk\drivers\isafloppy\isafloppy.h
efi1.1\edk\drivers\isafloppy\isafloppyblock.c
efi1.1\edk\drivers\isafloppy\isafloppyctrl.c
efi1.1\edk\drivers\isaserial\serial.c
efi1.1\edk\drivers\pcatpcirootbridge\pcatpcirootbridgeio.c
efi1.1\edk\drivers\pcatpcirootbridge\ia32\pcatio.c
efi1.1\edk\drivers\pcatpcirootbridge\ipf\pcatio.c
efi1.1\edk\drivers\pcibus\pcibus.h
efi1.1\edk\drivers\pcibus\pcienumeratorsupport.c
efi1.1\edk\drivers\pcibus\pciio.c
efi1.1\edk\drivers\ps2mouse\commps2.c
efi1.1\edk\drivers\ps2mouse\ps2mouse.c
efi1.1\edk\drivers\ps2mouse\ps2mouse.h

efi1.1\edk\drivers\pxebc\bc.c
efi1.1\edk\drivers\pxebc\bc.h
efi1.1\edk\drivers\pxebc\dhcp.h
efi1.1\edk\drivers\pxebc\hton.h
efi1.1\edk\drivers\pxebc\ip.h
efi1.1\edk\drivers\pxebc\pxe_bc_arp.c
efi1.1\edk\drivers\pxebc\pxe_bc_dhcp.c
efi1.1\edk\drivers\pxebc\pxe_bc_igmp.c
efi1.1\edk\drivers\pxebc\pxe_bc_ip.c
efi1.1\edk\drivers\pxebc\pxe_bc_mtftp.c
efi1.1\edk\drivers\pxebc\pxe_bc_tcp.c
efi1.1\edk\drivers\pxebc\pxe_bc_udp.c
efi1.1\edk\drivers\pxebc\pxe_loadfile.c
efi1.1\edk\drivers\pxebc\tftp.h
efi1.1\edk\drivers\scsibus\scsibus.c
efi1.1\edk\drivers\scsidisk\scsidisk.c
efi1.1\edk\drivers\serialmouse\serialmouse.c
efi1.1\edk\drivers\snp32_64\initialize.c
efi1.1\edk\drivers\snp32_64\make.inf
efi1.1\edk\drivers\snp32_64\shutdown.c
efi1.1\edk\drivers\snp32_64\snp.c
efi1.1\edk\drivers\undi\decode.c
efi1.1\edk\drivers\undi\e100b.c
efi1.1\edk\drivers\usb\uhci\uhchlp.c
efi1.1\edk\drivers\usb\uhci\uhci.c
efi1.1\edk\drivers\usb\uhci\uhci.h
efi1.1\edk\drivers\usb\usbbot\bot.c
efi1.1\edk\drivers\usb\usbbus\hid.h
efi1.1\edk\drivers\usb\usbbus\hub.c
efi1.1\edk\drivers\usb\usbbus\usb.c
efi1.1\edk\drivers\usb\usbbus\usbbus.c
efi1.1\edk\drivers\usb\usbbus\usbbus.h
efi1.1\edk\drivers\usb\usbbus\usbio.c
efi1.1\edk\drivers\usb\usbbus\usblib.c
efi1.1\edk\drivers\usb\usbcbi\cbi0\cbi0.c
efi1.1\edk\drivers\usb\usbcbi\cbi1\cbi1.c

efi1.1\edk\drivers\usb\usbkb\keyboard.c
efi1.1\edk\drivers\usb\usbkb\keyboard.h
efi1.1\edk\drivers\usb\usbmassstorage\usbmassstorage.c
efi1.1\edk\drivers\usb\usbmassstorage\usbmassstoragedata.h
efi1.1\edk\drivers\usb\usbmassstorage\usbmassstoragehelper.c
efi1.1\edk\drivers\usb\usbmassstorage\usbmassstoragehelper.h
efi1.1\edk\drivers\usb\usbmouse\usbmouse.c
efi1.1\edk\drivers\vgaclass\vgaclass.c
efi1.1\edk\drivers\winntthunk\blockio\winntblockio.c
efi1.1\edk\drivers\winntthunk\blockio\winntblockio.h
efi1.1\edk\drivers\winntthunk\console\console.c
efi1.1\edk\drivers\winntthunk\console\consoleout.c
efi1.1\edk\drivers\winntthunk\serialio\winntserialio.c
efi1.1\edk\drivers\winntthunk\simplefilesystem\winntsimplefilesystem.c
efi1.1\edk\drivers\winntthunk\uga\winntugadriver.c
efi1.1\edk\drivers\winntthunk\uga\winntugascreen.c
efi1.1\edk\guid\bmp\bmp.c
efi1.1\edk\guid\globalvariable\globalvariable.c
efi1.1\edk\guid\globalvariable\globalvariable.h
efi1.1\edk\guid\gpt\gpt.c
efi1.1\edk\guid\gpt\gpt.h
efi1.1\edk\guid\pcansi\pcansi.c
efi1.1\edk\guid\pcansi\pcansi.h
efi1.1\edk\guid\salsystemtable\salsystemtable.c
efi1.1\edk\guid\salsystemtable\salsystemtable.h
efi1.1\edk\guid\smbios\smbios.c
efi1.1\edk\guid\smbios\smbios.h
efi1.1\edk\include\efi.h
efi1.1\edk\include\efiapi.h
efi1.1\edk\include\efibuild.h
efi1.1\edk\include\efidevicepath.h
efi1.1\edk\include\efitypes.h
efi1.1\edk\include\pxe.h
efi1.1\edk\include\scsi.h
efi1.1\edk\include\ebc\efibind.h
efi1.1\edk\include\ia32\efibind.h

efi1.1\edk\include\ipf\efibind.h
efi1.1\edk\lib\efidriverlib\efidriverlib.c
efi1.1\edk\protocol\make.inf
efi1.1\edk\protocol\blockio\blockio.h
efi1.1\edk\protocol\debugport\debugport.h
efi1.1\edk\protocol\debugsupport\debugsupport.h
efi1.1\edk\protocol\decompress\decompress.c
efi1.1\edk\protocol\devicepath\devicepath.c
efi1.1\edk\protocol\diskio\diskio.c
efi1.1\edk\protocol\ebc\ebc.c
efi1.1\edk\protocol\ebc\ebc.h
efi1.1\edk\protocol\efinetworkinterfaceidentifier\efinetworkinterfaceidentifier.h
efi1.1\edk\protocol\isaio\isaio.c
efi1.1\edk\protocol\legacyboot\legacyboot.h
efi1.1\edk\protocol\pciio\pciio.h
efi1.1\edk\protocol\pcirootbridgeio\pcirootbridgeio.h
efi1.1\edk\protocol\platformdriveroverride\platformdriveroverride.h
efi1.1\edk\protocol\pxebasecode\pxebasecode.h
efi1.1\edk\protocol\scsipassthru\scsipassthru.h
efi1.1\edk\protocol\serialio\serialio.c
efi1.1\edk\protocol\simplefilesystem\simplefilesystem.c
efi1.1\edk\protocol\tcp\tcp.c
efi1.1\edk\protocol\ugaio\ugaio.h
efi1.1\edk\protocol\ugasplash\ugasplash.c
efi1.1\edk\protocol\ugasplash\ugasplash.h
efi1.1\edk\protocol\unicodecollation\unicodecollation.h
efi1.1\edk\protocol\winntthunk\winntthunk.h
efi1.1\inc\efi.h
efi1.1\inc\efiapi.h
efi1.1\inc\efidevp.h
efi1.1\inc\efiprot.h
efi1.1\inc\efipxebc.h
efi1.1\inc\pci22.h
efi1.1\inc\ia32\efibind.h
efi1.1\inc\ia32\pe.h
efi1.1\inc\ipf\efibind.h

