intel.

EFI 1.1 Shell Commands Specification

Version 0.3

January 3, 2003

intel.

Revision History

Revision	Revision History	Date	Author
0.1	Initial review draft	08/01/01	Intel Corporation
0.3	Review draft.	01/03/03	Intel Corporation

intel.

Contents

Tables

1
Introduction

The Extensible Firmware Interface (EFI) Specification describes a set of Application Program Interfaces (APIs) and data structures that are exported by a system's firmware. Software that takes advantage of these APIs and data structures may take one of many forms. These include an EFI device driver, an EFI shell, an EFI system utility, an EFI system diagnostic, or an Operating System (OS) loader. In addition, the EFI Specification describes a set of run-time services that are available to an OS while the OS has full control of the system.

The sample implementation of the EFI Specification includes an EFI Shell. An EFI Shell is a special type of EFI Application that allows other EFI Applications to be launched. The combination of the EFI firmware and the EFI Shell provide an environment that can be modified to easily adapt to many different hardware configurations. The EFI shell is a simple, interactive environment that allows EFI device drivers to be loaded, EFI applications to be launched, and operating systems to be booted. In addition, the shell also provides a set of basic commands used to manage files and the system environment variables. This document describes the capabilities of the EFI Shell and the EFI Shell commands in detail.

1.1   Organization of this Document

This specification is organized as follows:

Table 1-1. Specification Organization and Contents

Chapter	Description
Chapter 1: Introduction	Introduction to EFI shell environment.
Chapter 2: EFI Shell Commands	Descriptions of EFI shell commands.

1.2   Goals

The primary goal of this document is to provide an overview of the EFI shell commands and their capabilities.

1.3   Target Audience

This document is intended for the following readers:

- General users of the EFI shell, and developers who will be utilizing and writing test and diagnostic scripts running in the EFI environment.

1.4   Related Information

The following publications and sources of information may be useful to you or are referred to by this specification:

- *Extensible Firmware Interface Specification*, Version 1.02, Intel Corporation, 2000.

1.5   Conventions Used in This Document

This document uses typographic and illustrative conventions described below.

1.5.1   Typographic Conventions

The following typographic conventions are used in this document to illustrate programming concepts:

`Prototype`	This typeface is use to indicate prototype code.
`Argument`	This typeface is used to indicate arguments.
`Name`	This typeface is used to indicate actual code or a programming construct.
register	This typeface is used to indicate a processor register.

int͞el

<div align="right">

2
EFI Shell and Commands

</div>

This section describes the features of the EFI Shell and the available shell commands. The EFI Shell supports a command line interface as well as batch scripting.

2.1   Invocation of the EFI Shell

When the EFI Shell is invoked, it first looks for commands in the file **startup.nsh** on the execution path defined by the environment. There is no requirement for a startup file to exist. Once the startup file commands are completed, the shell looks for commands from console input device.

2.2   EFI Shell Command Syntax

The EFI Shell implements a programming language that provides control over the execution of individual commands. When the shell scans its input, it always treats certain characters specially: (**#**, **>**, **%**, *****, **?**, **[**, **^**, **space**, and **newline**) . Care should be exercised in the use of these characters.

When a command contains a defined alias the shell replaces the alias with its definition (see **alias** command in this chapter). If the argument is prefixed with the **^** character, however, the argument is treated as a literal argument and alias processing is not performed.

In interactive execution, the shell performs variable substitution, then expands wildcards before the command is executed. In batch script execution, the shell performs argument substitution, then variable substitution, then expands wildcards before the command is executed.

2.2.1   Variable Substitution

Environment variables can be set and viewed through the use of the **set** command (see **set** command in this chapter). To access the value of an environment variable as an argument to a shell command, delimit the name of the variable with the **%** character before and after the variable name; for example, *%myvariable%*.

The shell maintains a special variable, named **lasterror**. The variable contains the return code of the most recently executed shell command.

2.2.2   Wildcard Expansion

The ***** , **?** and **[** characters can be used as wildcard characters in filename arguments to shell commands. If an argument contains one or more of these characters, the shell processes the argument for *file meta-arguments* and expands the argument list to include all filenames matching the pattern. These characters are part of patterns which represent file and directory names.

Table 2-1. Wildcard Character Expansion

Character Sequence	Meaning
"*"	Matches zero or more characters in a file name
"?"	Matches exactly one character of a file name
"[*chars*]"	Defines a set of characters; the pattern matches any single character in the set. Characters in the set are not separated. Ranges of characters can be specified by specifying the first character in a range, then the – character, then the last character in the range. Example: [a-zA-Z]

2.2.3 Output Redirection

Output of EFI Shell commands can be redirected to files. The syntax of this is as follows:

```
Command > unicode_output_file_pathname
Command >a ascii_output_file_pathname
Command 1> unicode_output_file_pathname
Command 1>a ascii_output_file_pathname
Command 2> unicode_output_file_pathname
Command 2>a ascii_output_file_pathname
Command >> unicode_output_file_pathname
Command >>a ascii_output_file_pathname
Command 1>> unicode_output_file_pathname
Command 1>>a ascii_output_file_pathname
```

The meanings of the special character sequences used to denote output redirection are shown in Table 2-1.

Table 2-2. Output Redirection Syntax

Character Sequence	Meaning
">"	redirect standard output to a unicode file
">a"	redirect standard output to an ascii file
"1>"	redirect standard output to a unicode file
"1>a"	redirect standard output to an ascii file
"2>"	redirect standard error to unicode file
"2>a"	redirect standard error to ascii file
">>"	redirect standard output appended to a unicode file
">>a"	redirect standard output appended to an ascii file
"1>>"	redirect standard output appended to a unicode file
"1>>a"	redirect standard output appended to an ascii file

The shell will redirect standard output to a single file and standard error to a single file. Redirecting both standard output and standard error to the same file is allowed. Redirecting Standard output to more than one file on the same command is not supported. Similarly, redirecting to multiple files is not supported for standard error.

2.2.4   Quoting

Quotation marks in the EFI Shell are used for argument grouping. A quoted string is treated as a single argument to a command, and any whitespace characters included in the quoted string are just part of that single argument. Quoting an environment variable does not have any effect on the de-referencing of that variable. Double quotation marks **""** are used to denote strings. Single quotation marks are not treated specially by the shell in any way. Empty strings are treated as valid command line arguments.

2.2.5   Execution of Batch Scripts

The EFI Shell has the capability of executing commands from a file (batch script). EFI Shell batch script files are named using the ".nsh" extension. Batch script files can be either UNICODE or ASCII format files. EFI Shell script files are invoked by entering the filename at the command prompt, with or without the filename extension.

Up to nine (9) positional arguments are supported for batch scripts. Positional argument substitution is performed before the execution of each line in the script file. Positional arguments are denoted by **%n**, where n is a digit between 0 and 9. By convention, **%0** is the name of the script file currently being executed. In batch scripts, argument substitution is performed first, then variable substitution. Thus, for a variable containing **%2**, the variable will be replaced with the literal string **%2**, not the second argument on the command line.

If no real argument is found to substitute for a positional argument, then the positional argument is ignored.

Script file execution can be nested; that is, script files may be executed from within other script files. Recursion is allowed.

Output redirection is fully supported. Output redirection on a command in a script file causes the output for that command to be redirected. Output redirection on the invocation of a batch script causes the output for all commands executed from that batch script to be redirected to the file, with the output of each command appended to the end of the file.

By default, both the input and output for all commands executed from a batch script are echoed to the console. Display of commands read from a batch file can be suppressed via the **echo –off** command (see **echo**). If output for a command is redirected to a file, then that output is not displayed on the console. Note that commands executed from a batch script are not saved by the shell for DOSkey history (up-arrow command recall).

2.2.6   Error Handling in Batch Scripts

By default, if an error is encountered during the execution of a command in a batch script, the script will continue to execute. The **lasterror** shell variable is provided allow batch scripts to test the results of the most recently executed command using the **if** command. This variable is not an

environment variable, but is a special variable maintained by the shell for the lifetime of that instance of the shell.

2.2.7   Comments in Script Files

Comments can be embedded in batch scripts. The # character on a line is used to denote that all characters on the same line and to the right of the # are to be ignored by the shell. Comments are not echoed to the console.

2.3   EFI Shell Commands

Most shell commands can be invoked from the EFI shell prompt. However there are several commands that are only available for use from within batch script files. Table 2-3 provides a list of all the commands. The "Batch-only" column indicates if the command is only available from within script files. The following sections provide more details on each of the individual commands.

Table 2-3.   List of EFI Shell Commands

Command	Batch-only	Description
alias	No	Displays, creates, or deletes aliases in the EFI shell
attrib	No	Displays or changes the attributes of files or directories
bcfg	No	Displays/modifies the driver/boot configuration
break	No	Executes a debugger break point
cd	No	Displays or changes the current directory
cls	No	Clears the standard output with an optional background color
comp	No	Compares the contents of two files
connect	No	Binds an EFI driver to a device and starts the driver
cp	No	Copies one or more files/directories to another location
date	No	Displays the current date or sets the date in the system
dblk	No	Displays the contents of blocks from a block device
devices	No	Displays the list of devices being managed by EFI drivers
devtree	No	Displays the tree of devices that follow the EFI Driver Model
dh	No	Displays the handles in the EFI environment
disconnect	No	Disconnects one or more drivers from a device
dmem	No	Displays the contents of memory
dmpstore	No	Displays all NVRAM variables
drivers	No	Displays the list of drivers that follow the EFI Driver Model
drvcfg	No	Invokes the Driver Configuration Protocol
drvdiag	No	Invokes the Driver Diagnostics Protocol
echo	No	Displays messages or turns command echoing on or off
edit	No	Edits an ASCII or UNICODE file in full screen.
EfiCompress	No	Compress a file
EfiDecompress	No	Decompress a file
err	No	Displays or changes the error level
exit	No	Exits the EFI Shell
for/endfor	Yes	Executes commands for each item in a set of items
getmtc	No	Displays the current monotonic counter value
goto	Yes	Makes batch file execution jump to another location

Command	Batch-only	Description
guid	No	Displays all the GUIDs in the EFI environment
help	No	Displays commands list or verbose help of a command
hexedit	No	Edits with hex mode in full screen
If/endif	Yes	Executes commands in specified conditions
load	No	Loads EFI drivers
LoadBmp	No	Displays a Bitmap file onto the screen
LoadPciRom	No	Loads a PCI Option ROM image from a file
ls	No	Displays a list of files and subdirectories in a directory
map	No	Displays or defines mappings
memmap	No	Displays the memory map
mkdir	No	Creates one or more directories
mm	No	Displays or modifies MEM/IO/PCI
mode	No	Displays or changes the mode of the console output device
mount	No	Mounts a file system on a block device
mv	No	Moves one or more files/directories to destination
OpenInfo	No	Displays the protocols on a handle and the agents
pause	No	Prints a message and suspends for keyboard input
pci	No	Displays PCI devices or PCI function configuration space
reconnect	No	Reconnects one or more drivers from a device
reset	No	Resets the system
rm	No	Deletes one or more files or directories
set	No	Displays, creates, changes or deletes EFI environment variables
setsize	No	Sets the size of a file
stall	No	Stalls the processor for some microseconds
time	No	Displays the current time or sets the time of the system
touch	No	Sets the time and date of a file to the current time and date
type	No	Displays the contents of a file
unload	No	Unloads a protocol image
ver	No	Displays the version information
vol	No	Displays volume information of the file system

2.3.1　alias

```
alias [-d|-v|-b][sname][value]

    -d        - Deletes an alias
    -v        - Volatile variable
    -b        - Displays one screen at a time
    sname     - Alias name
    value     - Original name

 Note:
    1. 'sname' shall not be an EFI shell command or a device mapping name.
    2. 'value' shall be an EFI shell command or an EFI application.
    3. ALIAS values are stored in EFI NVRAM and will be retained between boots
       unless the option -v is specified.
```

This command displays, creates, or deletes aliases in the EFI shell environment. An alias provides a new name for an existing EFI shell command or an EFI application. Once the alias is created, it can be used to run the command or launch the EFI application. There are some aliases that are predefined in the EFI shell environment. These aliases provide the DOS and UNIX equivalent names for the file manipulation commands. The example below shows typical output from help or this command.

Examples

```
    * To display all aliases in the current EFI environment:
      Shell> alias
          dir  : ls
          md   : mkdir
          rd   : rm
          del  : rm
          copy : cp


    * To create an alias to the EFI environment:
      Shell> alias myguid guid
      Shell> alias
          dir    : ls
          md     : mkdir
          rd     : rm
          del    : rm
          copy   : cp
          myguid : guid


    * To delete an alias in the EFI environment:
      Shell> alias -d myguid
      Shell> alias
          dir    : ls
          md     : mkdir
```

```
        rd      : rm
        del     : rm
        copy    : cp


  * To add a volatile alias in current EFI environment, which has a star at
  * the line head. And this volatile alias will disappear at next boot.
    Shell> alias -v fs0 floppy
    Shell> alias
        dir   : ls
        md    : mkdir
        rd    : rm
        del   : rm
        copy : cp
      * fs0   : floppy


  * To add an alias with parameters:
    Shell> alias "dir /p" "ls -b"
    Shell> alias
        dir     : ls
        md      : mkdir
        rd      : rm
        del     : rm
        copy    : cp
        dir /p : ls -b
    Shell> "dir /p"
```

2.3.2 attrib

```
attrib [+a|-a][+s|-s][+h|-h][+r|-r][-b][file...][directory...]


    +a|-a       - Sets or clears 'archive' attribute
    +s|-s       - Sets or clears 'system' attribute
    +h|-h       - Sets or clears 'hidden' attribute
    +r|-r       - Sets or clears 'read only' attribute
    -b          - Displays one screen at a time
    file        - File name (wildcards are permitted)
    directory   - Directory name (wildcards are permitted)
```

Displays or sets file attributes. There are four attribute types that are supported for the EFI File System. These are archive[A], system[S], hidden[H], and read only[R]. If a file is a directory, then it is also shown to have the attribute [D].

