congatec Application Note

Affected Products	All Products Featuring UEFI
Subject	UEFI Shell Debugging
Confidential/Public	Public
Author	CJR

Revision History

Revision	Date (yyyy-mm-dd)	Author	Changes
1.0	2021-05-25	CJR	Initial release of document.
1.1	2021-08-27	CJR	Corrected the BootCurrent variable name in section 2.2.3. Added setvar command to section 2.2.9.

Preface

This application note describes helpful UEFI Shell commands for system debugging and diagnosis. The UEFI Shell can be executed on all congatec x86 products with UEFI boot firmware.

Disclaimer

The information contained within this Application Note, including but not limited to any product specification, is subject to change without notice.

congatec GmbH provides no warranty with regard to this Application Note or any other information contained herein and hereby expressly disclaims any implied warranties of merchantability or fitness for any particular purpose with regard to any of the foregoing. congatec GmbH assumes no liability for any damages incurred directly or indirectly from any technical or typographical errors or omissions contained herein or for discrepancies between the product and the Application Note. In no event shall congatec GmbH be liable for any incidental, consequential, special, or exemplary damages, whether based on tort, contract or otherwise, arising out of or in connection with this Application Note or any other information contained herein or the use thereof.

Intended Audience

This Application Note is intended for technically qualified personnel. It is not intended for general audiences.

Electrostatic Sensitive Device

All congatec GmbH products are electrostatic sensitive devices and are packaged accordingly. Do not open or handle a congatec GmbH product except at an electrostatic-free workstation. Additionally, do not ship or store congatec GmbH products near strong electrostatic, electromagnetic, magnetic, or radioactive fields unless the device is contained within its original manufacturer's packaging. Be aware that failure to comply with these guidelines will void the congatec GmbH Limited Warranty.

Technical Support

congatec GmbH technicians and engineers are committed to providing the best possible technical support for our customers so that our products can be easily used and implemented. We request that you first visit our website at www.congatec.com for the latest documentation, utilities and drivers, which have been made available to assist you. If you still require assistance after visiting our website then contact our technical support department by email at support@congatec.com

Symbols

The following are symbols used in this application note.

Note

Notes call attention to important information that should be observed.

Caution

Cautions warn the user about how to prevent damage to hardware or loss of data.

Warning

Warnings indicate that personal injury can occur if the information is not observed.

Copyright Notice

Trademarks

Product names, logos, brands, and other trademarks featured or referred to within this user’s guide or the congatec website, are the property of their respective trademark holders. These trademark holders are not affiliated with congatec GmbH, our products, or our website.

Terminology

Term	Description
NVRAM	Non-volatile RAM
SMBIOS	System Management BIOS
UEFI	Unified Extensible Firmware Interface

1 Introduction

Like MS-DOS on legacy PCs, the UEFI Shell provides an API, a command prompt, and a rich set of commands. However, the UEFI firmware is not compatible with the old MS-DOS operating system.

The UEFI Shell can be used for testing, debugging, and diagnosing systems. The shell can run as a built-in UEFI Shell or on a bootable USB flash drive compiled with UEFI Shell. The congatec embedded BIOS does not support the built-in UEFI Shell because of security reasons. Having a built-in shell may allow users to bypass security mechanisms that are contained within the BIOS.

To start the UEFI Shell on congatec x86 products, you need to create a bootable USB flash drive with compiled UEFI Shell. The procedure to create such a boot USB flash drive is explained in the application note "AN31_Creating_a_UEFI_Stick". You can download this document from:

https://www.congatec.com/us/support/application-notes/article/an31-how-to-create-a-bootable-usb-stick-with-an-uefi-shell/

This application note explains commonly used UEFI Shell commands during system development and production.

For additional information about the UEFI Shell, refer to the UEFI Shell specification. The document can be downloaded from https://www.uefi.org/specifications.

2 Common UEFI Shell Commands

This section describes some commonly used UEFI Shell commands for shell navigation, file management, system debugging and testing.

The commands can be used at the UEFI Shell prompt **`Shell>`** or at the file system prompt, typically **`FS0:\>`**.

2.1 Basic Commands

The following commands can be used for shell navigation or file management.

2.1.1 help

The **`help -b`** command lists all the UEFI Shell commands.

The **`-b`** parameter displays the output one screen at a time.

The **`-verbose`** parameter displays the details of each UEFI Shell command. It also shows an example of how to use the command. For example, the command:

```
help -b -verbose pci
```

will parse and pause on each screen, giving verbose description of the **`pci`** command.

2.1.2 cls

The **`cls`** command can be used to clear the screen and optionally change the background and foreground color of the display.