efi1.1\inc\ipf\pe.h
efi1.1\lib\misc.c
efi1.1\lib\print.c
efi1.1\lib\sread.c
efi1.1\notes\efi1.1shellcommands.doc
efi1.1\notes\relnote.doc
efi1.1\shell\attrib\attrib.c
efi1.1\shell\bcfg\bcfg.c
efi1.1\shell\comp\comp.c
efi1.1\shell\cp\cp.c
efi1.1\shell\date\date.c
efi1.1\shell\debug\dblk.c
efi1.1\shell\debug\efidump.c
efi1.1\shell\dmpstore\dmpstore.c
efi1.1\shell\edit\libeditor.c
efi1.1\shell\edit\libfilebuffer.c
efi1.1\shell\edit\libstatusbar.c
efi1.1\shell\edit\libstatusbar.h
efi1.1\shell\edit\libtitlebar.c
efi1.1\shell\edit\libtitlebar.h
efi1.1\shell\eficompress\compressmain.c
efi1.1\shell\getmtc\getmtc.c
efi1.1\shell\helpdata\helpdata.src
efi1.1\shell\hexedit\libbufferimage.c
efi1.1\shell\hexedit\libdiskimage.c
efi1.1\shell\hexedit\libeditor.c
efi1.1\shell\hexedit\libfileimage.c
efi1.1\shell\hexedit\libmemimage.c
efi1.1\shell\hexedit\libstatusbar.c
efi1.1\shell\hexedit\libstatusbar.h
efi1.1\shell\inc\efiimage.h
efi1.1\shell\inc\shell.h
efi1.1\shell\lib\helpinfo.c
efi1.1\shell\lib\init.c
efi1.1\shell\lib\misc.c
efi1.1\shell\load\load.c

efi1.1\shell\loadpcirom\loadpcirom.c
efi1.1\shell\ls\ls.c
efi1.1\shell\memmap\memmap.c
efi1.1\shell\mkdir\mkdir.c
efi1.1\shell\newshell\init.c
efi1.1\shell\pci\pci.c
efi1.1\shell\pci\pci_class.h
efi1.1\shell\reset\reset.c
efi1.1\shell\rm\rm.c
efi1.1\shell\shellenv\batch.c
efi1.1\shell\shellenv\cmddisp.c
efi1.1\shell\shellenv\connect.c
efi1.1\shell\shellenv\dprot.c
efi1.1\shell\shellenv\exec.c
efi1.1\shell\shellenv\for.c
efi1.1\shell\shellenv\help.c
efi1.1\shell\shellenv\if.c
efi1.1\shell\shellenv\map.c
efi1.1\shell\shellenv\marg.c
efi1.1\shell\shellenv\mount.c
efi1.1\shell\shellenv\protid.c
efi1.1\shell\shellenv\shelle.h
efi1.1\shell\shellenv\var.c
efi1.1\shell\time\time.c
efi1.1\shell\type\type.c
efi1.1\shell\ver\ipf\ver64.c

B.3 EFI 1.10.14.60

B.3.1 Core Bug Fixes

1. Updated version to EFI 1.10.14.60.
2. Updated the BIOS32 boot floppy image to EFI 1.10.14.60.
3. Updated the release notes to EFI 1.10.14.60.
4. Fixed bug in the EFI Compression Algorithm. This was fixed in the Compress build tool and the EfiCompress Shell command.
5. Removed ASSERTs in the Event Services that were causing the HCT ProtocolInstallNotify test to fail when EFI was built with EFI_DEBUG=YES.

6. Updated DisconnectController() to only stop a bus controller is the child parameter is NULL. This will keep a hot plug bus driver (like USB) active even if all of the USB devices are temporarily removed.
7. Removed ASSERTs in the ReinstallProtocolInterface() that were causing the HCT ProtocolInstallNotify test to fail when EFI was built with EFI_DEBUG=YES.
8. Removed ASSERTs in the Memory Services that were causing the HCT DeviceIo test to fail when EFI was built with EFI_DEBUG=YES.
9. Fixed a corner case with the Delta parameter in the Blt() function of all the UGA Draw Protocol implementations including the Cirrus Logic 5430 UGA driver and the WinNtThunk UGA driver.
10. Updated the ConPlatform driver to not automatically add hot-plug consoles to the standard error console.
11. Updated the ConSplitter driver to preserve the background bitmap when the size of the graphical text console is adjusted.
12. Fixed a cursor management bug in the ConSplitter driver that appeared when the size of the graphical text console is adjusted.
13. Fixed the ConSplitter driver to guarantee that the Simple Pointer Splitter never generates a divide by 0 condition.
14. Updated the GraphicsConsole driver to clear the entire display when a 100x31 text display is generated on top of an 800x600 UGA display.
15. Removed an extra FreePool() call from the GraphicsConsole driver.
16. Fixed a handle bug in the Stop() function of the Terminal driver.
17. Fixed a bug in the Terminal driver that was not properly maintaining the ConInDev, ConOutDev, and StdErrDev environment variables.
18. Fixed a bug in the VT-UTF8 terminal emulation driver.
19. Cleaned up the DebugSupport driver for the Itanium processor family.
20. Fixed a bug in the FAT file system driver that was hanging when media was removed in the middle of a copy operation.
21. Reduced the device detection time in the native IDE bus driver.
22. Added a check for the Protective MBR to the GPT detection code in the Partition driver.
23. Fixed an un-initialized variable in the Partition driver.
24. Cleaned up the PxeBc driver.
25. Cleaned up the PxeDhcp4 driver.
26. Cleaned up the Snp3264 driver.
27. Cleaned up the Intel® 100 Mb UNDI driver.
28. Added media detection support to the Intel 100 Mb UNDI driver.
29. Fixed a bug in the UHCI driver for memory configurations greater than 4 GB.
30. Updated the USB Bus Driver to force the maximum packet size to 8 bytes.
31. Fixed a bug in the repeat key functionality of the USB keyboard driver and some mapping bugs.
32. Fixed a Fill operation bug in the WinNtThunk UGA driver.
33. Changed PC-CARD device path field name from “SocketNumber” to “FunctionNumber” to match naming conventions in the PC-CARD related specifications.
34. Added checks for a valid EFI System Table in the EFI Driver Library Initialization function.

35. Updated the UGA I/O Protocol include file to match the EFI 1.10 Specification Draft 0.95.
36. Displayed the LUN field when a Fiber Channel device path node is converted to a string.
37. Removed BUGBUG comment from the BCFG Shell Command.
38. Fixed some grammar errors in the EFI shell help text.
39. Fixed the ELSE command in the EFI Shell.
40. Fixed hang in the EFI Shell when media is changed.