Examples

```
* To display the attributes of a directory:
  fs0:\> attrib fs0:\
  D    fs0:\

* To add system attribute to all files of extension '.efi':
  fs0:\> attrib +s *.efi

* To display attributes of all files/directories in current directory:
  fs0:\> attrib *
   AS   fs0:\serial.efi
  DA    fs0:\test1
   A HR fs0:\bios.inf
   A    fs0:\VerboseHelp.txt
   AS   fs0:\IsaBus.efi

* To remove attributes of files, using -a,-s,-h,-r option:
  fs0:\> attrib -r *.inf
   AS    fs0:\serial.efi
  DA     fs0:\test1
   A H   fs0:\bios.inf
   A     fs0:\VerboseHelp.txt
   AS    fs0:\IsaBus.efi
```

2.3.3   bcfg

```
BCFG driver|boot [dump [-v]] [add # file "desc"] [rm #] [mv # #]


    driver         - Display/modify the driver option list
    boot           - Display/modify the boot option list
    dump           - Display the option list
    -v             - Display the option list with extra info
    add            - Add an option
    #              - The number of the option to add in hex
    file           - The file name of the EFI application/driver for the option
    "desc"         - The description of the option being added
    rm             - Remove an option
    #              - The number of the option to remove in hex
    mv             - Move an option
    #              - The number of the option to move in hex
    #              - The new number of the option being moved
```

Manages the boot and driver options stored in NVRAM. This command can display the
Boot#### or **Driver####** environment variables by using the **dump** option. The **add** option
can be used to add a new **Boot####** or **Driver####** environment variable. The **rm** option can
be used to delete a **Boot####** or **Driver####** environment variable, and finally, then **mv** option
can be used to reorder the **Boot####** and **Driver####** environment variables. The **add**, **rm**,
and **mv** options also update the **BootOrder** or **DriverOrder** environment variables as
appropriate. The following example shows typical output from help for this command.

Examples

```
* To display driver options:
  Shell> bcfg driver dump


* To display boot options:
  Shell> bcfg boot dump


* To display verbosely of boot options:
  Shell> bcfg boot dump -v


* To add a driver option #5
  Shell> bcfg driver add 5 mydriver.efi "My Driver"


* To add a boot option #3
  Shell> bcfg boot add 3 osloader.efi "My OS"


* To remove boot option #3
  Shell> bcfg boot rm 3
```

```
* To move boot option #3 to boot option #7
  Shell> bcfg boot mv 3 7
```

2.3.4   break

break

This command is used to execute a debugger breakpoint. The code executed is EFI_BREAKPOINT(), which is only valid during check builds. The effect of this command will be different depending on the target system. Under the best of circumstances this command will cause a debugger to be invoked and pull up the source code where the EFI_BREAKPOINT() was invoked.

Examples

Shell> break

2.3.5   cd

```
CD [path]
CD [..]

Note:
    1. Type CD without parameters to display the current fs and directory.
    2. Type "CD .." to change to the parent directory, and pay attention to
       the space after CD is required.
    3. CD shall be used in the same volume.
```

This command changes the current working directory used by the EFI shell environment. The EFI shell environment uses the directory name "." to refer to the current directory, and the directory name ".." to refer to the directory's parent. The following example shows typical output for help on this command.

Examples

```
* To change current fs to the mapped fs0:
  Shell> fs0:

* To change the current directory to subdirectory 'efi':
  fs0:\> cd efi

* To change the current directory to the parent dirctory(fs0:\):
  fs0:\efi\> cd ..

* To change the current directory to 'fs0:\efi\tools':
  fs0:\> cd efi\tools

* To change the current directory to the root of current fs(fs0):
  fs0:\efi\tools\> cd \
  fs0:\>

* To change volumes with cd will not work!! For example:
  fs0:\efi\tools\> cd fs1:\  !!!! will not work !!!!
  must first type fs1: then cd to desired directory

* Moving between volumes, the current path is maintained.
  fs0:\> cd \efi\tools
  fs0:\efi\tools\> fs1:
  fs1:\> cd tmp
  fs1:\tmp> cp fs0:*.* .
  copies all of files in fs0:\efi\tools into fs1:\tmp directory
```

2.3.6   cls

```
CLS [color]

    color      - New background color
                 0    - Black
                 1    - Blue
                 2    - Green
                 3    - Cyan
                 4    - Red
                 5    - Magenta
                 6    - Yellow
                 7    - Light gray

  Note:
      1. Type CLS without parameters to clear the stand output device, the
         background color is not changed.
      2. If background color is out of range (0-7), black will be set as default.
```

This command clears the standard output device with an optional background color attribute. If color is not specified, then the background is cleared to black. The following example shows typical output for help on this command.

Examples

```
  * To clear the output but not to change the background color:
    fs0:\> cls

  * To clear the output and change the background color to Cyan:
    fs0:\> cls 3

  * To clear the output and change the background with default color(black):
    fs0:\> cls 10

  * The effect of above command line is same as:
    fs0:\> cls 0
```

2.3.7   comp

```
COMP file1 file2

    file   - File name (directory name or wildcards are not permitted)

 Note:
    1. COMP will exit immediately if the lengths of the compared files are
       different.
    2. COMP will exit if 10 differences encountered.
```

Compares the contents of file1 to file2. The first 10 differences are shown before the compare operation is terminated. The following example shows typical output for help on this command.

Examples

```
* To compare two files with different length:
  fs0:\> comp bios.inf legacy.inf
  Compare fs0:\bios.inf to fs0:\legacy.inf
  Difference #1: File sizes mismatch
  [difference(s) encountered]

* To compare two files with the same contents:
  fs0:\> comp bios.inf rafter.inf
  Compare fs0:\bios.inf to fs0:\rafter.inf
  [no difference encountered]

* To compare two files with the same length but different contents:
  fs0:\> comp bios.inf bios2.inf
  Compare fs0:\bios.inf to fs0:\bios2.inf
  Difference #1: File1: fs0:\bios.inf
   00000000: 5F                                        *_*
  File2: fs0:\bios2.inf
   00000000: 33                                        *3*
  Difference #2: File1: fs0:\bios.inf
   0000000C: 00 00 00 00                               *....*
  File2: fs0:\bios2.inf
   0000000C: 25 32 03 03                               *%2..*
  [difference(s) encountered]
```

2.3.8   connect

```
connect [-r] Handle# | DeviceHandle# DriverHandle#


    -r              - Connect recursively
    -c              - Connect console devices described in the
                       EFI Environment Varibles
    Handle#         - Device handle or Driver handle (hex)
    DeviceHandle#   - Device handle (hex)
    DriverHandle#   - Driver handle (hex)


Note:
    1. Recursive option causes EFI to scan all handles and checks to see if any
       loaded or embedded driver's SUPPORT function matches with the device.
       The drive's START function is called. If the driver's START function
       creates more device handles, these handles will also be checked to see
       if a matching driver can bind to these devices as well. The process is
       repeated until no more drivers are able to connect to any devices.
    2. If only a single handle is specified and that handle has an
       EFI_DRIVER_BINDING_PROTOCOL on that handle, then the handle is assumed
       to be a driver handle.  Otherwise, it is assumed to be a device handle.
    3. A driver handle may have an 'Image' in the 'dh' output information.
       A device driver may have a 'DevPath' in the 'dh' output.
```

This command is used to bind a driver to a specific device, and start the driver. If the **–r** option is used, then the connect is done recursively until no further connections between devices and drivers are made. The example below is typical output from help for this command.

Examples

```
* To connect all drivers to all devices recursively:
  Shell> connect -r
  ConnectController(1) : Status = Success
  ConnectController(2) : Status = Success
  ConnectController(3) : Status = Success
  ...
  ConnectController(3D) : Status = Success

* To connect driver 17 to all the devices it can manage:
  Shell> connect 17

* To connect all possible drivers to device 19:
  Shell> connect 19

* To connect driver 17 to device 19:
  Shell> connect 19 17
```

2.3.9　cp

Copies one or more files/directories to another location.

CP [-r][-q] src [src...] [dst]

    -r   - Recursive copy
    -q   - Quiet copying (replace existing files without prompt)
    src   - Source file/directory name (wildcards are permitted)
    dst   - Destination file/directory name (wildcards are not permitted)

Note:
    1. If dst is not specified, current directory is assumed to be the dst.
    2. 'CP -r src1 src2 dst' is copy all files and subdirectories in 'src1'
       and 'src2' to the 'dst', 'src1' and 'src2' themselves are not copied.
    3. Copies a directory to itself is not allowed (eg: cp -r test* test).
    4. If error occurs, CP will exit immediately and the remaining files or
       directories will not be copied.
    5. To remove directories please refer to RM.
    6. When 'cp' is executed within a script file, it always performs quiet
       copying regardless of whether '-q' is specified.

This command copies one or more files from one location to another location. The following example shows how to copy the file MEMMAP.EFI in the TOOLS directory on the floppy drive to the file MM.EFI. The following example shows typical output for help on this command.

Examples

* To display the contents of current directory first of all:
  fs0:\> ls
  Directory of: fs0:\

    06/18/01 01:02p <DIR>   512 efi
    06/18/01 01:02p <DIR>   512 test1
    06/18/01 01:02p <DIR>   512 test2
    06/13/01 10:00a   28,739 IsaBus.efi
    06/13/01 10:00a   32,838 IsaSerial.efi
    06/18/01 08:04p   29 temp.txt
    06/18/01 08:05p <DIR>   512 test
          3 File(s)   61,606 bytes
          4 Dir(s)

* To copy a file in the same directory as different file name:
  fs0:\> cp temp.txt readme.txt
  copying fs0:\temp.txt -> fs0:\readme.txt
  - [ok]

* To copy multiple files to another directory:

```
fs0:\> cp temp.txt isaBus.efi \test
copying fs0:\temp.txt -> fs0:\test\temp.txt
 - [ok]
copying fs0:\isaBus.efi -> fs0:\test\IsaBus.efi
 - [ok]
```

```
* To copy multiple dirctories recursively to another directory:
  fs0:\> cp -r test1 test2 efi \test
  copying fs0:\test1\test1.txt -> fs0:\test\test1.txt
   - [ok]
  copying fs0:\test2\test2.txt -> fs0:\test\test2.txt
   - [ok]
  making dir fs0:\test\boot
  copying fs0:\efi\boot\nshell.efi -> fs0:\test\boot\nshell.efi
   - [ok]
```

```
* To see the results of above operations:
  fs0:\> ls \test
  Directory of: fs0:\test
```

```
  06/18/01  01:01p <DIR>           512  .
  06/18/01  01:01p <DIR>             0  ..
  01/28/01  08:21p                  30  test1.txt
  01/28/01  08:21p                  30  test2.txt
  01/28/01  08:21p <DIR>           512  boot
  01/28/01  08:23p                  29  temp.txt
  01/28/01  08:23p              28,739  IsaBus.efi
          4 File(s)      28,828 bytes
          3 Dir(s)
```

2.3.10 date

```
DATE [mm/dd/[yy]yy]


    mm    - Month of date to be set
    dd    - Day of date to be set
    yyyy  - Year of date to be set


Note:
    1. yy: 98=1998, 99=1999, 00=2000, 01=2001, ..., 97=2097.
    2. yyyy: 1998 - 2099, other values are invalid.
    3. EFI may behave unpredictably if illegal date values are used.
```

This command displays to sets the current date for the system. If no parameters are used, it shows the current date. If a valid month, day, and year are provided, then the system's date will be updated. The following example shows typical output from help for this command.

Examples

```
* To display the current date in the system:
  fs0:\> date
  06/18/2001

* To set the date with long year format:
  fs0:\> date 01/01/2050
  fs0:\> date
  01/01/2050

* To set the date with short year format:
  fs0:\> date 06/18/01
  fs0:\> date
  06/18/2001

* The attempt to set the date with an invalid year will result a failure:
  fs0:\> date 06/18/1997
  date: Invalid Year. Year range : 1998 - 2099
```

2.3.11 dblk

```
DBLK device [Lba] [blocks]


    device          - The name of the block device to be displayed
    Lba             - The index(hex) of the first block to be displayed
    blocks          - The number(hex) of blocks to be displayed


Note:
    1. If 'blocks' is larger than 0x10, DBLK displays the first 0x10 blocks.
    2. See dh and map command to find which blocks can be displayed.
    3. If a FAT files system is detected, some FAT parameters will also be
       displayed (label, systemid, oemid, sectorsize, clustersize, media etc)
       after all the blocks have been displayed.
    4. All units are in hex.
```

Displays the contents of one or more blocks from a block device. If Lba is not specified or it is greater than the last block on that block device, then block #0 is displayed. If blocks is not specified, then on1y one block will be displayed. The maximum number of blocks that can be displayed at one time is 0x10. The following example shows typical output for help on this command.