To clear the screen without changing the background or foreground color, type:

```
cls
```

To clear the screen and change the background color to blue and the foreground color to yellow, type:

```
cls 1 6
```

where 1 is the blue background color and 6 is the yellow foreground color.

2.1.3 exit

The **`exit`** command is used to exit the UEFI Shell. If other boot options are available after exiting the shell, the system will boot from another loader. Otherwise, BIOS setup will be launched.

2.1.4 mode

The **`mode`** command can change the number of characters per line and the number of lines that can be displayed.

You can check the assigned mode for your display by typing the command:

```
mode
```

Typical modes supported by an HD display are:

Col	80	Row	25
Col	80	Row	50
Col	100	Row	31
Col	240	Row	56

For example, to change the display mode to 100 characters per line and 31 lines, use:

```
mode 100 31
```

2.1.5 Output Redirection

The output of UEFI Shell commands can be redirected to a file. This is especially important during system debugging. The stored file can be analyzed later or shared with others for easy collaboration. Output redirection works with all UEFI Shell commands.

The command below for example, redirects the help text for the pci command to a UCS-2 encoded file pcihelp.txt.

```
help pci > pcihelp.txt
```

Although there are several special character combinations for the output redirection operation, the "**>**" character in the example above is commonly used for output redirection.

2.1.6 type / edit

The **`type`** and **`edit`** commands send the content of a file to the standard output device. While the type command allows you to only view the text file (without modification), the edit command allows you to view and modify the text file.

For example, the type command below will display the content of the pcihelp.txt file on the screen:

```
type pcihelp.txt
```

In contrast, the edit command below will open the pcihelp.txt file in full-screen text editor mode with full modification capability.

```
edit pcihelp.txt
```

The editor supports UCS-2 and ASCI encoded text files. In the editor, you can use the arrow keys as well as page up and page down keys to move the cursor around. The editor does not have a menu and must be controlled with the function (Fx) keys.

The table below shows the supported keys with their associated functions.

F1	Go to line	**F5**	Search and replace a string
F2	Save file	**F6**	Delete line
F3	Save a modified file	**F7**	Insert previously deleted line
F4	Search a string	**F8**	Open a new file

2.1.7 hexedit

With **hexedit** command, you can view and modify binary files in UEFI Shell. You can also use it for block devices and memory regions.

For example, to open the binary file dump.bin in a full screen hex editor, type:

```
hexedit dump.bin
```

To open block device fs0 for editing, starting from block 0 with size of 1 block, type:

```
hexedit -d fs0 0 1
```

To open the memory region for editing, starting from address 0x8CC80B98 with size of 100 bytes, type:

```
hexedit -m 8CC80B98 100
```

In the editor, you can use the arrow keys as well as the page up and page down keys to move the cursor around. The editor does not have a menu and must be controlled with the function (Fx) keys.

The table below shows the supported keys with their associated functions.

F1	Go to offset	F6	Delete block
F2	Save buffer/file	F7	Insert previously deleted block
F3	Save a modified buffer/file	F8	Open a new file
F4	Mark block start	F9	Open a new block device
F5	Mark block end	F10	Open a new memory region

Caution

Ensure adequate care is taken when editing block devices and memory at runtime. Failure to do so may cause loss of data or system crash.

2.1.8 ver

The **ver** command displays the UEFI Firmware and the UEFI Shell versions. With **-s** parameter, the command displays only the UEFI Shell version.

For example, **ver -s** displays **2.2**. This is the current version of the UEFI Shell as at the date of this document.

2.2 Debugging Commands

The following commands can be used in the UEFI Shell to diagnose and debug the system.

2.2.1 bcfg

The **bcfg** command can be used to display and modify the boot option variables stored in the UEFI firmware NVRAM.

With different bcfg command options, you can add, remove, or reorder boot option variables.

Note

You must change the setup node "Boot Priority Selection" to "UEFI Standard" in order to support modification of boot variable ordering at runtime.

The most common bcfg command options are listed below:

- **bcfg boot dump**
  Displays all boot options
- **bcfg boot add 2 OEMOS.efi "OEM OS"**
  Adds the boot loader OEMOS.efi in the root directory, as new boot option #2
- **bcfg boot mv 1 0**
  Moves boot option #1 to #0
- **bcfg boot mv 1 0**
  Removes boot option #0

The syntax for the boot option variables is **Bootxxxx** (where xxxx is a four-digit number)

2.2.2 dmem / mem

Both **dmem** and **mem** commands display the content of system memory or memory-mapped device.

The mem command is a built-in alias for the dmem command.