B.3.2 List of Deleted Files

efi1.1\edk\drivers\debugsupport\ipf\common.inc

efi1.1\edk\drivers\debugsupport\ipf\ds64macros.inc

B.3.3 List of New Files

efi1.1\edk\drivers\debugsupport\ipf\common.i

efi1.1\edk\drivers\debugsupport\ipf\ds64macros.i

B.3.4 List of Modified Files

efi1.1\build\tools\bin\efiboot.img

efi1.1\build\tools\src\common\compress.c

efi1.1\corefw\fw\efi\exec\event.c

efi1.1\corefw\fw\efi\exec\exec.h

efi1.1\corefw\fw\efi\exec\timer.c

efi1.1\corefw\fw\efi\hand\driversupport.c

efi1.1\corefw\fw\efi\hand\notify.c

efi1.1\corefw\fw\efi\mem\page.c

efi1.1\edk\drivers\cirruslogic5430\cirruslogic5430ugadraw.c

efi1.1\edk\drivers\console\conplatform\conplatform.c

efi1.1\edk\drivers\console\consplitter\consplitter.c

efi1.1\edk\drivers\console\consplitter\consplittergraphics.c

efi1.1\edk\drivers\console\graphicsconsole\graphicsconsole.c

efi1.1\edk\drivers\console\graphicsconsole\graphicsconsole.h

efi1.1\edk\drivers\console\terminal\terminal.c

efi1.1\edk\drivers\console\terminal\vtutf8.c

efi1.1\edk\drivers\debugsupport\ipf\asmfuncs.s

efi1.1\edk\drivers\debugsupport\ipf\pldebugsupport.h

efi1.1\edk\drivers\filesystem\fat\cache.c

efi1.1\edk\drivers\ide\ide.c

efi1.1\edk\drivers\partition\gpt.c

efi1.1\edk\drivers\partition\mbr.c

efi1.1\edk\drivers\pxebc\bc.c
efi1.1\edk\drivers\pxebc\bc.h
efi1.1\edk\drivers\pxebc\dhcp.h
efi1.1\edk\drivers\pxebc\ip.h
efi1.1\edk\drivers\pxebc\pxe_bc_arp.c
efi1.1\edk\drivers\pxebc\pxe_bc_dhcp.c
efi1.1\edk\drivers\pxebc\pxe_bc_igmp.c
efi1.1\edk\drivers\pxebc\pxe_bc_ip.c
efi1.1\edk\drivers\pxebc\pxe_bc_mtftp.c
efi1.1\edk\drivers\pxebc\pxe_bc_tcp.c
efi1.1\edk\drivers\pxebc\pxe_bc_udp.c
efi1.1\edk\drivers\pxebc\pxe_loadfile.c
efi1.1\edk\drivers\pxebc\tftp.h
efi1.1\edk\drivers\pxedhcp4\pxedhcp4.h
efi1.1\edk\drivers\pxedhcp4\pxedhcp4initselect.c
efi1.1\edk\drivers\pxedhcp4\pxedhcp4release.c
efi1.1\edk\drivers\pxedhcp4\pxedhcp4renewrebind.c
efi1.1\edk\drivers\pxedhcp4\pxedhcp4setup.c
efi1.1\edk\drivers\pxedhcp4\support.c
efi1.1\edk\drivers\snp32_64\callback.c
efi1.1\edk\drivers\snp32_64\get_status.c
efi1.1\edk\drivers\snp32_64\initialize.c
efi1.1\edk\drivers\snp32_64\mcast_ip_to_mac.c
efi1.1\edk\drivers\snp32_64\nvdata.c
efi1.1\edk\drivers\snp32_64\receive.c
efi1.1\edk\drivers\snp32_64\receive_filters.c
efi1.1\edk\drivers\snp32_64\reset.c
efi1.1\edk\drivers\snp32_64\shutdown.c
efi1.1\edk\drivers\snp32_64\snp.c
efi1.1\edk\drivers\snp32_64\snp.h
efi1.1\edk\drivers\snp32_64\start.c
efi1.1\edk\drivers\snp32_64\station_address.c
efi1.1\edk\drivers\snp32_64\statistics.c
efi1.1\edk\drivers\snp32_64\stop.c
efi1.1\edk\drivers\snp32_64\transmit.c
efi1.1\edk\drivers\undi\decode.c

efi1.1\edk\drivers\undi\e100b.c
efi1.1\edk\drivers\undi\e100b.h
efi1.1\edk\drivers\undi\init.c
efi1.1\edk\drivers\undi\undi32.h
efi1.1\edk\drivers\usb\uhci\uhci.c
efi1.1\edk\drivers\usb\usbbus\usbbus.c
efi1.1\edk\drivers\usb\usbbus\usbio.c
efi1.1\edk\drivers\usb\usbkb\keyboard.c
efi1.1\edk\drivers\winntthunk\uga\winntugascreen.c
efi1.1\edk\include\efi.h
efi1.1\edk\include\efidevicepath.h
efi1.1\edk\include\efipxe.h
efi1.1\edk\include\pci22.h
efi1.1\edk\include\ipf\efibind.h
efi1.1\edk\lib\efidriverlib\efidriverlib.c
efi1.1\edk\protocol\pxebasecodecallback\pxebasecodecallback.h
efi1.1\edk\protocol\ugaio\ugaio.h
efi1.1\inc\efi.h
efi1.1\inc\efidevp.h
efi1.1\inc\ipf\efibind.h
efi1.1\lib\dpath.c
efi1.1\notes\relnote.doc
efi1.1\shell\bcfg\bcfg.c
efi1.1\shell\eficompress\compress.c
efi1.1\shell\helpdata\helpdata.src
efi1.1\shell\shellenv\cmddisp.c
efi1.1\shell\shellenv\dprot.c
efi1.1\shell\shellenv\marg.c

B.4 EFI 1.10.14.59

B.4.1 Core Bug Fixes

1. Updated version to EFI 1.10.14.59.
2. Updated the BIOS32 boot floppy image to EFI 1.10.14.59.
3. Updated the release notes.
4. Added a document that is a review draft of the SCSI Driver Model.
5. Added a document that describes the SAL to EFI Handoff State.

6. Updated the EFI 1.10 Sample Implementation to match the 0.95 draft of the EFI 1.10 Specification.
7. Added the Console Control, PXE DHCP, SCSI I/O, and TCP Protocols
8. Added the Hot Plug Device GUID.
9. Added the USB CBI0 Driver.
10. Added the SCSI Bus Driver.
11. Added the SCSI Disk Driver.
12. Added the PXE DHCP Driver.
13. Performed a general code style clean up of the entire EFI 1.10 source base.
14. Cleaned up the implementation of the EFI Driver Library.
15. Updated the BIOS32 and IA-32EMB builds to convert the EFI core from physical to virtual mappings in SetVirtualAddressMap().
16. Updated all the build tips to support being compiled with /O1. This is now the default for NT32, BIOS32, and IA-32EMB.
17. Fixed a hazard in the Boot Services RaiseTPL() and RestoreTPL().
18. Fixed the EFI System Table so the FirmwareVendor field is allocated from EfiRuntimeServicesData.
19. Added image handle for the EFI Core.
20. Fixed a bug in the Boot Service AllocatePages() for IA-32 systems with more than 4 GB of system memory.
21. Updated CopyMem() to support overlapping buffers.
22. Fixed a bug in the initialization of the LangCode and Lang variables in the EFI Boot Manager.
23. Updated the SALEFIHANDOFF structure to pass up the base and size of the EFI Core image.
24. Fixed ResetSystem() for IA-32EMB and BIOS32 to use a runtime safe function to induce a hardware reset.
25. Added support for multiple timer modes in the 8253/8254 driver.
26. Fixed hazard in the runtime library lock functions.
27. Fixed BIOS32 and IA-32EMB build tips to support multiple runtime images at the end of the EFILDR file.
28. Updated the Console Splitter Driver to include a Simple Pointer Splitter, a UGA Draw Splitter, and a Console Control Protocol.
29. Updated the PCI Bus Driver to support 64-bit BARs, to prevent interrupts during BAR scanning, and support overlapping buffers in CopyMem().
30. Updated WinNtThunk Simple File System driver to support GetInfo() and SetInfo().
31. Fixed a bug in the Blt() API of all the sample UGA drivers.
32. Add glyphs 0xa0-0xff to the Graphics Console driver.
33. Updated EBC interpreter to match 0.95 draft of the EFI 1.10 Specification.
34. Fixed possible infinite recursion in the Partition driver.
35. Fixed several bugs and cleaned up code in the network drivers.
36. Fixed several bugs and cleaned up code in the USB drivers.
37. Fixed USB stack to allow USB keyboard input during a PXE boot.
38. Updated to UNDI driver to match the 0.95 draft of the EFI 1.10 Specification
39. Fixed the resolution data for all the Simple Pointer drivers.
40. Fixed EDIT and HEXEDIT to use the resolution data from the Simple Pointer devices.