Examples

```
* To display one block of blk0, beginning from 0 block:
  Shell>dblk blk0


* To display one block of fs0, beginning from 0x2 block:
  Shell>dblk fs0 2


* To display 0x5 blocks of fs0, beginning from 0x12 block:
  Shell>dblk fs0 12 5


* To display 0x10 blocks of fs0, beginning from 0x12 block:
  Shell>dblk fs0 12 10


* The attempt to display more than 0x10 blocks will display only 0x10 blocks:
  Shell>dblk fs0 12 20


* Sample: To display one block of blk2, beginning from first block (block 0):
  fs1:\tmps1> dblk blk2 0 1


 LBA 0x0000000000000000 Size 0x00000200 bytes BlkIo 0x3F0CEE78
    00000000: EB 3C 90 4D 53 44 4F 53-35 2E 30 00 02 04 08 00  *.<.MSDOS5.0.....*
    00000010: 02 00 02 00 00 F8 CC 00-3F 00 FF 00 3F 00 00 00  *........?...?...*
    00000020: 8E 2F 03 00 80 01 29 2C-09 1B D0 4E 4F 20 4E 41  *./....),...NO NA*
    00000030: 4D 45 20 20 20 20 46 41-54 31 36 20 20 20 33 C9  *ME    FAT16   3.*
    00000040: 8E D1 BC F0 7B 8E D9 B8-00 20 8E C0 FC BD 00 7C  *......... ......*
```

```
00000050: 38 4E 24 7D 24 8B C1 99-E8 3C 01 72 1C 83 EB 3A   *8N$.$....<.r...:*
00000060: 66 A1 1C 7C 26 66 3B 07-26 8A 57 FC 75 06 80 CA   *f...&f;.&.W.u...*
00000070: 02 88 56 02 80 C3 10 73-EB 33 C9 8A 46 10 98 F7   *..V....s.3..F...*
00000080: 66 16 03 46 1C 13 56 1E-03 46 0E 13 D1 8B 76 11   *f..F..V..F....v.*
00000090: 60 89 46 FC 89 56 FE B8-20 00 F7 E6 8B 5E 0B 03   *`.F..V.. ....^..*
000000A0: C3 48 F7 F3 01 46 FC 11-4E FE 61 BF 00 00 E8 E6   *.H...F..N.a.....*
000000B0: 00 72 39 26 38 2D 74 17-60 B1 0B BE A1 7D F3 A6   *.r9&8-t.`.......*
000000C0: 61 74 32 4E 74 09 83 C7-20 3B FB 72 E6 EB DC A0   *at2Nt... ;.r....*
000000D0: FB 7D B4 7D 8B F0 AC 98-40 74 0C 48 74 13 B4 0E   *........@t.Ht...*
000000E0: BB 07 00 CD 10 EB EF A0-FD 7D EB E6 A0 FC 7D EB   *...............*
000000F0: E1 CD 16 CD 19 26 8B 55-1A 52 B0 01 BB 00 00 E8   *.....&.U.R......*
00000100: 3B 00 72 E8 5B 8A 56 24-BE 0B 7C 8B FC C7 46 F0   *;.r.[.V$......F.*
00000110: 3D 7D C7 46 F4 29 7D 8C-D9 89 4E F2 89 4E F6 C6   *=..F.)....N..N..*
00000120: 06 96 7D CB EA 03 00 00-20 0F B6 C8 66 8B 46 F8   *........ ...f.F.*
00000130: 66 03 46 1C 66 8B D0 66-C1 EA 10 EB 5E 0F B6 C8   *f.F.f..f....^...*
00000140: 4A 4A 8A 46 0D 32 E4 F7-E2 03 46 FC 13 56 FE EB   *JJ.F.2....F..V..*
00000150: 4A 52 50 06 53 6A 01 6A-10 91 8B 46 18 96 92 33   *JRP.Sj.j...F...3*
00000160: D2 F7 F6 91 F7 F6 42 87-CA F7 76 1A 8A F2 8A E8   *......B...v.....*
00000170: C0 CC 02 0A CC B8 01 02-80 7E 02 0E 75 04 B4 42   *...........u..B*
00000180: 8B F4 8A 56 24 CD 13 61-61 72 0B 40 75 01 42 03   *...V$..aar.@u.B.*
00000190: 5E 0B 49 75 06 F8 C3 41-BB 00 00 60 66 6A 00 EB   *^.Iu...A...`fj..*
000001A0: B0 4E 54 4C 44 52 20 20-20 20 20 20 0D 0A 52 65   *.NTLDR      ..Re*
000001B0: 6D 6F 76 65 20 64 69 73-6B 73 20 6F 72 20 6F 74   *move disks or ot*
000001C0: 68 65 72 20 6D 65 64 69-61 2E FF 0D 0A 44 69 73   *her media....Dis*
000001D0: 6B 20 65 72 72 6F 72 FF-0D 0A 50 72 65 73 73 20   *k error...Press *
000001E0: 61 6E 79 20 6B 65 79 20-74 6F 20 72 65 73 74 61   *any key to resta*
000001F0: 72 74 0D 0A 00 00 00 00-00 00 00 AC CB D8 55 AA   *rt............U.*
```

```
Fat 16 BPB  FatLabel: 'NO NAME    '  SystemId: 'FAT16   ' OemId: 'MSDOS5.0'
 SectorSize 0x200  SectorsPerCluster 4 ReservedSectors 8  # Fats 2
 Root Entries 0x200  Media 0xF8  Sectors 0x32F8E  SectorsPerFat 0xCC
 SectorsPerTrack 0x3F Heads 255
```

2.3.12 devices

```
DEVICES [-b] [-lXXX]


    -b        - Displays one screen at a time
    -lXXX     - Displays devices using the ISO 639-2 language specified by XXX


Display Format:
    CTRL          - The handle number of the EFI device
    TYPE          - The device type.
                        [R] Root Controller
                        [B] Bus Controller
                        [D] Device Controller
    CFG           - A managing driver supports the Driver Configuration Protocol
    DIAG          - A managing driver supports the Driver Diagnostics Protocol
    #P            - The number of parent controllers that the device has
    #D            - The number of drivers that are managing the device
    #C            - The number of child controllers this device has produced
    DEVICE NAME   - The name of the device from the Component Name Protocol
```

Displays the list of devices that are being managed by EFI drivers that follow the EFI Driver Model.

Examples

```
 Shell> devices

 C  T   D
 T  Y C I
 R  P F A
 L  E G G #P #D #C Device Name
 == = = = == == == ============================================================
 0C R - -  - 1 11 Acpi(PNP0A03,0)
 15 D - -  1  -  - Acpi(PNP0A03,0)/Pci(0|0)
 16 D - -  1  -  - Acpi(PNP0A03,0)/Pci(1|0)
 17 B - -  1  5  2 Acpi(PNP0A03,0)/Pci(1|0)/Pci(0|0)
 18 D - -  1  -  - Acpi(PNP0A03,0)/Pci(1E|0)
 19 B - -  1  1  1 Acpi(PNP0A03,0)/Pci(1E|0)/Pci(0|0)
 1A D - -  1  -  - Acpi(PNP0A03,0)/Pci(1E|0)/Pci(1|0)
 1B B - -  1  2  6 Acpi(PNP0A03,0)/Pci(1F|0)
 1C B X X  1  1  3 PCI IDE/ATAPI Controller
 1D B - -  1  2  4 Usb Universal Host Controller
 1E D - -  1  -  - Acpi(PNP0A03,0)/Pci(1F|3)
 1F D - -  1  -  - Acpi(PNP0A03,0)/Pci(1F|5)
 20 B - -  1  1  1 Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0501,0)
 21 B - -  1  1  1 Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0501,1)
 22 B - -  1  3  1 PS/2 Keyboard Device
 23 D - -  1  -  - Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,1)
```

```
24 D - -  1  3  - FAT File System [FAT12] 1440 KB
25 D - -  1  -  - Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0604,1)
26 D - -  1  -  - Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0501,0)/Uart(115200 81)
27 D - -  1  -  - Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0501,1)/Uart(115200 81)
35 D - -  1  -  - Acpi(PNP0A03,0)/Pci(1E|0)/Pci(0|0)/Mac(00D0B7E81394)
36 B X X  1  2  2 IBM-DTLA-307030
37 D X X  1  1  - MATSHITADVD-ROM SR-8585
38 D X X  1  1  - Hewlett-Packard CD-Writer Plus 9100b
39 D - -  1  2  - FAT File System [FAT32] 28 GB
3A D - -  1  -  - Acpi(PNP0A03,0)/Pci(1F|2)/Usb(1, 0)
3B D - -  1  -  - Acpi(PNP0A03,0)/Pci(1F|2)/Usb(1, 0)/Usb(0, 0)
3C D - -  1  -  - Acpi(PNP0A03,0)/Pci(1F|2)/Usb(1, 0)/Usb(2, 0)
3D D - -  1  -  - Acpi(PNP0A03,0)/Pci(1F|2)/Usb(1, 0)/Usb(3, 0)
54 D - -  1  -  - Primary Console Output Device
55 D - -  1  -  - Primary Standard Error Device
56 D - -  1  -  - Primary Console Input Device
7B D - -  1  2  - FAT File System [FAT12] 15 MB
```

2.3.13 devtree

```
DEVTREE [-b] [-d] [lXXX] [DeviceHandle]


    -b          - Displays one screen at a time
    -d          - Displays device tree using device paths
    -lXXX       - Displays device tree using the ISO 639-2 language
                  specified by XXX.
    DeviceHandle - Displays device tree below a certain handle
```

Displays the tree of devices that are being managed by drivers that follow the EFI Driver Model.

Examples

```
  * To display the tree of all devices that follow the EFI Driver Model
    Shell> devtree


  * To display the tree of all devices below device 28 that follow the
  * EFI Driver Model
    Shell> devtree 28


  * To display the tree of all devices that follow the EFI Driver Model,
  * break when screen is full:
    Shell> devtree -b


 Device Tree
   Ctrl[03]
   Ctrl[0C] Acpi(PNP0A03,0)
     Ctrl[15] Acpi(PNP0A03,0)/Pci(0|0)
     Ctrl[16] Acpi(PNP0A03,0)/Pci(1|0)
     Ctrl[17] Acpi(PNP0A03,0)/Pci(1|0)/Pci(0|0)
       Ctrl[54] Primary Console Output Device
       Ctrl[55] Primary Standard Error Device
     Ctrl[18] Acpi(PNP0A03,0)/Pci(1E|0)
     Ctrl[19] Acpi(PNP0A03,0)/Pci(1E|0)/Pci(0|0)
       Ctrl[35] Acpi(PNP0A03,0)/Pci(1E|0)/Pci(0|0)/Mac(00D0B7E81394)
     Ctrl[1A] Acpi(PNP0A03,0)/Pci(1E|0)/Pci(1|0)
     Ctrl[1B] Acpi(PNP0A03,0)/Pci(1F|0)
       Ctrl[20] Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0501,0)
         Ctrl[26] Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0501,0)/Uart(115200 81)
       Ctrl[21] Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0501,1)
         Ctrl[27] Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0501,1)/Uart(115200 81)
       Ctrl[22] PS/2 Keyboard Device
         Ctrl[56] Primary Console Input Device
       Ctrl[23] Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0303,1)
       Ctrl[24] FAT File System [FAT12] 1440 KB
       Ctrl[25] Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0604,1)
     Ctrl[1C] PCI IDE/ATAPI Controller
```

```
    Ctrl[36] IBM-DTLA-307030
       Ctrl[39] FAT File System [FAT32] 28 GB
       Ctrl[7B] FAT File System [FAT12] 15 MB
    Ctrl[37] MATSHITADVD-ROM SR-8585
    Ctrl[38] Hewlett-Packard CD-Writer Plus 9100b
  Ctrl[1D] Usb Universal Host Controller
    Ctrl[3A] Acpi(PNP0A03,0)/Pci(1F|2)/Usb(1, 0)
    Ctrl[3B] Acpi(PNP0A03,0)/Pci(1F|2)/Usb(1, 0)/Usb(0, 0)
    Ctrl[3C] Acpi(PNP0A03,0)/Pci(1F|2)/Usb(1, 0)/Usb(2, 0)
    Ctrl[3D] Acpi(PNP0A03,0)/Pci(1F|2)/Usb(1, 0)/Usb(3, 0)
  Ctrl[1E] Acpi(PNP0A03,0)/Pci(1F|3)
  Ctrl[1F] Acpi(PNP0A03,0)/Pci(1F|5)
 Ctrl[3E] Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Master)/HD(Part2,SigF9D8F9D8)
/\Efi\BootStr
 Ctrl[4D] VenHw(D65A6B8C-71E5-4DF0-A909-F0D2992B5AA9)
 Ctrl[50] Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0501,0)/Uart(115200
81)/VenMsg(EBA4E8D2-3858-41EC-A281-2647BA9660D0)
 Ctrl[7C] Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0604,0)/\/devtree.example
```

2.3.14   dh

```
DH [-b] [-d] [-lXXX] [-v] [handle]|[-p prot_id]


    -b           - Displays one screen at a time
    handle       - Dumps information of a certain handle
    -p prot_id   - Dumps all handles of a certain protocol
    -d           - Dumps EFI Driver Model related information
    -lXXX        - Dums information using the ISO 639-2 language
                   specified by XXX.
    -v           - Dumps information on all handles
```

This command displays the device handles in the EFI environment. When the **dh** command is used without any parameters, a list of all the device handles in the EFI environment is displayed. A single device handle can contain one or more protocol instances. If the **dh** command is used with a specific handle number, the details of all the protocols associated with that device handle are displayed. If the **–p** option is used, the list of device handles containing a specific protocol will be displayed. The following examples show how the **dh** command can be used. The example below shows typical output from help for this command.

Examples

```
* To display all handles, display one screen at a time:
  Shell> dh -b
  Handle dump
    1: Image(DXE Core)
    2: FwVol FwFileSys FwVolBlk DevPath(MemMap(11:1B50000-1D4FFC8))
    3:
    4:
    5: Image(WinNtThunk)
    6: WinNtThunk DevPath(..76F3-11D4-BCEA-0080C73C8881))
    7: Image(WinNtBusDriver) DriverBinding
    ...

* To display the detailed information handle 30:
  Shell> dh 30
  Handle 30 (01AF5308)
    IsaIo
      ROM Size......: 00000000
      ROM Location..: 00000000
      ISA Resource List :
        IO  : 000003F8-000003FF  Attr : 00000000
        INT : 00000004-00000000  Attr : 00000000

    dpath
      PNP Device Path for PnP
       HID A0341D0, UID 0
```

```
       Hardware Device Path for PCI
       PNP Device Path for PnP
        HID 50141D0, UID 0
     AsStr: 'Acpi(PNP0A03,0)/Pci(1F|0)/Acpi(PNP0501,0)'



  * To display all handles with 'diskio' protocol:
    Shell> dh -p diskio
    Handle dump by protocol 'Diskio'
     15: diskio blkio DevPath(..i(3|1)/Ata(Secondary,Master))
     16: diskio blkio DevPath(..,1)/PCI(0|0)/Scsi(Pun0,Lun0))
     44: diskio blkio fs DevPath(..ABD0-01C0-507B-9E5F8078F531)) ESP
     45: diskio blkio fs DevPath(..i(Pun0,Lun0)/HD(Part4,SigG0)) ESP
     17: diskio blkio DevPath(..PCI(3|1)/Ata(Primary,Master))


  * To display all handles with 'Image' protocol, break when screen is full:
    Shell> dh -p Image -b
    Handle dump by protocol 'image'
      1: Image(DXE Core)
      5: Image(WinNtThunk)
      7: Image(WinNtBusDriver) DriverBinding
      8: Image(Metronome)
      A: Image(IsaBus) DriverBinding
      B: Image(WinNtConsole) DriverBinding
      ...
```

2.3.15 disconnect

```
disconnect DeviceHandle# [DriverHandle# [ChildHandle#]] | [-r]


    DeviceHandle#   - Device handle (hex)
    DriverHandle#   - Driver handle (hex)
    ChildHandle#    - Child handle of device (hex)
    -r              - Disconnect drivers from all devices
```

This command is used to disconnect one or more drivers from devices. If the **–r** option is used, then all drivers are disconnected from all devices in the system. The following example is typical output from help for this command.

Examples

```
* To disconnect all drivers from all devices:
  Shell> disconnect -r

* To disconnect all drivers from device 28:
  fs0:\> disconnect 28

* To disconnect driver 17 from device 28:
  fs0:\> disconnect 28 17

* To disconnect driver 17 from device 28 and destroy child 32:
  fs0:\> disconnect 28 17 32
```

2.3.16 dmem

dmem [Address] [Size] [;MMIO]

    address - Starting address (hext) to display. This needs to be on an even
            boundry for the processor that this command is run on.
    size   - Number of bytes to display in hex.
    ;MMIO - Memory mapped IO. It will turn on any bits required in the
            chipset to force memory access out to the PCI bus.

Note:
    1. If no address is given the EFI, system table entry point will be
       displayed. Also it will display all other system table pointer entries
       as well (runtime services, boot services etc.).
    2. Address must be on a even boundry address for the processor being used.
    3. All units are in hex.