Syntax:

```
dmem [-b] [address] [size]
```

The address and size must be typed in hex values. For example, to display 100 bytes of system memory, starting from address 0x90000000, type:

```
dmem 90000000 100
```

If the size is not specified, the default size of 512 bytes is displayed.

The **dmem** command without a parameter displays the contents of the EFI System Table. To redirect the contents of the UEFI System Table to a file UST.txt, type:

```
dmem > UST.txt
```

2.2.3 dmpstore

The **dmpstore** command is used to display and manage UEFI variables. The main debugging purpose of this command is to dump certain variables in the UEFI Shell.

Some examples of the dmpstore command are listed below:

- Dump all variables to a file DumpVar.txt. The file can be analyzed on a host PC using a text editor.

  ```
  dmpstore > DumpVar.txt
  ```

- Dump the EFIGlobalVariable "BootOrder". The output helps to understand boot priority handling of the UEFI firmware.

  ```
  dmpstore BootOrder
  ```

- Dump the EFIGlobalVariable "BootCurrent". The output helps to understand boot priority handling of the UEFI firmware.

  ```
  dmpstore BootCurrent
  ```

- Dump the Secure Boot Platform Key variable one screen at a time.

  ```
  dmpstore *PK* -b
  ```

- Dump the Secure Boot Key Exchange Key variable one screen at a time.

  ```
  dmpstore *KEK* -b
  ```

- Dump the Secure Boot db and dbx variables one screen at a time.

  ```
  dmpstore *DB* -b
  ```

Note

The asterisk (*) can be used as a placeholder for the variable name. All variables that match the naming convention will be dumped.

2.2.4 memmap

The **memmap** command displays the memory map maintained by the UEFI environment. The UEFI firmware keeps track of all the physical memory in the system and how it is being used.

For the description of how the Memory Attribute Descriptors are used, refer to the UEFI specification.

Examples:

To display the memory map one page at a time, type:

```
memmap -b
```

To dump the memory map to a file MemoryMap.txt, type:

```
memmap > MemoryMap.txt
```

2.2.5 mm

The **mm** command can be used to display or modify memory, memory mapped IO, PCI(e) configuration space and legacy IO.

Syntax

mm address [value] [-w 1|2|4|8] [-MEM | -MMIO | -IO | -PCIE] [-n]

The parameters and options are described below:

- **address**
  This parameter is mandatory and must be in hexadecimal format. It specifies the start address of the memory area or the IO register to be modified or read from.
- **value**
  If value is specified, the mm command writes the value to the address location without user interaction. This parameter must be in hexadecimal format.
- **-w**
  The -w option defines the access width in bytes. Default width is 1 byte.
- **-n**
  The -n option switches the command to non-interactive mode and displays the value of the specified address. This value cannot be overridden. Without -n option, the user can edit the value of the specified memory or IO location.
- **-MEM, -MMIO, -IO, -PCIE**
  These parameter options define the address type. If none of these types is specified, **-MEM** is assumed.

Examples

Some examples of the mm command are listed below:

- Read one byte of memory in interactive mode

  ```
  mm 0x40000000
  Output: MEM 0x0000000040000000 : 0xC2 > _
  ```

  In the interactive mode, you can override the current value or press the Enter key to read the next byte of data.

- Read four bytes of memory in non-interactive mode

  ```
  mm 0x40000000 -w 4 -n
  Output: MEM 0x0000000040000000 : 0x1E4E3EC2
  ```

- Write two bytes of memory (writes are always non-interactive)

  ```
  mm 0x40000000 0x1234 -w 2
  ```

- Read one byte of IO port 80h (BIOS POST code port) in interactive mode

  ```
  mm 0x80 -IO
  Output: MEM 0x0000000000000080 : 0x78 > _
  ```

  On congatec evaluation carrier boards, you can see this IO port on a 7-segment display.

- Read the Vendor and Device ID in PCI Express configuration space

  ```
  mm 0x0000000200000 -w 4 -PCIE -n
  Output: PCIE 0x0000000000200000 : 0x3E9B8086
  ```

  You do not need Interactive mode for Vendor and Device ID because these registers are read only.

In addition to the **-PCIE** address type, the mm command also supports a **-PCI** address type. The PCI address type supports only 256 bytes of config space, whereas PCIE address type supports 4k bytes. Therefore, their address format is slightly different.

The **-PCIE** format has 3-digit register while the **-PCI** format has 2-digit register. The formats are:

-PCIE: ssssbbddffrrr

-PCI: ssssbbddffrr

where ssss = Segment, bb = Bus, dd = Device, ff = Function and rr or rrr = Register.

Note

With the -PCIE address type, you can read or modify the PCI config space by setting the uppermost digit of the register in -PCIE format to 0.