41. Updated LOADBMP Shell command to use the Console Control Protocol.
42. Updated PCI Shell command to support 64-bit BARs.
43. Fixed use of ConnectController() for context overrides throughout the EFI Shell.

B.4.2 List of New Files

efi1.1\build\tools\src\vccheck\vccheck.c

efi1.1\corefw\fw\efi\exec\ia32\processorasms.asm

efi1.1\corefw\fw\platform\inc\ia32\efildrhandoff.h

efi1.1\corefw\fw\platform\pldriver\efildr\make.inf

efi1.1\corefw\fw\platform\pldriver\efildr\plefildr.c

efi1.1\corefw\fw\platform\pldriver\efildr\plefildr.h

efi1.1\corefw\fw\platform\pldriver\govirtual\ipf\iva.s

efi1.1\edk\drivers\console\consplitter\consplittergraphics.c

efi1.1\edk\drivers\ide\configuretiming.c

efi1.1\edk\drivers\pcibus\pcicommand.c

efi1.1\edk\drivers\pcibus\pcicommand.h

efi1.1\edk\drivers\pcibus\pcidevicesupport.c

efi1.1\edk\drivers\pcibus\pcidevicesupport.h

efi1.1\edk\drivers\pcibus\pcidriveroverride.c

efi1.1\edk\drivers\pcibus\pcidriveroverride.h

efi1.1\edk\drivers\pcibus\pcienumerator.c

efi1.1\edk\drivers\pcibus\pcienumerator.h

efi1.1\edk\drivers\pcibus\pcienumeratorsupport.c

efi1.1\edk\drivers\pcibus\pcienumeratorsupport.h

efi1.1\edk\drivers\pcibus\pciio.h

efi1.1\edk\drivers\pcibus\pcioptionromsupport.c

efi1.1\edk\drivers\pcibus\pcioptionromsupport.h

efi1.1\edk\drivers\pcibus\pcipowermanagement.c

efi1.1\edk\drivers\pcibus\pcipowermanagement.h

efi1.1\edk\drivers\pxebc\hton.h

efi1.1\edk\drivers\pxebc\pxe_bc_tcp.c

efi1.1\edk\drivers\pxedhcp4\componentname.c

efi1.1\edk\drivers\pxedhcp4\make.inf

efi1.1\edk\drivers\pxedhcp4\pxedhcp4.c

efi1.1\edk\drivers\pxedhcp4\pxedhcp4.h

efi1.1\edk\drivers\pxedhcp4\pxedhcp4initselect.c

efi1.1\edk\drivers\pxedhcp4\pxedhcp4release.c

efi1.1\edk\drivers\pxedhcp4\pxedhcp4renewrebind.c
efi1.1\edk\drivers\pxedhcp4\pxedhcp4run.c
efi1.1\edk\drivers\pxedhcp4\pxedhcp4setup.c
efi1.1\edk\drivers\pxedhcp4\support.c
efi1.1\edk\drivers\scsibus\componentname.c
efi1.1\edk\drivers\scsibus\make.inf
efi1.1\edk\drivers\scsibus\scsibus.c
efi1.1\edk\drivers\scsibus\scsibus.h
efi1.1\edk\drivers\scsibus\scsilib.c
efi1.1\edk\drivers\scsibus\scsilib.h
efi1.1\edk\drivers\scsidisk\componentname.c
efi1.1\edk\drivers\scsidisk\make.inf
efi1.1\edk\drivers\scsidisk\scsidisk.c
efi1.1\edk\drivers\scsidisk\scsidisk.h
efi1.1\edk\drivers\usb\usbcbi\cbi0\cbi0.c
efi1.1\edk\drivers\usb\usbcbi\cbi0\componentname.c
efi1.1\edk\drivers\usb\usbcbi\cbi0\make.inf
efi1.1\edk\guid\hotplugdevice\hotplugdevice.c
efi1.1\edk\guid\hotplugdevice\hotplugdevice.h
efi1.1\edk\include\scsi.h
efi1.1\edk\lib\eficommonlib\eficompareguid.c
efi1.1\edk\lib\eficommonlib\eficomparemem.c
efi1.1\edk\lib\eficommonlib\eficopymem.c
efi1.1\edk\lib\eficommonlib\efisetmem.c
efi1.1\edk\lib\eficommonlib\make.inf
efi1.1\edk\lib\eficommonlib\string.c
efi1.1\edk\lib\eficommonlib\ebc\ebcmath.c
efi1.1\edk\lib\eficommonlib\ia32\cpufuncs.c
efi1.1\edk\lib\eficommonlib\ia32\eficopymem.asm
efi1.1\edk\lib\eficommonlib\ia32\efisetmem.asm
efi1.1\edk\lib\eficommonlib\ia32\efizeromem.asm
efi1.1\edk\lib\eficommonlib\ia32\math.c
efi1.1\edk\lib\eficommonlib\ipf\eficopymem.s
efi1.1\edk\lib\eficommonlib\ipf\efizeromem.s
efi1.1\edk\lib\eficommonlib\ipf\math.c
efi1.1\edk\lib\include\eficommonlib.h

efi1.1\edk\lib\print\boxdraw.c
efi1.1\edk\lib\print\stderr.c
efi1.1\edk\lib\print\ascii\printwidth.h
efi1.1\edk\lib\print\ascii\sprint.c
efi1.1\edk\lib\print\unicode\make.inf
efi1.1\edk\lib\print\unicode\printwidth.h
efi1.1\edk\lib\print\unicode\sprint.c
efi1.1\edk\protocol\consolecontrol\consolecontrol.c
efi1.1\edk\protocol\consolecontrol\consolecontrol.h
efi1.1\edk\protocol\pxedhcp4\pxedhcp4.c
efi1.1\edk\protocol\pxedhcp4\pxedhcp4.h
efi1.1\edk\protocol\pxedhcp4callback\pxedhcp4callback.c
efi1.1\edk\protocol\pxedhcp4callback\pxedhcp4callback.h
efi1.1\edk\protocol\scsiio\scsiio.c
efi1.1\edk\protocol\scsiio\scsiio.h
efi1.1\edk\protocol\tcp\tcp.c
efi1.1\edk\protocol\tcp\tcp.h
efi1.1\notes\salefihandoffstate.doc
efi1.1\notes\scsidrivermodel.doc

B.4.3 List of Modified Files

efi1.1\build\bios32\makefile
efi1.1\build\bios32\master.env
efi1.1\build\ebcdrivers\makefile
efi1.1\build\ia-32emb\makefile
efi1.1\build\ia-32emb\master.env
efi1.1\build\ia32drivers\makefile
efi1.1\build\ipfdrivers\makefile
efi1.1\build\nt32\makefile
efi1.1\build\nt32\master.env
efi1.1\build\sal64\makefile
efi1.1\build\tools\bin\col.exe
efi1.1\build\tools\bin\efiboot.img
efi1.1\build\tools\bin\eficompress.exe
efi1.1\build\tools\bin\efildrimage.exe
efi1.1\build\tools\bin\efirom.exe