Displays the contents of system memory or device memory. If Address is not specified, then the contents of the EFI System Table are displayed. Otherwise, memory starting at Address is displayed. Size specifies the number of bytes to display. If Size is not specified, then this command defaults to 512 bytes. If MMIO is not specified, then main system memory is displayed. Otherwise, device memory is displayed through the use of the DEVICE_IO protocol. The following example shows typical output for help on this command.

Examples

```
* To display default content:
  fs0:\> dmem

  Memory Address 000000003FF7D808 200 Bytes
  3FF7D808: 49 42 49 20 53 59 53 54-02 00 01 00 78 00 00 00  *IBI SYST....x...*
  3FF7D818: 5C 3E 6A FE 00 00 00 00-88 2E 1B 3F 00 00 00 00  *\>j........?....*
  3FF7D828: 26 00 0C 00 00 00 00 00-88 D3 1A 3F 00 00 00 00  *&..........?....*
  3FF7D838: A8 CE 1A 3F 00 00 00 00-88 F2 1A 3F 00 00 00 00  *...?.......?....*
  3FF7D848: 28 EE 1A 3F 00 00 00 00-08 DD 1A 3F 00 00 00 00  *(..?.......?....*
  3FF7D858: A8 EB 1A 3F 00 00 00 00-18 C3 3F 3F 00 00 00 00  *...?...........*
  3FF7D868: 00 4B 3F 3F 00 00 00 00-06 00 00 00 00 00 00 00  *.K.............*
  3FF7D878: 08 DA F7 3F 00 00 00 00-70 74 61 6C 88 00 00 00  *...?....ptal....*
  3FF7D888: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *...............*
  3FF7D898: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *...............*
  3FF7D8A8: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *...............*
  3FF7D8B8: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *...............*
  3FF7D8C8: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *...............*
  3FF7D8D8: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *...............*
  3FF7D8E8: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *...............*
  3FF7D8F8: 00 00 00 00 00 00 00 00-70 68 06 30 88 00 00 00  *........ph.0....*
  3FF7D908: 65 76 6E 74 00 00 00 00-02 02 00 60 00 00 00 00  *evnt.......`....*
  3FF7D918: 18 6F 1A 3F 00 00 00 00-10 E0 3F 3F 00 00 00 00  *.o.?..........*
```

```
3FF7D928: 10 00 00 00 00 00 00 00-40 C0 12 3F 00 00 00 00  *........@..?....*
3FF7D938: 10 80 13 3F 00 00 00 00-00 00 00 00 00 00 00 00  *...?............*
3FF7D948: 00 00 00 00 00 00 00 00-40 7D 3F 3F 00 00 00 00  *........@.....*
3FF7D958: 50 6F 1A 3F 00 00 00 00-00 00 00 00 00 00 00 00  *Po.?............*
3FF7D968: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
3FF7D978: 00 00 00 00 00 00 00 00-70 74 61 6C 88 00 00 00  *........ptal....*
3FF7D988: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
3FF7D998: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
3FF7D9A8: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
3FF7D9B8: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
3FF7D9C8: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
3FF7D9D8: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
3FF7D9E8: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
3FF7D9F8: 00 00 00 00 00 00 00 00-70 68 06 30 A0 00 00 00  *........ph.0....*


Valid EFI Header at Address 000000003FF7D808
--------------------------------------------
System: Table Structure size 00000078 revision 00010002
ConIn (0x3F1AD388) ConOut (0x3F1AF288) StdErr (0x3F1ADD08)
Runtime Services 0x000000003F3FC318
Boot Services    0x000000003F3F4B00
SAL System Table 0x000000003FF22760
ACPI Table       0x000000003FFD9FC0
ACPI 2.0 Table   0x00000000000E2000
MPS Table        0x000000003FFD0000
SMBIOS Table     0x00000000000F0020
```

* To display memory contents from 0x1af3088 with size of 16 bytes:
  Shell> dmem 1af3088 16

```
  Memory Address 0000000001718E08 16 Bytes
  01718E08: 49 42 49 20 53 59 53 54-00 00 02 00 18 00 00 00 *IBI SYST........*
  01718E18: FF 9E D7 9B 00 00                               *......*
```

* To display memory mapped IO contents, from 0x1af3088 with size of 16 bytes:
  Shell> dmem 1af3088 16 ;MMIO

2.3.17 dmpstore

dmpstore

Displays all the environment variables being managed by EFI. The following example shows typical output for help on this command.

Examples

```
Shell> dmpstore
 Dump NVRAM

 Variable RT+BS 'Efi:BootCurrent' DataSize = 2
   00000000: FF FF                                       *..*
 Variable NV+RT+BS 'Efi:LangCodes' DataSize = 2A
   00000000: 65 6E 67 65 6E 6D 61 6E-67 63 68 69 7A 68 6F 64 *engenmangchizhod*
   00000010: 65 75 67 65 6D 67 65 72-67 6D 68 67 6F 68 66 72 *eugemgergmhgohfr*
   00000020: 61 66 72 65 66 72 6D 66-72 6F             *afrefrmfro*
 Variable NV+RT+BS 'Efi:Lang' DataSize = 3
   00000000: 65 6E 67                                   *eng*
  ...
 Variable NV+BS 'ShellAlias:del' DataSize = 6
   00000000: 72 00 6D 00 00 00                           *r.m...*
 Variable NV+BS 'ShellAlias:copy' DataSize = 6
   00000000: 63 00 70 00 00 00                           *c.p...*
 Variable NV+BS 'SEnv:path' DataSize = 4
   00000000: 2E 00 00 00                                 *....*
```

2.3.18 drivers

```
DRIVERS [-b] [-lXXX]


    -b      - Displays one screen at a time
    -lXXX   - Displays drivers using the ISO 639-2 language specified by XXX


Display Format:
    DRV           - The handle number of the EFI driver
    TYPE          - The driver type.
                       [B] Bus Driver
                       [D] Device Driver
    CFG           - The driver supports the Driver Configuration Protocol
    DIAG          - The driver supports the Driver Diagnostics Protocol
    #D            - The number of devices that the driver is managing
    #C            - The number of child devices that this driver has produced
    DRIVER NAME   - The name of the driver from the Component Name Protocol
    IMAGE NAME    - The file path from which the driver was loaded
```

Displays the list of drivers that follow the EFI Driver Model.

Examples

```
Shell> drivers
           T  D
D          Y C I
R          P F A
V  VERSION E G G #D #C DRIVER NAME                       IMAGE NAME
== ======== = = = == == ================================ ====================
0D 00000001 B - -  1 11 PCI Bus Driver                   PciBus
0E 00000001 D - -  1  - PC-AT ISA Device Enumeration Driver PcatIsaAcpi
0F 00000001 B - -  1  6 ISA Bus Driver                   IsaBus
10 00000001 B - -  2  2 ISA Serial Driver                IsaSerial
11 00000000 D - -  1  - PCI VGA Mini Port Driver         PciVgaMiniPort
12 00000001 D - -  1  - VGA Class Driver                 VgaClass
13 00000001 ? - -  -  - Cirrus Logic 5430 UGA Driver     CirrusLogic5430
14 00000001 ? - -  -  - UGA Console Driver               GraphicsConsole
28 00000001 B X X  1  3 PCI IDE/ATAPI Bus Driver         IdeBus
29 00000000 ? - -  -  - ATAPI SCSI Pass Thru Driver      AtapiScsiPassThru
2A 00000001 D - -  1  - Usb Uhci Driver                  UsbUhci
2B 00000001 B - -  1  4 USB Bus Driver                   UsbBus
2C 00000001 ? - -  -  - Generic USB Mass Storage Driver  UsbMassStorage
2D 00000001 ? - -  -  - Usb Bot Mass Storage Driver      UsbBot
2E 00000001 ? - -  -  - <UNKNOWN>                        UsbCbi1
2F 00000001 D - -  6  - Generic Disk I/O Driver          DiskIo
30 00000001 B - -  1  2 Partition Driver(MBR/GPT/El Torito) Partition
31 00000001 D - -  3  - FAT File System Driver           Fat
32 00000001 D - -  1  - PS/2 Keyboard Driver             Ps2Keyboard
```

```
33 00000001 ? - -  -  - Usb Keyboard Driver               UsbKeyboard
34 00000001 B - -  1  1 Intel(R) PRO 100 UNDI Driver      Bin\Undi.Z
3F 00000001 ? - -  -  - Serial Terminal Driver            Terminal
40 00000001 D - -  1  - Platform Console Management Driver ConPlatform
41 00000001 D - -  1  - Platform Console Management Driver ConPlatform
42 00000001 B - -  1  1 Console Splitter Driver           ConSplitter
43 00000001 B - -  1  1 Console Splitter Driver           ConSplitter
44 00000001 B - -  1  1 Console Splitter Driver           ConSplitter
45 00000001 ? - -  -  - Simple Network Protocol Driver    Snp3264
46 00000001 ? - -  -  - PXE Base Code Driver              PxeBc
49 00000001 D - -  1  - ISA Floppy Driver                 IsaFloppy
4A 00000001 ? - -  -  - PS/2 Mouse Driver                 Ps2Mouse
4B 00000001 ? - -  -  - Usb Mouse Driver                  UsbMouse
4C 00000001 ? - -  -  - Serial Mouse Driver               SerialMouse
```

2.3.19 drvcfg

```
DRVCFG [-c] [-lXXX] [-f] [-v] [-s] [DriverHandle [DeviceHandle [ChildHandle]]]


    -c           - Configure all child devices
    -lXXX        - Configure using the ISO 639-2 language specified by XXX
    -f           - Force defaults
    -v           - Validate options
    -s           - Set options
    DriverHandle - The handle of the driver to configurre
    DeviceHandle - The handle of a device that DriverHandle is managing
    ChildHandle  - The handle of a device that is a child of DeviceHandle
```

Invokes the Driver Configuration Protocol from the EFI driver specified by **DriverHandle**. Configuration is either invoked on a device that is being managed by **DriverHandle**, or a child device that **DriverHandle** has produced. If **ChildHandle** is not specified, then the Driver Configuration Protocol is invoked for **DeviceHandle**. If **ChildHandle** is specified, then the Driver Configuration Protocol is invoked for **ChildHandle**. The options for a specific device can be validated with the **–v**, and default options for a device can be set with **–f**. If the **–s** switch is used, then the Driver Configuration Protocol for the EFI driver specified by **DriverHandle** will use the standard console devices to interact with the user.

Examples

```
    * To display the list of devices that are available for configuration
      Shell> drvcfg

    * To display the list of devices and child devices that are available
    * for configuration
      Shell> drvcfg -c

    * To force defaults on all devices
      Shell> drvcfg -f

    * To force defaults on all devices that are managed by driver 17
      Shell> drvcfg -f 17

    * To force defaults on device 28 that is managed by driver 17
      Shell> drvcfg -f 17 28

    * To force defaults on all child devices of device 28 that is managed by
    * driver 17
      Shell> drvcfg -f 17 28 -c

    * To force defaults on child device 30 of device 28 that is managed by
    * driver 17
      Shell> drvcfg -f 17 28 30
```

* To validate options on all devices
  Shell> drvcfg -v

* To validate options on all devices that are managed by driver 17
  Shell> drvcfg -v 17

* To validate options on device 28 that is managed by driver 17
  Shell> drvcfg -v 17 28

* To validate options on all child devices of device 28 that is managed by
* driver 17
  Shell> drvcfg -v 17 28 -c

* To validate options on child device 30 of device 28 that is managed by
* driver 17
  Shell> drvcfg -v 17 28 30

* To set options on device 28 that is managed by driver 17
  Shell> drvcfg -s 17 28

* To set options on child device 30 of device 28 that is managed by driver 17
  Shell> drvcfg -s 17 28 30

* To set options on device 28 that is managed by driver 17 in english
  Shell> drvcfg -s 17 28 -leng

* To set options on device 28 that is managed by driver 17 in spanish
  Shell> drvcfg -s 17 28 -lspa

2.3.20 drvdiag

```
DRVDIAG [-c] [-lXXX] [-s] [-e] [-m] [DriverHandle [DeviceHandle [ChildHandle]]]


    -c           - Diagnose all child devices
    -lXXX        - Diagnose using the ISO 639-2 language specified by XXX
    -s           - Run diagnostics in standard mode
    -e           - Run diagnostics in extended mode
    -m           - Run diagnostocs in manufacturing mode
    DriverHandle - The handle of the driver to configurre
    DeviceHandle - The handle of a device that DriverHandle is managing
    ChildHandle  - The handle of a device that is a child of DeviceHandle
```

Invokes the Driver Diagnostics Protocol from the EFI driver specified by **DriverHandle**. Diagnostics are either invoked on a device that is being managed by **DriverHandle**, or a child device that **DriverHandle** has produced. If **ChildHandle** is not specified, then the Driver Diagnostics Protocol is invoked for **DeviceHandle**. If **ChildHandle** is specified, then the Driver Diagnostics Protocol is invoked for **ChildHandle**.

Examples

```
* To display the list of devices that are available for diagnostics
  Shell> drvdiag

* To display the list of devices and child devices that are available for
* diagnostics
  Shell> drvdiag -c

* Run diagnostics in standard mode on all devices
  Shell> drvdiag -s

* Run diagnostics in standard mode on all devices in english
  Shell> drvdiag -s -leng

* Run diagnostics in standard mode on all devices in spanish
  Shell> drvdiag -s -lspa

* Run diagnostics in standard mode on all devices and child devices
  Shell> drvdiag -s -c

* Run diagnostics in extended mode on all devices
  Shell> drvdiag -e

* Run diagnostics in manufacturing mode on all devices
  Shell> drvdiag -m

* Run diagnostics in standard mode on all devices managed by driver 17
  Shell> drvdiag -s 17
```

```
* Run diagnostics in standard mode on device 28 managed by driver 17
  Shell> drvdiag -s 17 28

* Run diagnostics in standard mode on all child devices of device 28
* managed by driver 17
  Shell> drvdiag -s 17 28 -c

* Run diagnostics in standard mode on child device 30 of device 28
* managed by driver 17
  Shell> drvdiag -s 17 28 30
```

2.3.21 echo

```
ECHO [-on|-off]
ECHO [message]


    -on         - Displays when reading command lines from batch files
    -off        - Doesn't display when reading batch command lines
    message     - Displays a message string


Note:
    1. Echo -off means not to display command line when reading from batch
       files. It's not like MS-DOS echo.
    2. Echo without a parameter is showing current echo setting.
```

The first form of the **echo** command controls whether or not batch commands are displayed as they are read from the batch file. If no argument is given the current "on" or "off" status is displayed. The second form prints the given *message* to the display. Wildcard substitution is performed on the second form of the command. The following example shows typical output for help on this command.