2.2.6 pci

The **pci** command displays the complete PCI device list. It can also display the configuration space of a PCI or PCIe device.

To display the PCI device list one screen at a time, type:

```
pci -b
```

To redirect the output of the pci command to a file pcitree.txt, type:

```
pci > pcitree.txt
```

When you type the pci command without a Bus Device Function (BDF) parameter, it lists the PCI device tree. When you type it with the BDF parameters, the command reads the config space of the selected device. For example:

```
pci 0 1F 6
```

The above command reads the 256 bytes configuration registers of the PCI function at bus 0, device 1Fh and function 6. The BDF numbers must be hexadecimal numbers.

The config space of a PCI device is only 256 bytes and can be displayed on one screen. For PCI Express devices with a 4k config space, use the **-b** parameter to pause the output after one page on the screen. For example:

```
pci 0 2 0 -b
```

Use the **-i** parameter to display interpreted (verbose) information for the configuration space registers. For example:

```
pci 0 17 0 -i
```

Note

The pci command only displays the PCI and PCI Express configuration space registers. To modify these registers, use the mm command.

2.2.7 reset

The **reset** command is used to reset the system. The command has three parameter options:

- **-s** for shutdown (transition to SoftOff (S5), no reboot)
- **-w** for warm boot (system reset without power cycle)
- **-c** for cold boot (system reboot with power off/on cycle)

If no parameter is used, the **reset** command will perform a cold reboot by default. With **-fwui** parameter, the UEFI firmware will launch its user interface (BIOS setup) on the next boot.

Examples:

reset -s

Performs a system shutdown

reset -c

Performs a cold reset of the system

reset -w -fwui

Performs a warm reset of the system and enters BIOS setup on next boot

2.2.8 sermode

The **sermode** command displays or sets baud rate, parity attribute, data bits and stop bits of a serial port. If no attributes are specified, then the current settings for all serial ports are displayed.

For example, to display the settings of all serial ports, type:

sermode

The output of the command on our test system is shown below:

```
150(88414B18) - (115200, N, 8, 1)
151(88414118) - (115200, N, 8, 1)
```

The system has two serial ports with device handle 150 and 151 enabled. Both serial ports have 115200 baud rate, no parity, 8 data and 1 stop bit.

To change the settings for the second serial port to 9600 baud rate and even parity, type:

sermode 151 9600 e 8 1

A confirmation message "**Mode set on handle 151 successfully**" displays.

Use the sermode command to verify the new settings:

sermode

This displays the following:

```
150(88414B18) - (115200, N, 8, 1)
151(88414118) - (9600, E, 8, 1)
```

For a list of all supported baud rate, parity, data and stop bit attribute settings, type:

help sermode -b

2.2.9 setvar

Similar to the **dmpstore** command, the **setvar** command can be used to display, create, delete or modify a UEFI variable. Unlike the **dmpstore** command, which can only load a variable from a file, the **setvar** command can directly modify a variable.

Some examples of the setvar command are listed below:

- Display the EfiGlobalVariable "BootCurrent" (similar to **dmpstore BootOrder**).

  ```
  setvar BootOrder
  ```

- Write a new value to the EfiGlobalVariable "BootCurrent". With **dmpstore** command, this action is not possible.

  ```
  setvar BootOrder =0x00030002
  ```

  In addition to the "0x" indicator for a hexadecimal number, you can also use the "H" notation for a hexadecimal byte array. For example:

  ```
  setvar BootOrder = H02000300
  ```

  (Both commands will write the same value to the BootCurrent variable)

- Display only the variable named "testvar1" associated with the given GUID value.

  ```
  setvar testvar1 -guid 12345678-1111-2222-3333-1234567890AB
  ```

  To access a variable that is not EFI_GLOBAL_VARIABLE, you must use the -guid option.

- Delete the variable named "testvar1" associated with the given GUID value.

  ```
  setvar testvar1 -guid 12345678-1111-2222-3333-1234567890AB =
  ```

Note

Having the equal sign "=" at the end of the setvar command (without data assigned) will delete the variable "testvar1".

2.2.10 smbiosview

The **smbiosview** command displays the SMBIOS information. When you type the command without parameter or with parameter **-a**, it displays the information for all SMBIOS structures. For example:

```
smbiosview or smbiosview -a
```

The **-t** parameter allows you to select and display a specific SMBIOS structure type.

To get an overview of all supported SMBIOS types, type:

```
help smbiosview -b
```

To save all SMBIOS structures to a file smbiosdata.txt, type:

```
smbiosview -a > smbiosdata.txt
```

To display SMBIOS structure 1 (system information), type:

```
smbiosview -t 1 -b
```