efi1.1\build\tools\bin\fwimage.exe
efi1.1\build\tools\bin\genhelp.exe
efi1.1\build\tools\bin\genmake.exe
efi1.1\build\tools\bin\splitfile.exe
efi1.1\build\tools\src\makefile
efi1.1\build\tools\src\efirom\efirom.c
efi1.1\corefw\fw\efi\make.inf
efi1.1\corefw\fw\efi\crc\crc.c
efi1.1\corefw\fw\efi\exec\event.c
efi1.1\corefw\fw\efi\exec\execdata.c
efi1.1\corefw\fw\efi\exec\tpl.c
efi1.1\corefw\fw\efi\hand\driversupport.c
efi1.1\corefw\fw\efi\hand\handle.c
efi1.1\corefw\fw\efi\hand\notify.c
efi1.1\corefw\fw\efi\inc\efifw.h
efi1.1\corefw\fw\efi\init\bsdata.c
efi1.1\corefw\fw\efi\init\init.c
efi1.1\corefw\fw\efi\loader\ldata.c
efi1.1\corefw\fw\efi\loader\load.c
efi1.1\corefw\fw\efi\loader\pe.c
efi1.1\corefw\fw\efi\mem\memdata.c
efi1.1\corefw\fw\efi\mem\page.c
efi1.1\corefw\fw\efi\mem\virt.c
efi1.1\corefw\fw\efi\misc\ia32\eficorecopymem.asm
efi1.1\corefw\fw\efi\variable\varinit.c
efi1.1\corefw\fw\efi\variable\vario.c
efi1.1\corefw\fw\inc\emulenv.h
efi1.1\corefw\fw\platform\bootmgr\default\bm.h
efi1.1\corefw\fw\platform\bootmgr\default\data.c
efi1.1\corefw\fw\platform\bootmgr\default\init.c
efi1.1\corefw\fw\platform\bootmgr\default\lang.c
efi1.1\corefw\fw\platform\bootmgr\default\make.inf
efi1.1\corefw\fw\platform\bootmgr\default\var.c
efi1.1\corefw\fw\platform\bootmgr\maint\console.c
efi1.1\corefw\fw\platform\buildtip\bios32\efildr.c
efi1.1\corefw\fw\platform\buildtip\bios32\init.c

efi1.1\corefw\fw\platform\buildtip\bios32\make.inf
efi1.1\corefw\fw\platform\buildtip\bootsector\start.asm
efi1.1\corefw\fw\platform\buildtip\ia-32emb\efildr.c
efi1.1\corefw\fw\platform\buildtip\ia-32emb\init.c
efi1.1\corefw\fw\platform\buildtip\ia-32emb\make.inf
efi1.1\corefw\fw\platform\buildtip\inc\drivers.h
efi1.1\corefw\fw\platform\buildtip\nt32\init.c
efi1.1\corefw\fw\platform\buildtip\nt32\winntthunk.c
efi1.1\corefw\fw\platform\buildtip\sal64\init.c
efi1.1\corefw\fw\platform\drivers\biosint\biossnp16\biossnp16.c
efi1.1\corefw\fw\platform\drivers\biosint\biossnp16\biossnp16.h
efi1.1\corefw\fw\platform\drivers\biosint\biossnp16\misc.c
efi1.1\corefw\fw\platform\drivers\biosint\disk\biosint13.c
efi1.1\corefw\fw\platform\inc\make.inf
efi1.1\corefw\fw\platform\inc\ipf\salhandoff.h
efi1.1\corefw\fw\platform\lib\platformlib.c
efi1.1\corefw\fw\platform\pldriver\biosint\ia32\int.c
efi1.1\corefw\fw\platform\pldriver\cache\ia32\cacheflush.c
efi1.1\corefw\fw\platform\pldriver\cpu_timer\ia32\cputimer.c
efi1.1\corefw\fw\platform\pldriver\defio\make.inf
efi1.1\corefw\fw\platform\pldriver\defio\ipf\definit.c
efi1.1\corefw\fw\platform\pldriver\govirtual\make.inf
efi1.1\corefw\fw\platform\pldriver\govirtual\plvirtual.h
efi1.1\corefw\fw\platform\pldriver\govirtual\ia32\virtual.c
efi1.1\corefw\fw\platform\pldriver\govirtual\ipf\virtual.c
efi1.1\corefw\fw\platform\pldriver\interruptcontroller\plintctrl.h
efi1.1\corefw\fw\platform\pldriver\interruptcontroller\8259\8259.c
efi1.1\corefw\fw\platform\pldriver\reset\make.inf
efi1.1\corefw\fw\platform\pldriver\reset\ia32\reset.c
efi1.1\corefw\fw\platform\pldriver\sal\plsal.h
efi1.1\corefw\fw\platform\pldriver\sal\rtdata.c
efi1.1\corefw\fw\platform\pldriver\sal\sal.c
efi1.1\corefw\fw\platform\pldriver\stall\8253\stall.c
efi1.1\corefw\fw\platform\pldriver\stall\8253\stall.h
efi1.1\corefw\fw\platform\pldriver\time\dallas1287\time.c
efi1.1\edk\drivers\atapipassthru\atapipassthru.c

efi1.1\edk\drivers\atapipassthru\atapipassthru.h
efi1.1\edk\drivers\atapipassthru\componentname.c
efi1.1\edk\drivers\atapipassthru\make.inf
efi1.1\edk\drivers\bis\basecode\make.inf
efi1.1\edk\drivers\cirruslogic5430\cirruslogic5430.c
efi1.1\edk\drivers\cirruslogic5430\cirruslogic5430ugadraw.c
efi1.1\edk\drivers\cirruslogic5430\make.inf
efi1.1\edk\drivers\console\conplatform\componentname.c
efi1.1\edk\drivers\console\conplatform\conplatform.c
efi1.1\edk\drivers\console\conplatform\conplatform.h
efi1.1\edk\drivers\console\conplatform\make.inf
efi1.1\edk\drivers\console\consplitter\componentname.c
efi1.1\edk\drivers\console\consplitter\consplitter.c
efi1.1\edk\drivers\console\consplitter\consplitter.h
efi1.1\edk\drivers\console\consplitter\make.inf
efi1.1\edk\drivers\console\graphicsconsole\graphicsconsole.c
efi1.1\edk\drivers\console\graphicsconsole\laffstd.c
efi1.1\edk\drivers\console\graphicsconsole\make.inf
efi1.1\edk\drivers\console\terminal\componentname.c
efi1.1\edk\drivers\console\terminal\make.inf
efi1.1\edk\drivers\console\terminal\pcansi.c
efi1.1\edk\drivers\console\terminal\terminal.c
efi1.1\edk\drivers\console\terminal\terminal.h
efi1.1\edk\drivers\console\terminal\terminalconin.c
efi1.1\edk\drivers\console\terminal\terminalconout.c
efi1.1\edk\drivers\console\terminal\vt100.c
efi1.1\edk\drivers\console\terminal\vt100plus.c
efi1.1\edk\drivers\console\terminal\vtutf8.c
efi1.1\edk\drivers\debugport\debugport.c
efi1.1\edk\drivers\debugport\make.inf
efi1.1\edk\drivers\debugsupport\make.inf
efi1.1\edk\drivers\debugsupport\ia32\asmfuncs.asm
efi1.1\edk\drivers\debugsupport\ia32\pldebugsupport.c
efi1.1\edk\drivers\debugsupport\ia32\pldebugsupport.h
efi1.1\edk\drivers\debugsupport\ipf\asmfuncs.s
efi1.1\edk\drivers\debugsupport\ipf\common.inc