Examples

```
* To display a message string of 'Hello World':
  fs0:\> echo Hello World
  Hello World

* To turns command echoing off:
  fs0:\> echo -off

* To displays current echo setting:
  fs0:\> echo
  Echo is off

* To execute a batch file named as HelloWorld.nsh:
  fs0:\> HelloWorld.nsh
  Hello World

* To turns command echoing on,
  fs0:\> echo -on

* To execute HelloWorld.nsh, display when reading lines from batch file:
  fs0:\> HelloWorld.nsh
  +HelloWorld.nsh> echo Hello World
  Hello World
```

2.3.22 edit

```
EDIT [file]

    file    - Name of file to be edited

 Note:
    1. If the file is not specified, NewFile.txt is edited.
    2. The size of file shall not be larger than 16 Mbytes.
```

This command allows a file to be edited using a full screen editor. The editor supports both Unicode and ASCII file types. The following example shows typical output for help on this command.

Examples

```
fs0:\> edit shell.log
```

2.3.23 EfiCompress

```
EfiCompress [InFile] [OutFile]


     InFile        - Name of file to compress
     OutFile       - Output file to write compressed data to
```

This command is used to compress a file and write the compressed form out to a new file. The example below shows typical output from help for this command

Examples

```
Shell> eficompress
```

2.3.24 EfiDecompress

```
EfiDecompress [InFile] [OutFile]


    InFile        - Name of file to decompress
    OutFile       - Output file to write uncompressed data to
```

This command is used to decompress a file and write the uncompressed form out to a new file. The example below shows typical output from help for this command

Example

```
Shell> efidecompress
```

2.3.25 err

```
ERR [Error Level]


   Error Level   - New error level bit mask.

Note:
   1. In debug version, the message whose error level is higher than this
      level will be displayed.
   2. Saving to NVRAM will cause the error level to be saved and used on all
      future reboots. Core of EFI will use the new error level as system is
      booting (all core EFI routines will then output using the new error
      level).
   3. Error console must be set to a device path (ie com port or console).
      This is typically done in the boot manager, boot option maintence menu,
      Active Standard Error Devices menu. Not all EFI implementations include
      an error console or support debug output. Consult the BIOS release
      notes for this support.
   4. To add your own errors/error level see debug macro in sample
      implementation source under inc/efidebug.h.
```

This command sets the current debug error level in the system. This commands only works if the EFI Shell and the ERR command are built into the core firmware. Error Level is a bit mask for different types of error messages. The following is the list of definitions for the bit mask.

```
#define EFI_D_INIT      0x00000001      // Initialization style messages
#define EFI_D_WARN      0x00000002      // Warnings
#define EFI_D_LOAD      0x00000004      // Load events
#define EFI_D_FS        0x00000008      // EFI File system
#define EFI_D_POOL      0x00000010      // Alloc & Free's
#define EFI_D_PAGE      0x00000020      // Alloc & Free's
#define EFI_D_INFO      0x00000040      // Verbose
#define EFI_D_VARIABLE  0x00000100      // Variable
#define EFI_D_BM        0x00000400      // Boot manager
#define EFI_D_BLKIO     0x00001000      // BlkIo Driver
#define EFI_D_NET       0x00004000      // Network
#define EFI_D_UNDI      0x00010000      // UNDI
#define EFI_D_LOADFILE  0x00020000      // Load File
#define EFI_D_EVENT     0x00080000      // Events
#define EFI_D_ERROR     0x80000000      // Error
```

The following example is typical output for help on this command.

Examples

```
* To display the current error message output level:
  Shell> err
  EFI ERROR 80000000
     00000001  EFI_D_INIT
     00000002  EFI_D_WARN
     00000004  EFI_D_LOAD
     00000008  EFI_D_FS
     00000010  EFI_D_POOL
     00000020  EFI_D_PAGE
     00000040  EFI_D_INFO
     00000100  EFI_D_VARIABLE
     00000400  EFI_D_BM
     00001000  EFI_D_BLKIO
     00004000  EFI_D_NET
     00010000  EFI_D_UNDI
     00020000  EFI_D_LOADFILE
     00080000  EFI_D_EVENT
     80000000  EFI_D_ERROR

* To change the error message output level:
  Shell> err 80000307
     Do you want to make this change permanent and save to NVRAM? [Y/N]n

  EFI ERROR 0000000080000307
     00000001  EFI_D_INIT
     00000002  EFI_D_WARN
     00000004  EFI_D_LOAD
     00000008  EFI_D_FS
     00000010  EFI_D_POOL
     00000020  EFI_D_PAGE
     00000040  EFI_D_INFO
     00000100  EFI_D_VARIABLE
     00000400  EFI_D_BM
     00001000  EFI_D_BLKIO
     00004000  EFI_D_NET
     00010000  EFI_D_UNDI
     00020000  EFI_D_LOADFILE
     00080000  EFI_D_EVENT
     80000000  EFI_D_ERROR
```

2.3.26 exit

exit

This command exits the EFI Shell environment and returns control to the EFI application that launched the EFI Shell. The following example is typical output from help for this command.

Examples

```
Shell> exit
```

2.3.27 for/endfor

```
FOR %indexvar IN set
    command [arguments]
    [command [arguments]]
      ...
ENDFOR

    %indexvar            - The variable to index a set
    set                  - The set to be searched
    command [arguments]  - The command to be executed with optional arguments


Note:
    1. Be available only in batch script files.
    2. FOR shall be matched with ENDFOR.
```

The **for** command executes one or more *commands* for each item in a *set* of items. The *set* may be text strings or filenames or a mixture of both, separated by spaces. Wildcards in filenames are expanded before *command* is executed. *Indexvar* is any single printable character, but it should not be a digit (0-9) because %digit will be interpreted as a positional argument (see "Execution of Batch Scripts"). The namespace for index variables is separate from that for environment variables, so if *indexvar* has the same name as an existing environment variable, the environment variable will remain unchanged by the **for** loop. Each c*ommand* is executed once for each item in the *set*, with any occurrence of *%indexvar* in the command replaced with the current item.

The **for** command is available only in batch scripts. The following example shows typical output for help on this command.

Examples

```
#
#  Sample for loop type contents of all *.txt files
#
for %a in *.txt
   type %a
   echo ===== %a done =====
endfor
```

2.3.28   getmtc

```
getmtc
```

```
Note:
    1. Every time GETMTC is executed, the lower 32 bits will be incremented.
    2. Every time the system is reset, the upper 32 bits will be incremented
       and the lower 32 bits will be reset to 0.
```

This command displays the current monotonic counter value. The lower 32 bits increment every time this command is executed. Every time the system is reset, the upper 32 bits will be incremented, and the lower 32 bits will be reset to 0. The following example is typical output from help for this command.

Examples

```
fs0:\> getmtc
Monotonic count = 100000000

fs0:\> getmtc
Monotonic count = 100000001
```

2.3.29 goto

```
goto label

    label      - Specifies a location in batch file

 Note:
    1. Only available in batch script files.
    2. Execution of batch file will jump to the next line of the label.
    3. GOTO can't jump inside a FOR cycle block.
```

The **goto** command directs batch file execution to the line in the batch file after the given *label*. The command is not supported from the interactive shell. A *label* is a line beginning with a colon (:). The search for *label* is done forward in the batch file. If the end of the file is reached, the search resumes at the top of the file and continues until *label* is found or the starting point is reached. If *label* is not found, the batch process terminates and an error message is displayed. When not searching for the target of a goto command, the shell reads *label* lines and ignores them.

The **goto** command is available only in batch scripts.

The following example shows typical output for help on this command.

Examples

```
#
#  Example script for "goto" command
#
goto Done
...
:Done
cleanup.nsh
```

2.3.30 guid

```
guid [-b]
      [-b]     - Display one screen at a time

Note:
      1. Only displays the guids that were included in the core EFI build at
         the time the core was built. Additional guids may have been added by
         the BIOS integrator. Any GUIDs that are not in the original core build
         or were added by a new protocol that was loaded by the user will show
         up as Unknown Device.
      2. The guid with a '*' at end means that there will probably be dump
         information or token avalible for the protocol to this GUID. We can use
         'dh' command to dump out those info.
```

This command displays a list of all the GUIDs that have been registered with the EFI environment. The following example shows the output from help for this command.

Examples

```
fs0:\> guid -b
  DevIo          : AF6AC311-84C3-11D2-8E3C-00A0C969723B
  diskio         : CE345171-BA0B-11D2-8E4F-00A0C969723B
  blkio          : 964E5B21-6459-11D2-8E39-00A0C969723B  *
  txtin          : 387477C1-69C7-11D2-8E39-00A0C969723B
  txtout         : 387477C2-69C7-11D2-8E39-00A0C969723B  *
  fs             : 964E5B22-6459-11D2-8E39-00A0C969723B
  load           : 56EC3091-954C-11D2-8E3F-00A0C969723B
  image          : 5B1B31A1-9562-11D2-8E3F-00A0C969723B  *
  ...
```

2.3.31 help

```
HELP [-b]|[cmd]


    -b     - Displays one screen at a time
    cmd    - Shell command

Note:
    1. 'cmd -?' also displays the verbose help of cmd, the same as 'help cmd'.
    2. If cmd has no verbose help, its line help will be displayed instead.
    3. HELP will only show commands that were documented in the shell.
```

The help command displays the list of commands that are built into the EFI Shell. The following example shows the typical output from help for this command.

Examples

```
* To display the list of commands of EFI shell:
  Shell> help -b
  help       - Displays commands list or verbose help of a command
  guid       - Displays all the GUIDs in the EFI environment
  set        - Displays, creates, changes or deletes EFI environment variables
  alias      - Displays, creates, or deletes aliases in the EFI shell
  dh         - Displays the handles in the EFI environment
  mount      - Mounts a file system on a block device
  cd         - Displays or changes the current directory
  cls        - Clears the standard output with an optional background color
  cp         - Copies one or more files/directories to another location
  ...

* To display help information of a shell command - ls:
  Shell> help ls
  Shell> ? ls
  Shell> ls -?
```

2.3.32 hexedit

```
hexedit [[-f]FileName|[-d DiskName Offset Size]|[-m Offset Size]]


    -f     - Open file to edit
    -d     - Open disk block to edit
             DiskName - Editing disk's name (for example fs0)
             Offset   - Starting block's No. (beginning from 0)
             Size     - Number of blocks that to be edited
    -m     - Open memory region to edit
             Offset   - Starting offset of memory region (beginning from 0)
             Size     - Size of memory region that to be edited
```

This command allows a file, block device, or memory region to be edited. The region being edited is displayed as hexadecimal bytes, and the contents can be modified and saved. The following example shows typical output for help on this command.

Examples

```
* To edit a file as hex mode:
  fs0:\> hexedit test.bin

* To edit disk block of fs0(floppy here)with 2 blocks:
  fs0:\> hexedit -d fs0 0 2

* To edit memory of fs0 with 2 blocks:
  fs0:\> hexedit -m 0 2
```

2.3.33 if/endif

```
IF [NOT] EXIST file THEN
    command [arguments]
[ELSE
    command [arguments]]
ENDIF


IF [NOT] string1 == string2 THEN
    command [arguments]
    [command [arguments]]
    ...
[ELSE
    command [arguments]
    [command [arguments]]
    ...]
ENDIF


    EXIST file          - TRUE if file exists in the directory
    string1 == string2  - TRUE if the two stings are same


Note:
    1. Only available in batch script files.
    2. If condition is TRUE, commands will execute.
    3. If condition is FALSE but keyword 'NOT' is prefixed, commands will
       also execute.
```

The **if** command executes one or more *commands* if the specified condition is true, unless the **not** keyword is given, in which case the command is executed if the condition is false. The **exist** condition is true if *filename* exists. The *filename* argument may include device and path information. Wildcard expansion is supported for the **exist** form of the command. If more than one file matches the wildcard pattern, the condition evaluates to TRUE. The *string1 == string2* condition is true if the two strings are identical.

The **if** command is available only in batch scripts. The following example shows typical output for help on this command

Examples

```
#
#  Example script for "if" command
#
if exist fs0:\myscript.sc then
myscript myarg1 myarg2
endif
if %myvar% == runboth then
myscript1
myscript2
```

```
endif
```

2.3.34 load

```
LOAD [-nc] file [file...]


    -nc     - Load the driver, but do not connect the driver.
    file    - File that contains the image of the driver, extension as '.efi'


Note:
    1. LOAD can deal with multiple files and 'file' supports wildcard.
    2. Use command Unload to unload a driver.
    3. If option -nc is not specified, then the loaded drivers will be
       automatically connected.  If -nc is specified, then none of the
       loaded drivers will be connected.
```

This command loads an EFI driver. The following example shows typical output from help for this command.

Examples

```
fs0:\> load Isabus.efi
load: Image 'fs0:\Isabus.efi' loaded at 18FE000. returned Success


fs0:\> load Isabus.efi IsaSerial.efi
load: Image 'fs0:\Isabus.efi' loaded at 18E5000. returned Success
load: Image 'fs0:\IsaSerial.efi' loaded at 18DC000. returned Success


fs0:\> load Isa*.efi
load: Image 'fs0:\IsaBus.efi' loaded at 18D4000. returned Success
load: Image 'fs0:\IsaSerial.efi' loaded at 18CB000. returned Success
```

2.3.35 LoadBmp

```
LOADBMP [-c] [-t] [-i[UGA Instance]] [-w[seconds]] file


    -c    - Center the bitmap in the middle of the screen
    -t    - Tile the bitmap file over the entire screen
    -i    - Specifies the UGA screen to display the bitmap on
    -w    - Specifies how long each bitmap file is displayed.
             If [seconds] is not specified or 0, then the bitmap
             Will be displayed until a key is pressed.
```

Loads and displays a BMP file on a UGA display device.

Examples

```
Shell> loadbmp logo.bmp

Shell> loadbmp logo.bmp -c

Shell> loadbmp logo.bmp -t

Shell> loadbmp logo.bmp –i1

Shell> loadbmp logo.bmp –w

Shell> loadbmp logo.bmp –w5
```

2.3.36 LoadPciRom

```
LoadPciRom [RomFile]
```

```
    RomFile    - File that contains a PCI option ROM
```

This command is used to load PCI option ROM images into memory for execution. The file can contain legacy images and multiple PE32 images, in which case all PE32 images will be loaded. The example below shows typical output from help for this command

Examples

```
Shell> loadpcirom
```

2.3.37 ls

```
LS [-b] [-r] [-a[attrib]] [file]


    -b          - Displays one screen at a time
    -r          - Displays recursively (including subdirectories)
    -a          - Displays files of the attributes specified by [attrib]
    attrib      - 'a', 's', 'h', 'r', 'd' or combination of them or NULL
                    a    - Archive
                    s    - System
                    h    - Hidden
                    r    - Read-only
                    d    - Directory
    file        - Name of file/directory (wildcards are permitted)
```

This command lists all the files and subdirectories present in a directory. If the **ls** command is used without any parameters, then the contents of the current working directory are displayed. If a parameter is used, then that parameter is interpreted as a file path, and the contents of the directory specified by the file path are displayed. The following example shows typical output for help on this command.