efi1.1\edk\drivers\debugsupport\ipf\ds64macros.inc
efi1.1\edk\drivers\debugsupport\ipf\pldebugsupport.c
efi1.1\edk\drivers\diskio\diskio.c
efi1.1\edk\drivers\diskio\make.inf
efi1.1\edk\drivers\ebc\ebcexecute.c
efi1.1\edk\drivers\ebc\ebcexecute.h
efi1.1\edk\drivers\ebc\ebcint.c
efi1.1\edk\drivers\ebc\ebcint.h
efi1.1\edk\drivers\ebc\make.inf
efi1.1\edk\drivers\ebc\ia32\ebcsupport.c
efi1.1\edk\drivers\ebc\ia32\ia32math.c
efi1.1\edk\drivers\ebc\ipf\ebcsupport.c
efi1.1\edk\drivers\filesystem\fat\cache.c
efi1.1\edk\drivers\filesystem\fat\componentname.c
efi1.1\edk\drivers\filesystem\fat\delete.c
efi1.1\edk\drivers\filesystem\fat\dirent.c
efi1.1\edk\drivers\filesystem\fat\dirsup.c
efi1.1\edk\drivers\filesystem\fat\fat.c
efi1.1\edk\drivers\filesystem\fat\fat.h
efi1.1\edk\drivers\filesystem\fat\flush.c
efi1.1\edk\drivers\filesystem\fat\fname.c
efi1.1\edk\drivers\filesystem\fat\fspace.c
efi1.1\edk\drivers\filesystem\fat\hash.c
efi1.1\edk\drivers\filesystem\fat\info.c
efi1.1\edk\drivers\filesystem\fat\init.c
efi1.1\edk\drivers\filesystem\fat\make.inf
efi1.1\edk\drivers\filesystem\fat\misc.c
efi1.1\edk\drivers\filesystem\fat\open.c
efi1.1\edk\drivers\filesystem\fat\openvol.c
efi1.1\edk\drivers\filesystem\fat\rw.c
efi1.1\edk\drivers\ide\ata.c
efi1.1\edk\drivers\ide\atapi.c
efi1.1\edk\drivers\ide\componentname.c
efi1.1\edk\drivers\ide\driverconfiguration.c
efi1.1\edk\drivers\ide\driverdiagnostics.c
efi1.1\edk\drivers\ide\ide.c

efi1.1\edk\drivers\ide\ide.h
efi1.1\edk\drivers\ide\idebus.c
efi1.1\edk\drivers\ide\idebus.h
efi1.1\edk\drivers\ide\idedata.h
efi1.1\edk\drivers\ide\make.inf
efi1.1\edk\drivers\isabus\componentname.c
efi1.1\edk\drivers\isabus\isabus.c
efi1.1\edk\drivers\isabus\isaio.c
efi1.1\edk\drivers\isabus\make.inf
efi1.1\edk\drivers\isafloppy\componentname.c
efi1.1\edk\drivers\isafloppy\isafloppy.c
efi1.1\edk\drivers\isafloppy\isafloppy.h
efi1.1\edk\drivers\isafloppy\isafloppyblock.c
efi1.1\edk\drivers\isafloppy\isafloppyctrl.c
efi1.1\edk\drivers\isafloppy\make.inf
efi1.1\edk\drivers\isaserial\componentname.c
efi1.1\edk\drivers\isaserial\make.inf
efi1.1\edk\drivers\isaserial\serial.c
efi1.1\edk\drivers\isaserial\serial.h
efi1.1\edk\drivers\partition\gpt.c
efi1.1\edk\drivers\partition\make.inf
efi1.1\edk\drivers\partition\mbr.c
efi1.1\edk\drivers\partition\mbr.h
efi1.1\edk\drivers\partition\partition.c
efi1.1\edk\drivers\pcatisaacpi\make.inf
efi1.1\edk\drivers\pcatisaacpi\pcatisaacpi.c
efi1.1\edk\drivers\pcatisaacpibios\make.inf
efi1.1\edk\drivers\pcatisaacpibios\pcatisaacpi.c
efi1.1\edk\drivers\pcatpcirootbridge\make.inf
efi1.1\edk\drivers\pcatpcirootbridge\pcatpcirootbridge.c
efi1.1\edk\drivers\pcatpcirootbridge\pcatpcirootbridge.h
efi1.1\edk\drivers\pcatpcirootbridge\pcatpcirootbridgeio.c
efi1.1\edk\drivers\pcatpcirootbridge\ia32\pcatio.c
efi1.1\edk\drivers\pcatpcirootbridge\ipf\pcatio.c
efi1.1\edk\drivers\pcibus\make.inf
efi1.1\edk\drivers\pcibus\pcibus.c

efi1.1\edk\drivers\pcibus\pcibus.h
efi1.1\edk\drivers\pcibus\pciio.c
efi1.1\edk\drivers\pcibus\pciromtable.c
efi1.1\edk\drivers\pcibus\pciromtable.h
efi1.1\edk\drivers\pcivgaminiport\make.inf
efi1.1\edk\drivers\ps2keyboard\make.inf
efi1.1\edk\drivers\ps2keyboard\ps2kbdctrller.c
efi1.1\edk\drivers\ps2keyboard\ps2keyboard.c
efi1.1\edk\drivers\ps2keyboard\ps2keyboard.h
efi1.1\edk\drivers\ps2mouse\commps2.c
efi1.1\edk\drivers\ps2mouse\commps2.h
efi1.1\edk\drivers\ps2mouse\make.inf
efi1.1\edk\drivers\ps2mouse\ps2mouse.c
efi1.1\edk\drivers\pxebc\bc.c
efi1.1\edk\drivers\pxebc\bc.h
efi1.1\edk\drivers\pxebc\componentname.c
efi1.1\edk\drivers\pxebc\dhcp.h
efi1.1\edk\drivers\pxebc\efibis.h
efi1.1\edk\drivers\pxebc\ip.h
efi1.1\edk\drivers\pxebc\make.inf
efi1.1\edk\drivers\pxebc\pxe_bc_arp.c
efi1.1\edk\drivers\pxebc\pxe_bc_dhcp.c
efi1.1\edk\drivers\pxebc\pxe_bc_igmp.c
efi1.1\edk\drivers\pxebc\pxe_bc_ip.c
efi1.1\edk\drivers\pxebc\pxe_bc_mtftp.c
efi1.1\edk\drivers\pxebc\pxe_bc_udp.c
efi1.1\edk\drivers\pxebc\pxe_loadfile.c
efi1.1\edk\drivers\pxebc\tftp.h
efi1.1\edk\drivers\serialmouse\make.inf
efi1.1\edk\drivers\serialmouse\serialmouse.c
efi1.1\edk\drivers\snp32_64\callback.c
efi1.1\edk\drivers\snp32_64\get_status.c
efi1.1\edk\drivers\snp32_64\make.inf
efi1.1\edk\drivers\snp32_64\receive.c
efi1.1\edk\drivers\snp32_64\receive_filters.c
efi1.1\edk\drivers\snp32_64\snp.c

efi1.1\edk\drivers\snp32_64\snp.h
efi1.1\edk\drivers\snp32_64\start.c
efi1.1\edk\drivers\snp32_64\transmit.c
efi1.1\edk\drivers\undi\decode.c
efi1.1\edk\drivers\undi\e100b.c
efi1.1\edk\drivers\undi\e100b.h
efi1.1\edk\drivers\undi\init.c
efi1.1\edk\drivers\undi\make.inf
efi1.1\edk\drivers\undi\undi32.h
efi1.1\edk\drivers\usb\uhci\componentname.c
efi1.1\edk\drivers\usb\uhci\make.inf
efi1.1\edk\drivers\usb\uhci\uhchlp.c
efi1.1\edk\drivers\usb\uhci\uhci.c
efi1.1\edk\drivers\usb\uhci\uhci.h
efi1.1\edk\drivers\usb\usbbot\bot.c
efi1.1\edk\drivers\usb\usbbot\make.inf
efi1.1\edk\drivers\usb\usbbus\make.inf
efi1.1\edk\drivers\usb\usbbus\usb.c
efi1.1\edk\drivers\usb\usbbus\usbbus.c
efi1.1\edk\drivers\usb\usbbus\usbio.c
efi1.1\edk\drivers\usb\usbcbi\cbi.h
efi1.1\edk\drivers\usb\usbcbi\cbi1\cbi1.c
efi1.1\edk\drivers\usb\usbcbi\cbi1\make.inf
efi1.1\edk\drivers\usb\usbkb\efikey.c
efi1.1\edk\drivers\usb\usbkb\efikey.h
efi1.1\edk\drivers\usb\usbkb\keyboard.c
efi1.1\edk\drivers\usb\usbkb\keyboard.h
efi1.1\edk\drivers\usb\usbkb\make.inf
efi1.1\edk\drivers\usb\usbmassstorage\make.inf
efi1.1\edk\drivers\usb\usbmassstorage\usbmassstorage.c
efi1.1\edk\drivers\usb\usbmassstorage\usbmassstorage.h
efi1.1\edk\drivers\usb\usbmassstorage\usbmassstoragedata.h
efi1.1\edk\drivers\usb\usbmassstorage\usbmassstoragehelper.c
efi1.1\edk\drivers\usb\usbmouse\make.inf
efi1.1\edk\drivers\usb\usbmouse\usbmouse.c
efi1.1\edk\drivers\vgaclass\componentname.c

efi1.1\edk\drivers\vgaclass\make.inf
efi1.1\edk\drivers\vgaclass\vgaclass.c
efi1.1\edk\drivers\winntthunk\blockio\make.inf
efi1.1\edk\drivers\winntthunk\blockio\winntblockio.c
efi1.1\edk\drivers\winntthunk\console\console.c
efi1.1\edk\drivers\winntthunk\console\make.inf
efi1.1\edk\drivers\winntthunk\serialio\make.inf
efi1.1\edk\drivers\winntthunk\serialio\winntserialio.c
efi1.1\edk\drivers\winntthunk\simplefilesystem\make.inf
efi1.1\edk\drivers\winntthunk\simplefilesystem\winntsimplefilesystem.c
efi1.1\edk\drivers\winntthunk\uga\make.inf
efi1.1\edk\drivers\winntthunk\uga\winntugadriver.c
efi1.1\edk\drivers\winntthunk\uga\winntugascreen.c
efi1.1\edk\drivers\winntthunk\winntbusdriver\make.inf
efi1.1\edk\drivers\winntthunk\winntbusdriver\winntbusdriver.c
efi1.1\edk\drivers\winntthunk\winntpcirootbridge\make.inf
efi1.1\edk\drivers\winntthunk\winntpcirootbridge\winntpcirootbridge.c
efi1.1\edk\drivers\winntthunk\winntpcirootbridge\winntpcirootbridgedevicepath.c
efi1.1\edk\drivers\winntthunk\winntpcirootbridge\winntpcirootbridgeio.c
efi1.1\edk\guid\make.inf
efi1.1\edk\include\efi.h
efi1.1\edk\include\efiapi.h
efi1.1\edk\include\efidebug.h
efi1.1\edk\include\efidevicepath.h
efi1.1\edk\include\efierror.h
efi1.1\edk\include\efipxe.h
efi1.1\edk\include\efistdarg.h
efi1.1\edk\include\efitypes.h
efi1.1\edk\include\pci22.h
efi1.1\edk\include\usb.h
efi1.1\edk\include\ebc\efibind.h
efi1.1\edk\include\ia32\efibind.h
efi1.1\edk\include\ia32\efiimage.h
efi1.1\edk\include\ipf\efibind.h
efi1.1\edk\include\ipf\efiimage.h
efi1.1\edk\include\ipf\salproc.h

efi1.1\edk\lib\efidriverlib\debug.c
efi1.1\edk\lib\efidriverlib\devicepath.c
efi1.1\edk\lib\efidriverlib\libglobals.c
efi1.1\edk\lib\efidriverlib\make.inf
efi1.1\edk\lib\include\efidriverlib.h
efi1.1\edk\lib\include\efiprintlib.h
efi1.1\edk\lib\print\make.inf
efi1.1\edk\lib\print\print.c
efi1.1\edk\protocol\make.inf
efi1.1\edk\protocol\blockio\blockio.h
efi1.1\edk\protocol\debugsupport\debugsupport.h
efi1.1\edk\protocol\efinetworkinterfaceidentifier\efinetworkinterfaceidentifier.c
efi1.1\edk\protocol\efinetworkinterfaceidentifier\efinetworkinterfaceidentifier.h
efi1.1\edk\protocol\loadfile\loadfile.h
efi1.1\edk\protocol\pciio\pciio.h
efi1.1\edk\protocol\pxebasecode\pxebasecode.c
efi1.1\edk\protocol\pxebasecode\pxebasecode.h
efi1.1\edk\protocol\pxebasecodecallback\pxebasecodecallback.c
efi1.1\edk\protocol\pxebasecodecallback\pxebasecodecallback.h
efi1.1\edk\protocol\scsipassthru\scsipassthru.h
efi1.1\edk\protocol\serialio\serialio.h
efi1.1\edk\protocol\simplenetwork\simplenetwork.h
efi1.1\edk\protocol\simpletextin\simpletextin.h
efi1.1\edk\protocol\simpletextout\simpletextout.h
efi1.1\edk\protocol\ugadraw\ugadraw.h
efi1.1\edk\protocol\unicodecollation\unicodecollation.h
efi1.1\edk\protocol\usbatapi\usbatapi.h
efi1.1\edk\protocol\usbhostcontroller\usbhostcontroller.h
efi1.1\edk\protocol\usbio\usbio.h
efi1.1\edk\protocol\winntthunk\winntthunk.h
efi1.1\inc\efi.h
efi1.1\inc\efi_nii.h
efi1.1\inc\efi_pxe.h
efi1.1\inc\efidevp.h
efi1.1\inc\efierr.h
efi1.1\inc\efilib.h

efi1.1\inc\pci22.h
efi1.1\inc\usb.h
efi1.1\lib\error.c
efi1.1\lib\guid.c
efi1.1\lib\runtime\lock.c
efi1.1\lib\runtime\rtdata.c
efi1.1\lib\runtime\str.c
efi1.1\notes\relnote.doc
efi1.1\shell\attrib\attrib.c
efi1.1\shell\attrib\make.inf
efi1.1\shell\bcfg\bcfg.c
efi1.1\shell\bcfg\make.inf
efi1.1\shell\cls\cls.c
efi1.1\shell\cls\make.inf
efi1.1\shell\comp\comp.c
efi1.1\shell\comp\make.inf
efi1.1\shell\cp\cp.c
efi1.1\shell\cp\make.inf
efi1.1\shell\date\date.c
efi1.1\shell\date\make.inf
efi1.1\shell\debug\dblk.c
efi1.1\shell\debug\efidump.c
efi1.1\shell\debug\make.inf
efi1.1\shell\dmpstore\dmpstore.c
efi1.1\shell\dmpstore\make.inf
efi1.1\shell\edit\editortype.h
efi1.1\shell\edit\libeditor.c
efi1.1\shell\edit\libfilebuffer.c
efi1.1\shell\edit\libinputbar.c
efi1.1\shell\edit\libmenubar.c
efi1.1\shell\edit\libmisc.c
efi1.1\shell\edit\libstatusbar.c
efi1.1\shell\edit\libtitlebar.c
efi1.1\shell\edit\main.c
efi1.1\shell\edit\make.inf
efi1.1\shell\eficompress\compress.c

efi1.1\shell\eficompress\make.inf
efi1.1\shell\efidecompress\make.inf
efi1.1\shell\err\make.inf
efi1.1\shell\getmtc\make.inf
efi1.1\shell\helpdata\helpdata.src
efi1.1\shell\hexedit\heditortype.h
efi1.1\shell\hexedit\libbufferimage.c
efi1.1\shell\hexedit\libclipboard.c
efi1.1\shell\hexedit\libdiskimage.c
efi1.1\shell\hexedit\libeditor.c
efi1.1\shell\hexedit\libfileimage.c
efi1.1\shell\hexedit\libinputbar.c
efi1.1\shell\hexedit\libmemimage.c
efi1.1\shell\hexedit\libmenubar.c
efi1.1\shell\hexedit\libmisc.c
efi1.1\shell\hexedit\libstatusbar.c
efi1.1\shell\hexedit\libtitlebar.c
efi1.1\shell\hexedit\main.c
efi1.1\shell\hexedit\make.inf
efi1.1\shell\inc\shell.h
efi1.1\shell\inc\shellenv.h
efi1.1\shell\iomod\iomod.c
efi1.1\shell\iomod\make.inf
efi1.1\shell\lib\helpinfo.c
efi1.1\shell\lib\io.c
efi1.1\shell\lib\misc.c
efi1.1\shell\load\load.c
efi1.1\shell\load\make.inf
efi1.1\shell\loadbmp\loadbmp.c
efi1.1\shell\loadbmp\make.inf
efi1.1\shell\loadpcirom\loadpcirom.c
efi1.1\shell\loadpcirom\make.inf
efi1.1\shell\ls\ls.c
efi1.1\shell\ls\make.inf
efi1.1\shell\mem\make.inf
efi1.1\shell\mem\mem.c