Examples

```
* To Hide files by adding hidden or system attribute to them:
  fs0:\> attrib +sh *.efi
   ASH  fs0:\IsaBus.efi
   ASH  fs0:\IsaSerial.efi


* To display all, except the files/directories with 'h' or 's' attribute:
  fs0:\> ls
  Directory of: fs0:\

    06/18/01  09:32p                  153  for.nsh
    06/18/01  01:02p <DIR>            512  efi
    06/18/01  01:02p <DIR>            512  test1
    06/18/01  01:02p <DIR>            512  test2
    06/18/01  08:04p                   29  temp.txt
    06/18/01  08:05p <DIR>            512  test
    01/28/01  08:24p        r          29  readme.txt
              3 File(s)        211 bytes
              4 Dir(s)


* To display files with all attributes in the current directory:
  fs0:\> ls -a
  Directory of: fs0:\

    06/18/01  09:32p                  153  for.nsh
```

```
     06/18/01  01:02p <DIR>            512  efi
     06/18/01  01:02p <DIR>            512  test1
     06/18/01  01:02p <DIR>            512  test2
     06/18/01  10:59p              28,739  IsaBus.efi
     06/18/01  10:59p              32,838  IsaSerial.efi
     06/18/01  08:04p                  29  temp.txt
     06/18/01  08:05p <DIR>            512  test
     01/28/01  08:24p        r          29  readme.txt
             5 File(s)      61,788 bytes
             4 Dir(s)
```

* To display files with read-only attributes in the current directory:
  fs0:\> ls -ar
  Directory of: fs0:\

```
     06/18/01  11:14p        r          29  readme.txt
             1 File(s)         29 bytes
             0 Dir(s)
```

* To display the files with attribute of 's':
  fs0:\> ls -as isabus.efi
  Directory of: fs0:\

```
     06/18/01  10:59p              28,739  IsaBus.efi
             1 File(s)      28,739 bytes
             0 Dir(s)
```

* To display all in fs0:\efi directory recursively:
  fs0:\> ls -r -a efi

* To search files with specified type in current directory recursively:
  fs0:\> ls -r -a *.efi -b

2.3.38   map

```
MAP [-r|-v|-d] [sname] [handle] [-b]


    -r      - Resets to default mappings
    -v      - Lists verbose information of mappings
    -d      - Deletes a mapping
    sname   - Defines a name for the mapping by users
    handle  - The number of handle, which is same as dumped from 'dh' command
    -b      - Displays one screen at a time
```

```
Note:
    1. Default mappings are the mappings chosen by the EFI integrator of the
       system.Typically stored in flash NVRAM in the system but may also
       reside on the EFI system partition in the /EFI/Boot directory.
       Core EFI implementation will determine which NVRAM source to use.
    2. The mapping order of FSx: to BlockIO devices is arbitrary. EFI
       applications should not rely on the system mapping blockio devices to a
       particular FSx mapping. Adding or removing media may arbitrarily rename
       the FSx mapping a map -r occurs. Applications should create their own
       mappings.
```

This command is used to define a mapping between a user defined name and a device handle. The most common use of this command is to assign drive letters to device handles that support a file system protocol. Once these mappings are created, the drive letters can be used with all the file manipulation commands. The EFI shell environment creates default mappings for all the device handles that support a recognized file systems. The floppy drive is typically **fs0**, and hard drive partitions with recognized file systems are typically **fs1**, **fs2**, …, **fsn**. This command can be used to create additional mappings, or it can be used to delete an existing mapping with the **-d** option. If the map command is used without any parameters, all the current mappings will be listed. If the **-v** option is used, the mappings will be shown with additional information on each mapped handle. The **-r** option is used to regenerate all the mappings in a system. This is useful if the system configuration has changed since the last boot. The following examples show typical output from help for this command.

Examples

```
* To reset the mapping table as default mappings:
shell> map -r
Device mapping table
  fs0 : VenHw(58C518B1-76F3-11D4-BCEA-0080C73C8881)/VenHw(0C95A92F-A006-11D4-BC
FA-0080C73C8881)
  blk0: VenHw(58C518B1-76F3-11D4-BCEA-0080C73C8881)/VenHw(0C95A92F-A006-11D4-BC
FA-0080C73C8881)

* To display all mappings in the device mapping table:
Shell> map
```

```
Device mapping table
  fs0 : VenHw(58C518B1-76F3-11D4-BCEA-0080C73C8881)/VenHw(0C95A92F-A006-11D4-BC
FA-0080C73C8881)
  blk0: VenHw(58C518B1-76F3-11D4-BCEA-0080C73C8881)/VenHw(0C95A92F-A006-11D4-BC
FA-0080C73C8881)


* To display mapping table verbosely:
Shell> map -v
Device mapping table
  fs0 : VenHw(58C518B1-76F3-11D4-BCEA-0080C73C8881)/VenHw(0C95A92F-A006-11D
4-BCFA-0080C73C8881)
        = Handle 3C: diskio blkio fs WinNtDriverIo
        > \

  blk0: VenHw(58C518B1-76F3-11D4-BCEA-0080C73C8881)/VenHw(0C95A92F-A006-11D
4-BCFA-0080C73C8881)
        = Handle 3C: diskio blkio fs WinNtDriverIo
        > \


* 3C is a valid handle from the above messages, so 3C can be mapped:
Shell> map floppy 3C
Shell> map
Device mapping table
  fs0   : VenHw(58C518B1-76F3-11D4-BCEA-0080C73C8881)/VenHw(0C95A92F-A006-11D4-
BCFA-0080C73C8881)
  blk0  : VenHw(58C518B1-76F3-11D4-BCEA-0080C73C8881)/VenHw(0C95A92F-A006-11D4-
BCFA-0080C73C8881)
  floppy: VenHw(58C518B1-76F3-11D4-BCEA-0080C73C8881)/VenHw(0C95A92F-A006-11D4-
BCFA-0080C73C8881)


* To display the information of mapped name:
Shell> map floppy
  floppy : VenHw(58C518B1-76F3-11D4-BCEA-0080C73C8881)/VenHw(0C95A92F-A006-1
1D4-BCFA-0080C73C8881)


* To operate with the mapped name:
Shell> floppy:
floppy:\> ls

* To delete a mapped name:
Shell> map -d floppy
Shell> map
Device mapping table
  fs0 : VenHw(58C518B1-76F3-11D4-BCEA-0080C73C8881)/VenHw(0C95A92F-A006-11D4-BC
FA-0080C73C8881)
  blk0: VenHw(58C518B1-76F3-11D4-BCEA-0080C73C8881)/VenHw(0C95A92F-A006-11D4-BC
FA-0080C73C8881)
```

```
* To rename a default mapping to a user defined map name, following steps:
* Map display of a system with EDD 3.0 implemented
fs1:\> map
Device mapping table
  fs0 : Acpi(PNP0A03,0)/PCI(3|1)/Ata(Secondary,Master)
  fs1 : Acpi(PNP0A03,1)/PCI(0|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig1B16CC00-ABD0-01C)
  fs2 : Acpi(PNP0A03,1)/PCI(0|0)/Scsi(Pun0,Lun0)/HD(Part4,SigG0)
  blk0: Acpi(PNP0A03,0)/PCI(3|1)/Ata(Secondary,Master)
  blk1: Acpi(PNP0A03,1)/PCI(0|0)/Scsi(Pun0,Lun0)
  blk2: Acpi(PNP0A03,1)/PCI(0|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig1B16CC00-ABD0-01C)
  blk3: Acpi(PNP0A03,1)/PCI(0|0)/Scsi(Pun0,Lun0)/HD(Part4,SigG0)
  blk4: Acpi(PNP0A03,0)/PCI(3|1)/Ata(Primary,Master)
fs1:\>
* To remap fs0: from a LS120 so that it is always called floppy:
fs1:\> map -d fs0
fs1:\> dh -p diskio
Handle dump by protocol 'diskio'
 15: diskio blkio fs DevPath(..i(3|1)/Ata(Secondary,Master))
 16: diskio blkio DevPath(..,1)/PCI(0|0)/Scsi(Pun0,Lun0))
 48: diskio blkio fs DevPath(..ABD0-01C0-507B-9E5F8078F531)) ESP
 49: diskio blkio fs DevPath(..i(Pun0,Lun0)/HD(Part4,SigG0)) ESP
 17: diskio blkio DevPath(..PCI(3|1)/Ata(Primary,Master))
fs1:\> map floppy 15
fs1:\> floppy:
floppy:\>map
Device mapping table
  fs1    : Acpi(PNP0A03,1)/PCI(0|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig1B16CC00-ABD0-0)
  fs2    : Acpi(PNP0A03,1)/PCI(0|0)/Scsi(Pun0,Lun0)/HD(Part4,SigG0)
  blk0   : Acpi(PNP0A03,0)/PCI(3|1)/Ata(Secondary,Master)
  blk1   : Acpi(PNP0A03,1)/PCI(0|0)/Scsi(Pun0,Lun0)
  blk2   : Acpi(PNP0A03,1)/PCI(0|0)/Scsi(Pun0,Lun0)/HD(Part1,Sig1B16CC00-ABD0-0)
  blk3   : Acpi(PNP0A03,1)/PCI(0|0)/Scsi(Pun0,Lun0)/HD(Part4,SigG0)
  blk4   : Acpi(PNP0A03,0)/PCI(3|1)/Ata(Primary,Master)
  floppy: Acpi(PNP0A03,0)/PCI(3|1)/Ata(Secondary,Master)
* Now the current directory is the root on floppy: which is the LS120 on the
  atapi secondary channel master device.
```

2.3.39 memmap

```
memmap [-b]

    -b      - Displays one screen at a time

Note:
    1. The EFI environment keeps track all the physical memory in the system
       and how it is currently being used.
    2. Check the EFI specification to lookup the memory Type.
    3. Use the mem command output to determine where the EFI system table is
       at and where the pointers are for boot services, runtime services,
       ACPI table Smbios table etc.
```

This command displays the memory map that is maintained by the EFI environment. The EFI environment keeps track all the physical memory in the system and how it is currently being used. The EFI Specification defines a set of Memory Type Descriptors. Please see the EFI Specification for a description of how each of these memory types is used. The following example shows typical output for help on this command.

Examples

```
fs0:\> memmap

Type       Start         End            # Pages       Attributes
available  0000000000750000-0000000001841FFF  00000000000010F2 0000000000000009
LoaderCode 0000000001842000-00000000018A3FFF  0000000000000062 0000000000000009
available  00000000018A4000-00000000018C1FFF  000000000000001E 0000000000000009
LoaderData 00000000018C2000-00000000018CAFFF  0000000000000009 0000000000000009
BS_code    00000000018CB000-0000000001905FFF  000000000000003B 0000000000000009
BS_data    0000000001906000-00000000019C9FFF  00000000000000C4 0000000000000009
...
RT_data    0000000001B2B000-0000000001B2BFFF  0000000000000001 8000000000000009
BS_data    0000000001B2C000-0000000001B4FFFF  0000000000000024 0000000000000009
reserved   0000000001B50000-0000000001D4FFFF  0000000000000200 0000000000000009

   reserved  :     512 Pages (2,097,152)
   LoaderCode:      98 Pages (401,408)
   LoaderData:      32 Pages (131,072)
   BS_code   :     335 Pages (1,372,160)
   BS_data   :     267 Pages (1,093,632)
   RT_data   :      19 Pages (77,824)
   available :   4,369 Pages (17,895,424)
Total Memory: 20 MB (20,971,520) Bytes
```

2.3.40 mkdir

```
MKDIR dir [dir...]


    dir     - Name of a directory to be created


Note:
    The parent directory shall already exist.
```

This command creates a new directory on a file system. The following example shows typical output for help on this command.

Examples

```
* To create a new directory:
  fs0:\> mkdir rafter

  fs0:\> ls
  Directory of: fs0:\

    06/18/01  08:05p <DIR>           512  test
    06/18/01  11:14p        r         29  readme.txt
    06/18/01  11:50p <DIR>           512  rafter
            1 File(s)       211 bytes
            2 Dir(s)


* To create multiple directories:
  fs0:\> mkdir temp1 temp2
  fs0:\> ls
  Directory of: fs0:\

    06/18/01  08:05p <DIR>           512  test
    06/18/01  11:14p        r         29  readme.txt
    06/18/01  11:50p <DIR>           512  rafter
    06/18/01  11:52p <DIR>           512  temp1
    06/18/01  11:52p <DIR>           512  temp2
            1 File(s)       211 bytes
            4 Dir(s)
```

2.3.41 mm

```
MM Address [Width 1|2|4|8] [;MMIO | ;MEM | ;IO | ;PCI] [:Value] [-n]


     Address  - Start address of MMIO or MEM or IO or PCI
     ;MEM     - Memory Address (range: 0 - 0xffffffff_ffffffff)
     ;MMIO    - Memory Mapped IO Address (range: 0 - 0xffffffff_ffffffff)
     ;IO      - IO Address (range: 0 - 0xffff)
     ;PCI     - PCI Config Address (format: 0x000000ssbbddffrr)

               ss    - SEG
               bb    - bus
               dd    - Device
               ff    - Func
               rr    - Register
     Width    - Size accessed in bytes
               1    - 1 byte
               2    - 2 bytes
               4    - 4 bytes
               8    - 8 bytes
     Value    - The value to write
     -n       - Non-interactive mode


  Note:
     1. MEM type is default.
     2. In Interactive mode, type a hex value to modify, 'q' or '.' to exit.
     3. When MM PCI configuration space, the 'Address' should follow the format
        of PCI Config Address - 0x000000ssbbddffrr.
     4. Use PCI command to get the ;PCI address needed for a given PCI device.
        This will be displayed in the upper right hand row from the PCI command
        listed after "EFI" 0x000000ssbbddffxx]
     5. '-n' non-interactive mode for use inside of .nsh shell files so that
        mm command can be called from the .nsh file without user intervention.
     6. Not all PCI register locations are writeable. PCI option will also not
        do read-modify write. Will only write the value posted.
```

This command allows the user to display and/or modify I/O register, memory contents, or PCI configuration space. The user can specify the start address and the access size they wish to perform using the Address and Width parameters. MEM accesses system memory, MMIO accesses device memory, IO accesses device I/O ports, and PCI accesses PCI Configuration Space. When this command is executed, the current contents of Address are displayed, and the user has the option of modifying the contents by typing in a hex value. When the user presses [ENTER], the next address is displayed. This is continued until the user enters 'q'. The following example is typical output from help for this command.

Examples

```
* To display or modify memory from 0x1b07288, width=1 byte, Interactive mode:
  fs0:\> mm 1b07288
  MEM  0x0000000001B07288 : 0x6D >
  MEM  0x0000000001B07289 : 0x6D >
  MEM  0x0000000001B0728A : 0x61 > 80
  MEM  0x0000000001B0728B : 0x70 > q


  fs0:\> mm 1b07288
  MEM  0x0000000001B07288 : 0x6D >
  MEM  0x0000000001B07289 : 0x6D >
  MEM  0x0000000001B0728A : 0x80 >           *Modified
  MEM  0x0000000001B0728B : 0x70 > q


* Modifies memory from 0x1b07288, width = 2 bytes, Interactive mode:
  Shell> mm 1b07288 2
  MEM  0x0000000001B07288 : 0x6D6D >
  MEM  0x0000000001B0728A : 0x7061 > 55aa
  MEM  0x0000000001B0728C : 0x358C > q


  Shell> mm 1b07288 2
  MEM  0x0000000001B07288 : 0x6D6D >
  MEM  0x0000000001B0728A : 0x55AA >        *Modified
  MEM  0x0000000001B0728C : 0x358C > q


* Operates with width = 4 bytes, type is IO:
  Shell> mm 80 4 ;IO
  IO  0x0000000000000080 : 0x000000FE >
  IO  0x0000000000000084 : 0x00FF5E6D > q


* To display PCI configuration space, ss=00, bb=00, ss=00, ff=00, rr=00:
  Shell> mm 0000000000 ;PCI
  PCI  0x000000000000000 : 0x86 >
  PCI  0x000000000000001 : 0x80 >
  PCI  0x000000000000002 : 0x30 >
  PCI  0x000000000000003 : 0x11 >
  PCI  0x000000000000004 : 0x06 >
  PCI  0x000000000000005 : 0x00 > q
* These contents can also be displayed by 'PCI 00 00 00'.


* To modify memory in non-interactive mode:
  Shell> mm 80 1 ;IO :52
  Shell> mm 80 1 ;IO
  IO  0x0000000000000080 : 0x52 > FE    *Modified
  IO  0x0000000000000081 : 0xFF >
  IO  0x0000000000000082 : 0x00 >
  IO  0x0000000000000083 : 0x00 >
```

```
IO  0x0000000000000084 : 0x6D >
IO  0x0000000000000085 : 0x5E >
IO  0x0000000000000086 : 0xFF >
IO  0x0000000000000087 : 0x00 > q
```

2.3.42 mode

```
MODE [row col]


    row     - Row number of the mode
    col     - Column number of the mode


Note:
    1. The mode with a star at line end is the current mode setting.
    2. When console redirection is turned on, only 80x25 mode is supported in
       current sample implementation.
    3. Most EFI command output was designed with 80x50 mode in mind (50 rows).
```

This command is used to change the display mode for the console output device. When this command is used without any parameters, it shows the list of modes that the standard output device currently supports. The **mode** command can then be used with the **row** and **col** parameter to change the number of rows and columns on the standard output device. The following example shows typical output for help on this command.

Examples

```
* To display available mode on standard output:
  Shell> mode
  Available modes on standard output
    col  80 row  25  *
    col  80 row  50
    col  80 row  43
    col 100 row 100
    col 100 row 999


* To change the current mode setting:
  Shell> mode 80 50
  Available modes on standard output
    col  80 row  25
    col  80 row  50  *
    col  80 row  43
    col 100 row 100
    col 100 row 999
```

2.3.43 mount

```
mount BlkDevice [sname]


    BlkDevice       - The name of the block device to mount
    sname           - The name of the newly mounted file system


Note:
    1. Mount uses the diskio protocol to read the FATxx format on a device. Name
       of mounted file system is stored in NVRAM for a given shell environment.
    2. The mounted names will be lost when "map -r" is called next time.
    3. If MOUNT without the second argument, it mounts the block device.
       Then there is an EFI_FILE_SYSTEM_PROTOCOL on the handle, but a drive name
       from the shell is not generated.
```

This command will define a mapping between a user defined name and a block device handle. The most common use of this command is to assign drive names to device handles that support a known file system protocol. Once these assignments are made, the drive names can be used with all the file manipulation commands. The following example shows typical output for help on this command. Please refer to the **map** command for further information on default mappings for all device handles that support recognized file systems.

Examples

```
* To mount device blk0 and name the file system fs0:
  Shell> map
  Device mapping table
    blk0 : VenHw(58C518B1-76F3-11D4-BCEA-0080C73C8881)/VenHw(0C95A92F-A006-
  11D4-BCFA-0080C73C8881)

  Shell> mount blk0 fs0

* To mount a block device without a name:
  Shell> mount blk1
```

2.3.44 mv

```
mv src [src...] [dst]

    src    - Source file/directory name (wildcards are permitted)
    dst    - Destination file/directory name (wildcards are not permitted)


 Note:
    1. If 'dst' is not specified, current directory is assumed to be the 'dst'.
    2. The attempt to move a read-only file/directory will result a failure.
```

This command moves a file or files from the path specified by **src** to the path specified by **dst**. This command can be used to rename file, or move one or more files from one directory into another directory. The following example shows typical output for help on this command.

Examples

```
* To rename a file:
  fs0:\> mv IsaBus.efi Bus.efi
  moving fs0:\IsaBus.efi -> \Bus.efi
   - [ok]

* To move a directory to current directory:
  fs0:\> mkdir test1\temp
  fs0:\> mv test1\temp
  moving fs0:\test1\temp -> \.\temp
   - [ok]

* To rename a directory:
  fs0:\> mv efi efi2.0
  moving fs0:\efi -> \efi2.0
   - [ok]

* To move multiple directories at a time:
  fs0:\> mv test1 test2 test
  moving fs0:\test1 -> \test\test1
   - [ok]
  moving fs0:\test2 -> \test\test2
   - [ok]

* To Move a read-only directory will result a failure:
  fs0:\test> attrib +r temp1
  DA  R fs0:\test\temp1
  fs0:\test> mv temp1 temp2
  moving fs0:\test\temp1 -> \test\temp2
   - error - Invalid Parameter
```

2.3.45 OpenInfo

```
OpenInfo Handle

    Handle     - The handle to show the open protocol information
```

This command is used to display the open protocols on a given handle. The example below is typical output from help for this command.

Examples

```
* To show open protocols on handle 23 that is the PCI Root Bridge.
* It shows that the PCI Root Bridge is being managed by the PCI
* Bus Driver, and the PCI Bus contains 7 PCI child controllers.
  Shell> openinfo 23
  Handle 23 (07DEE108)
  PciRootBridgeIo
    Drv[1D] Ctrl[23] Cnt(01) Driver   Image(PciBus)
    Drv[1D] Ctrl[28] Cnt(01) Child    Image(PciBus)
    Drv[1D] Ctrl[29] Cnt(01) Child    Image(PciBus)
    Drv[1D] Ctrl[2A] Cnt(01) Child    Image(PciBus)
    Drv[1D] Ctrl[2B] Cnt(01) Child    Image(PciBus)
    Drv[1D] Ctrl[2C] Cnt(01) Child    Image(PciBus)
    Drv[1D] Ctrl[2D] Cnt(01) Child    Image(PciBus)
    Drv[1D] Ctrl[2E] Cnt(01) Child    Image(PciBus)
    Drv[00] Ctrl[  ] Cnt(01) HandProt
  dpath
    Drv[1D] Ctrl[23] Cnt(01) Driver   Image(PciBus)
    Drv[00] Ctrl[  ] Cnt(0D) HandProt
```

2.3.46 pause

```
pause
```

```
Note:
    1. Only available in batch script files.
    2. The prompt message is "Enter 'q' to quit, any other key to continue".
```

The **pause** command prints a message to the display and then suspends batch file execution and waits for keyboard input. Pressing any key resumes execution, except for **q** or **Q**. If **q** or **Q** is pressed, batch processing terminates; otherwise execution continues with the next line after the pause command.

The **pause** command is available only in batch scripts.

The following example shows typical output for help on this command.

Example

```
* Following script is a sample of 'pause' command:
  fs0:\> type pause.nsh
  File: fs0:\pause.nsh, Size 204
  #
  #  Example script for 'pause' command
  #
  echo pause.nsh begin..
  date
  time
  pause
  echo pause.nsh done.

* To execute the script with echo on:
  fs0:\> pause.nsh
  +pause.nsh> echo pause.nsh begin..
  pause.nsh begin..
  +pause.nsh> date
  06/19/2001
  +pause.nsh> time
  00:51:45
  +pause.nsh> pause
  Enter 'q' to quit, any other key to continue:
  +pause.nsh> echo pause.nsh done.
  pause.nsh done.

* To execute the script with echo off:
  fs0:\> echo -off
  fs0:\> pause.nsh
  pause.nsh begin..
```

intel

```
06/19/2001
00:52:50
Enter 'q' to quit, any other key to continue: q
fs0:\>
```

2.3.47 pci

```
PCI [Bus Dev [Func] [-i] [-s [Seg]]]


Bus      - Bus number in hex
Dev      - Device number in hex
Func     - Function number in hex
Seg      - Segment number in hex
-i       - Information interpreted
-s       - Segment number specified


Note:
    1. If only Bus and Dev are specified, Func is set as default value(0).
    2. If segment number is not specified, Seg is set as default value(0).
```

Displays all the PCI devices or PCI configuration space. The following example shows typical output for help on this command.

Examples

```
* To display all the PCI devices found in the system:
Shell> PCI


   Seg  Bus  Dev  Func
   ---  ---  ---  ----
   00   00   00    00 ==> Bridge Device - Host/PCI bridge
             Vendor 0x8086 Device 0x1130 Prog Interface 0
   00   00   01    00 ==> Bridge Device - PCI/PCI bridge
             Vendor 0x8086 Device 0x1131 Prog Interface 0
   00   00   1E    00 ==> Bridge Device - PCI/PCI bridge
             Vendor 0x8086 Device 0x244E Prog Interface 0
   00   00   1F    00 ==> Bridge Device - PCI/ISA bridge
             Vendor 0x8086 Device 0x2440 Prog Interface 0
   00   00   1F    01 ==> Mass Storage Controller - IDE controller
             Vendor 0x8086 Device 0x244B Prog Interface 80
   00   00   1F    02 ==> Serial Bus Controllers - USB
             Vendor 0x8086 Device 0x2442 Prog Interface 0
   00   00   1F    03 ==> Serial Bus Controllers - System Management Bus
             Vendor 0x8086 Device 0x2443 Prog Interface 0
   00   00   1F    04 ==> Serial Bus Controllers - USB
             Vendor 0x8086 Device 0x2444 Prog Interface 0
   00   00   1F    05 ==> Multimedia Device - Audio device
             Vendor 0x8086 Device 0x2445 Prog Interface 0
   00   00   1F    06 ==> Simple Communications Controllers - Modem
             Vendor 0x8086 Device 0x2446 Prog Interface 0
   00   01   00    00 ==> Display Controller - VGA/8514 controller
             Vendor 0x1002 Device 0x5246 Prog Interface 0
   00   02   07    00 ==> Multimedia Device - Audio device
```

```
         Vendor 0x1274 Device 0x1371 Prog Interface 0
   00    02   0A    00 ==> Bridge Device - CardBus bridge
         Vendor 0x1180 Device 0x0476 Prog Interface 0
   00    02   0A    01 ==> Bridge Device - CardBus bridge
         Vendor 0x1180 Device 0x0476 Prog Interface 0


* To display the configuration space of function 0, device 0 on bus 0:
Shell> PCI 00 00 00 -i

  PCI Segment 00 Bus 00 Device 00 Func 00
  00000000: 86 80 30 11 06 00 90 20-02 00 00 06 00 00 00 00  *..0.... ........*
  00000010: 08 00 00 20 00 00 00 00-00 00 00 00 00 00 00 00  *... ............*
  00000020: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  00000030: 00 00 00 00 88 00 00 00-00 00 00 00 00 00 00 00  *................*

  00000040: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  00000050: 50 00 09 38 00 00 00 00-00 00 00 00 00 00 00 00  *P..8............*
  00000060: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  00000070: 00 00 18 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  00000080: DE 2C CF 00 00 00 00 00-09 A0 04 F1 00 00 00 00  *.,..............*
  00000090: 00 00 D6 FF FE FF 00 00-33 80 33 80 85 84 C4 00  *........3.3.....*
  000000A0: 02 00 20 00 07 02 00 1F-00 00 00 00 00 00 00 00  *.. ............*
  000000B0: 00 00 00 00 30 00 00 00-00 00 00 00 00 08 00 00  *....0...........*
  000000C0: 00 00 00 00 00 00 00 00-08 00 00 00 00 00 00 00  *................*
  000000D0: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  000000E0: 00 00 00 00 00 00 00 00-00 00 90 14 00 00 00 00  *................*
  000000F0: 00 00 00 00 74 F8 00 00-00 00 00 00 08 00 00 00  *....t...........*


  Vendor ID(0x0): 8086                  Device ID(0x2): 1130

  Command(0x4): 0006
    (00)I/O space access enabled:      0  (01)Memory space access enabled:    1
    (02)Behave as bus master:          1  (03)Monitor special cycle enabled:  0
    (04)Mem Write & Invalidate enabled: 0  (05)Palette snooping is enabled:    0
    (06)Assert PERR# when parity error: 0  (07)Do address/data stepping:       0
    (08)SERR# driver enabled:          0  (09)Fast back-to-back transact...:  0

  Status(0x6): 2090
    (04)New Capabilities linked list:  1  (05)66MHz Capable:                  0
    (07)Fast Back-to-Back Capable:     1  (08)Master Data Parity Error:       0
    (09)DEVSEL timing:              Fast  (11)Signaled Target Abort:          0
    (12)Received Target Abort:         0  (13)Received Master Abort:          1
    (14)Signaled System Error:         0  (15)Detected Parity Error:          0

  Revision ID(0x8):     02              BIST(0x0F):  Incapable
  Cache Line Size(0xC): 00              Latency Timer(0xD): 00
  Header Type(0x0E):    0, Single function, PCI device
```

```
Class: Bridge Device - Host/PCI bridge -
Base Address Registers(0x10):
    Start   Type    Space    Prefectchable?      Size       Limit
  ----------------------------------------------------------------
  20000000  Mem     32 bits  YES                 04000000   24000000
  ----------------------------------------------------------------
No Expansion ROM(0x30)

Cardbus CIS ptr(0x28):    00000000
Sub VendorID(0x2C):          0000        Subsystem ID(0x2E):        0000
Capabilities Ptr(0x34):        88
Interrupt Line(0x3C):          00        Interrupt Pin(0x3D):        00
Min_Gnt(0x3E):                 00        Max_Lat(0x3F):              00

* To display configuration space of function 0, device 0 on bus 0, segment 0:
Shell> PCI 00 00 00 -s 0

  PCI Segment 00 Bus 00 Device 00 Func 00
  00000000: 86 80 30 11 06 00 90 20-02 00 00 06 00 00 00 00  *..0.... ........*
  00000010: 08 00 00 20 00 00 00 00-00 00 00 00 00 00 00 00  *... ............*
  00000020: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  00000030: 00 00 00 00 88 00 00 00-00 00 00 00 00 00 00 00  *................*

  00000040: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  00000050: 50 00 09 38 00 00 00 00-00 00 00 00 00 00 00 00  *P..8............*
  00000060: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  00000070: 00 00 18 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  00000080: DE A8 CE 00 00 00 00 00-09 A0 04 F1 00 00 00 00  *................*
  00000090: 00 00 D6 FF FE FF 00 00-33 80 33 80 85 84 C4 00  *........3.3.....*
  000000A0: 02 00 20 00 07 02 00 1F-00 00 00 00 00 00 00 00  *.. ............*
  000000B0: 00 00 00 00 30 00 00 00-00 00 00 00 00 00 08 00  *....0...........*
  000000C0: 00 00 00 00 00 00 00 00-00 08 00 00 00 00 00 00  *................*
  000000D0: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  *................*
  000000E0: 00 00 00 00 00 00 00 00-00 00 A0 18 00 00 00 00  *................*
  000000F0: 00 00 00 00 74 F8 00 00-00 00 00 00 08 00 00 00  *....t...........*
```