efi1.1\shell\memmap\make.inf
efi1.1\shell\memmap\memmap.c
efi1.1\shell\mkdir\make.inf
efi1.1\shell\mode\make.inf
efi1.1\shell\mv\make.inf
efi1.1\shell\mv\mv.c
efi1.1\shell\newshell\init.c
efi1.1\shell\newshell\make.inf
efi1.1\shell\pci\make.inf
efi1.1\shell\pci\pci.c
efi1.1\shell\pci\pci_class.c
efi1.1\shell\reset\make.inf
efi1.1\shell\reset\reset.c
efi1.1\shell\rm\make.inf
efi1.1\shell\rm\rm.c
efi1.1\shell\setsize\make.inf
efi1.1\shell\setsize\setsize.c
efi1.1\shell\shellenv\batch.c
efi1.1\shell\shellenv\conio.c
efi1.1\shell\shellenv\connect.c
efi1.1\shell\shellenv\data.c
efi1.1\shell\shellenv\dprot.c
efi1.1\shell\shellenv\exec.c
efi1.1\shell\shellenv\help.c
efi1.1\shell\shellenv\init.c
efi1.1\shell\shellenv\make.inf
efi1.1\shell\shellenv\map.c
efi1.1\shell\shellenv\marg.c
efi1.1\shell\shellenv\mount.c
efi1.1\shell\shellenv\pause.c
efi1.1\shell\shellenv\protid.c
efi1.1\shell\shellenv\shelle.h
efi1.1\shell\shellenv\var.c
efi1.1\shell\stall\make.inf
efi1.1\shell\time\make.inf
efi1.1\shell\time\time.c

efi1.1\shell\touch\make.inf

efi1.1\shell\touch\touch.c

efi1.1\shell\type\make.inf

efi1.1\shell\type\type.c

efi1.1\shell\ver\make.inf

efi1.1\shell\ver\ipf\ver64.c

efi1.1\shell\vol\make.inf

efi1.1\shell\vol\vol.c

B.5 EFI 1.10.14.56

B.5.1 Core Bug Fixes

1. Updated copyright statements from 2001 to 2002. This change affected every source file in the build.
2. Added GraphicsConsole to Ia32Drivers makefile.
3. Added GraphicsConsole to IpfDrivers makefile.
4. EFIROM : Updated to compute byte checksum of legacy images
5. GENMAKE : Added support for EBC subdirectories
6. AllocatePages() : Fixed very rare memory allocation ASSERT().
7. CopyMem() : Fixed reentrancy bug because MM0 was not being preserved
8. ZeroMem() : Fixed reentrancy bug because MM0 was not being preserved
9. Cirrus Logic Driver: Optimized for improved EBC performance.
10. Graphics Console Driver: Optimized for improved EBC performance.
11. Fixed bug in serial driver to be able to use baud rates other than 15200.
12. Updated DebugPort Driver to follow the EFI 1.10 Driver Model.
13. Fixed several bugs in DebugSupport Driver.
14. Fixed alignment fault in the EBC interpreter.
15. Updated the EBC DebugSupport Protocol.
16. Optimized the IA-32 EBC Interpreter by using the /O2 compiler switch
17. Fixed SNP driver to work on Itanium-based systems with > 4 GB of memory.
18. Updated the version from EFI 1.10.14.54 to EFI 1.10.14.56.
19. Updated the EFI Driver Lib to support EBC.
20. Updated DebugPort Protocol definition.
21. Updated DebugSupport Protocol definition.
22. Fixed a hang in EFI Shell Command “Connect.”

B.5.2 List of New Files

\efi1.1\build\EbcDrivers\Build.cmd

\efi1.1\build\EbcDrivers\Makefile

\efi1.1\build\EbcDrivers\Master.env

\efi1.1\build\EbcDrivers\Master.mak

\efi1.1\edk\Drivers\DebugPort\ComponentName.c

\efi1.1\edk\Drivers\DebugPort\DebugPort.h

\efi1.1\edk\Include\Ebc\EfiImage.h

\efi1.1\edk\Lib\EfiDriverLib\Ebc\EbcMath.c

B.5.3 List of Modified Files

\efi1.1\notes\Relnote.doc

\efi1.1\inc\efi.h

\efi1.1\edk\include\efi.h

\efi1.1\build\ia32drivers\makefile

\efi1.1\build\ipfdrivers\makefile

\efi1.1\build\tools\src\efirom\efirom.c

\efi1.1\build\tools\src\genmake\genmake.c

\efi1.1\build\tools\bin\EFIBOOT.IMG

\efi1.1\corefw\fw\efi\mem\page.c

\efi1.1\corefw\fw\efi\misc\EfiCoreCopyMem.asm

\efi1.1\corefw\fw\efi\misc\EfiCoreZeroMem.asm

\efi1.1\edk\drivers\CirrusLogic5430\CirrusLogic5430.c

\efi1.1\edk\drivers\CirrusLogic5430\CirrusLogic5430.h

\efi1.1\edk\drivers\CirrusLogic5430\CirrusLogic5430UgaDraw.c

\efi1.1\edk\drivers\Console\GraphicsConsole\GraphicsConsole.c

\efi1.1\edk\drivers\Cpnsole\GraphicsConsole\GraphicsConsole.h

\efi1.1\edk\drivers\Console\Terminal\Terminal.c

\efi1.1\edk\drivers\IsaSerial\Serial.c

\efi1.1\edk\drivers\IsaSerial\Serial.h

\efi1.1\edk\drivers\WinNtThunk\SerialIo\WinNtSerialIo.c

\efi1.1\edk\drivers\WinNtThunk\SerialIo\WinNtSerialIo.h

\efi1.1\edk\drivers\DebugPort\DebugPort.c

\efi1.1\edk\drivers\DebugPort\make.inf

\efi1.1\edk\drivers\DebugSupport\ia32\AsmFuncs.asm

\efi1.1\edk\drivers\DebugSupport\ia32\PlDebugSupport.c

\efi1.1\edk\drivers\DebugSupport\ia32\PlDebugSupport.h

\efi1.1\edk\drivers\DebugSupport\ipf\PlDebugSupport.c

\efi1.1\edk\drivers\DebugSupport\ipf\PlDebugSupport.h

\efi1.1\edk\drivers\Ebc\EbcExecute.c

\efi1.1\edk\drivers\Ebc\EbcExecute.c

\efi1.1\edk\drivers\Ebc\EbcInt.c
\efi1.1\edk\drivers\Ebc\EbcInt.h
\efi1.1\edk\drivers\Ebc\Make.inf
\efi1.1\edk\drivers\snp32_64\receive.c
\efi1.1\edk\drivers\snp32_64\snp.c
\efi1.1\edk\drivers\snp32_64\snp.h
\efi1.1\edk\drivers\snp32_64\transmit.c
\efi1.1\edk\lib\EfiDriverLib\Make.inf
\efi1.1\efk\Protocol\DebugPort\DebugPort.h
\efi1.1\efk\Protocol\DebugSupport\DebugSupport.h
\efi1.1\shell\ShellEnv\Connect.c