2.3.48 reconnect

```
RECONNECT DeviceHandle# [DriverHandle# [ChildHandle#]] | [-r]


    DeviceHandle#   - Device handle (hex)
    DriverHandle#   - Driver handle (hex)
    ChildHandle#    - Child handle of device (hex)
    -r              - Reconnect drivers from all devices


Note:
    1. This command disconnects the drivers from the controller, just like
       disconnect, but it then immediately reconnects them.
    2. This command tests to see if drivers are following the
       EFI Driver Model.
```

Reconnects one or more drivers from a device. This involves disconnecting all the drivers from all the devices, and then reconnecting them. If a **DeviceHandle** is not specified, then the reconnect operation is performed on all the handles in the system. If a **DeviceHandle** is specified, then only **DeviceHandle** and the devices below **DeviceHandle** are reconnected.

Examples

```
  * To reconnect all drivers from all devices:
    Shell> reconnect -r

  * To reconnect all drivers from device 28:
    fs0:\> reconnect 28

  * To reconnect driver 17 from device 28:
    fs0:\> reconnect 28 17


  * To reconnect child 32 on driver 17 from device 28:
    fs0:\> reconnect 28 17 32
```

2.3.49 reset

```
RESET [-w [string]] [-s [string]]


   -w          - Performs a warm reset
   -s          - Performs a shutdown
   string      - String to be passed to reset service


Note:
   1. Not all systems implement -w option. This may mean different things
      depending on which BIOS EFI is implemented on.
   2. Reset will be guaranteed to reset the chipset as well as the processor
      when cold reset is called.
   3. If warm reset is not supported, then a cold reset will be performed.
   4. If a shutdown is not supported, then a cold reset will be performed.
```

Resets the system. The default is to perform a cold reset unless the **-w** parameter is specified. If **string** is specified, then it is passed into the Reset() function to provide additional information for the reason of the system reset request. The following example is typical output for help on this command.

Examples

```
Shell> reset


Shell> reset -w


Shell> reset -s
```

2.3.50   rm

```
RM [-q] file [file ...]
RM [-q] directory [directory ...]


    -q          - Quite mode, doesn't prompt user for a confirmation
    file        - File name (wildcards are permitted)
    directory   - Directory name (wildcards are permitted)


Note:
    1. To remove a read-only file/directory will result failure.
    2. If error occurs, RM will exit immediately and later files/directories
       will not be removed.
```

This command deletes one or more files or directories. The following examples show typical output for help on this command.

Examples

```
* To remove multiple directories at a time:
  fs0:\> ls test
  Directory of: fs0:\test

    06/18/01  01:01p <DIR>             512  .
    06/18/01  01:01p <DIR>               0  ..
    06/19/01  12:59a <DIR>             512  temp1
    06/19/01  12:59a <DIR>             512  temp2
            0 File(s)         0 bytes
            4 Dir(s)


* Error occurs and RM will exit:
  fs0:\> rm test\temp11 temp2
  rm: Cannot find 'fs0:\test\temp11' - Not Found


* To remove multiple directories with wildcards:
  fs0:\> rm test\temp*
  rm: Remove subtree 'fs0:\test\temp1' [y/n]? y
  removing fs0:\test\temp1\temp1.txt
   - [ok]
  removing fs0:\test\temp1\boot\nshell.efi
   - [ok]
  removing fs0:\test\temp1\boot
   - [ok]
  removing fs0:\test\temp1
   - [ok]
  rm: Remove subtree 'fs0:\test\temp2' [y/n]? y
  removing fs0:\test\temp2\temp2.txt
   - [ok]
```

```
      removing fs0:\test\temp2
       - [ok]


  * To remove a directory that contains a read-only file will fail:
      fs0:\> attrib +r test\temp1\readme.txt
       A  R fs0:\test\temp1\readme.txt

      fs0:\> rm test\temp1
      rm: Cannot open 'readme.txt' under 'fs0:\test\temp1' in writable mode
      - Access denied
```

2.3.51 set

```
SET [-d|-v|-b] [sname [value]]


    -d     - Deletes the environment variable
    -v     - Volatile variable
    -b     - Displays one screen at a time
    sname  - Environment variable name
    value  - Environment variable value


  Notes:
     1. See dmpstore command to see all of NVRAM variables used by the shell.
     2. Size of NVRAM for set command will depend on system implementation.
     3. May send NVRAM variables to /efi/boot/bootstr.nvr on files system if
        no NVRAM is implemented in the core EFI routines.
     4. SET values are stored in EFI NVRAM and will be retained between boots
        unless the option -v is specified.
```

This command is used maintain the environment variables that are available from the EFI environment. This command can display the environment variables, create new environment variables, change the value of existing environment variables, or delete environment variables. The **set** command will set the environment variable specified by **sname** to **value**. This form of the command can be used to create a new environment variable, or to modify an existing environment variable. If the **set** command is used without any parameters, then all the environment variables are displayed. If the **set** command is used with the **–d** option, then the environment variable specified by **sname** will be deleted. The following example shows the typical output from help for this command

Examples

```
   * To create an environmemt variable:
     Shell> set DiagnosticPath fs0:\efi\diag;fs1:\efi\diag

   * To Display environment variables:
     Shell> set
         path           : .
         diagnosticPath : fs0:\efi2.0\diag;fs1:\efi2.0\diag

   * To delete an environment variable:
     Shell> set -d diagnosticpath
     Shell> set
         path           : .

   * To changes an environment variable:
     fs0:\> set src efi
     fs0:\> set
         path : .;fs0:\efi\tools;fs0:\efi\boot;fs0:\
         src  : efi
```

```
        fs0:\> set src efi2.0
        fs0:\> set
            path : .;fs0:\efi\tools;fs0:\efi\boot;fs0:\
            src  : efi2.0


    * To set a volatile variable which will be disappear at next boot:
      Shell> set -v EFI_SOURCE c:\project\EFI2.0
      Shell> set
          path        : .;fs0:\efi\tools;fs0:\efi\boot;fs0:\
        * EFI_SOURCE : c:\project\EFI2.0
```

2.3.52 setsize

```
setsize newsize file

    newsize  - The new size of the file in bytes
    file     - The file to be resized(supports asterisk wildcard)

Note:
    1. To set file size smaller will truncate the tail of file.
    2. To set file size larger will pad the back part with zero.
    3. The size of an Unicode file shall be an even number.
```

This command sets the size of the file specified by **file** to **newsize** bytes. This command can be used to either shrink or grow an existing file.

Example

```
* Set the size of a file:
  Shell> setsize 100 fs0:\a.txt
  setsize: fs0:\a.txt 100

* Set the sizes of multiple files:
  Shell> setsize 100 fs0:\*.txt
  setsize: fs0:\a.txt 100
  setsize: fs0:\b.txt 100
  setsize: fs0:\c.txt 100
```

2.3.53 stall

```
STALL microseconds


    microseconds    - Microseconds to stall
Note:
    1. To STALL in emulation NT environment will sleep for 'microseconds'.
    2. To STALL in some other platforms will wait for 'microseconds.
    3. Microseconds is in decimal.
```

This command stalls the processor for the number of microseconds specified by **microseconds**. The following example is typical output for help on this command.

Examples

```
Shell> stall 1000000
Stall for 1000000 uS
```

2.3.54 time

```
TIME [hh:mm[:ss]]


    hh    - Hour of time
    mm    - Minute of time
    ss    - Second of time


Note:
    1. Hour and minute are required to set the time.
    2. If second is not specified, 0 will be used as default.
```

This command displays to sets the current time for the system. If no parameters are used, it shows the current time. If valid hours, minutes, and seconds are provided, then the system's time will be updated. The following example shows typical output from help for this command.

Examples

```
* To display current time:
  fs0:\> time
  16:51:03

* To set the system time:
  fs0:\> time 9:51:30
  fs0:\> time
  09:51:31
```

2.3.55 touch

```
touch [-r] filename


    -r    - Recursive to subdirectories
```

This command updates the time and date on file specified by **filename** to the current time and date. The following example is typical output from help for this command.

Examples

```
* To touch a file, the time of file will be changed after TOUCH:
  fs0:\> ls for.nsh
  Directory of: fs0:\
    06/18/01  09:32p                    153  for.nsh
            1 File(s)        153 bytes
            0 Dir(s)


  fs0:\> touch for.nsh
  touch: fs0:\for.nsh [ok]


  fs0:\> ls for.nsh
  Directory of: fs0:\
    06/19/01  09:54a                    153  for.nsh
            1 File(s)        153 bytes
            0 Dir(s)


* To touch a directory recursively:
  fs0:\> touch -r efi2.0
  touch: fs0:\efi2.0 [ok]
  touch: fs0:\efi2.0\boot [ok]
  touch: fs0:\efi2.0\boot\nshell.efi [ok]
```

2.3.56 type

```
type [-a|-u] [-b] file [file...]

    -a      - Displays the file as ASCII characters
    -u      - Displays the file as Unicode characters
    -b      - Displays one screen at a time
    file    - Name of file to display
```

This command sends the contents of a file to the standard output device. If no options are used, then the file type is auto-detected and sent to the standard output device. If the **–a** option is used, the file is sent to the standard output device as a stream of ASCII characters. If the **–u** option is used, the file is sent to the standard output device as a stream of Unicode characters. The following example shows typical output for help on this command.

Examples

```
* To displays the file as Unicode characters:
  fs0:\> type -u pause.nsh
  File: fs0:\pause.nsh, Size 204
  #
  #  Example script for 'pause' command
  #
  echo pause.nsh begin..
  date
  time
  pause
  echo pause.nsh done.

* To displays the file as ASCII characters:
  fs0:\> type -a pause.nsh
  File: fs0:\pause.nsh, Size 204
   #
   #    E x a m p l e   s c r i p t   f o r   ' p a u s e '   c o m m a n d
   #
   e c h o   p a u s e . n s h   b e g i n . .
   d a t e
   t i m e
   p a u s e
   e c h o   p a u s e . n s h   d o n e .

* To type multiple files at a time:
  fs0:\> type test.*
  File: fs0:\test.txt, Size 23
          How to Install?
  File: fs0:\test.nsh, Size 48
  time
```

```
stall 3000000
time
```

2.3.57 Unload

```
unload [-n][-v] HandleIndex


    -n           - No prompt
    -v           - Verbose
    HandleIndex  - Handle of protocol to unload


 Note:
    1. This command is obsolete.
    2. LOAD is opposite.
```

This command is used to unload an image from memory. The **HandleIndex** comes from the output of the **dh** shell command. The following example is typical output from help for this command.

Examples

```
 * To find the handle index protocol image to unload:
   Shell> dh -b
   Handle dump
     1: Image(DXE Core)
     2: FwVol FwFileSys FwVolBlk DevPath(MemMap(11:1760000-189FFC8))
        ...
    27: Image(Reset)
    28: Image(WinNtBlockIo) DriverBinding
    29: Image(Timer)
       ...

 * To unload the protocol image of 'Reset':
   Shell> unload 27
    27: Image(Reset)
   Unload protocol image (y/n)? n
   Exit status code: Aborted
```

2.3.58 ver

ver

Displays the version information for this EFI Firmware. This information is retrieved through the EFI System Table. The following example shows typical output for help on this command.

Examples

```
* To display version information of a platform:
  fs0:\> ver
  EFI Specification Revision    2.0
    EFI Vendor    = INTEL
    EFI Revision   = 8192.1


* To display version information of another platform:
  fs0:\> ver
  EFI Specification Revision     1.02
  EFI Vendor        = INTEL
    EFI Revision       = 12.38

  SAL Specification Revision      3. 0
    SAL_A Revision    = 1. 1
    SAL_B Revision    = 1. 1

  PAL_A Revision         66.23
  PAL_B Revision         66.23

  Other modules mentioned in FIT (Firmware Interface Table)
  FIT_Entry Type   0, Revision    2.60
  FIT_Entry Type  15, Revision    66.23
  FIT_Entry Type  16, Revision    0.90
  FIT_Entry Type  32, Revision    0.30
  FIT_Entry Type  30, Revision    1. 0
  FIT_Entry Type  17, Revision    0.90
  FIT_Entry Type  18, Revision    6. 0
  FIT_Entry Type  20, Revision    0.80

 SalProc Entry 000000003FE3F720 and GP 000000003FF22480
 PalProc Entry 000000003FF48010 IO Port Base 00000FFFFC000000
 Cache Enabled
```

2.3.59 vol

```
vol [fs] [Volume Label]


    fs              - The name of the file system
    Volume Label    - New volume label
```

Displays volume information for the file system specified by **fs**. If **Volume Label** is specified, then the volume label for **fs** will be set to **Volume Label**. The maximum length for **Volume Label** is 11 characters. The following example shows typical output for help on this command.

Examples

```
* To display the volume of current fs:
  fs0:\> vol
  Volume has no label (rw)
        1,457,664 bytes total disk space
        1,149,440 bytes available on disk
              512 bytes in each allocation unit

* To change the label of a fs:
  fs0:\> vol fs0 help_test
  Volume HELP_TEST (rw)
        1,457,664 bytes total disk space
        1,149,440 bytes available on disk
              512 bytes in each allocation unit
```