AMD BIOSDBG User's Manual

Revision 1.2

Revision	Date
Draft 0.1	2011/4/7
Draft 0.3	2011/6/12
Revision 1.0	2011/7/1
Revision 1.1	2012/4/6
Revision 1.2	2012/6/7

Contents

1 Overview

1.1 Introduction

AMD BIOSDBG is a source level debugger for SimNow™, Purple Possum and Wombat. It communicates with SimNow™ using the SimNow™ Command API interface. AMD BIOSDBG must be run on the same machine as SimNow™. When debugging through Purple Possum, AMD BIOSDBG uses the AMD USB driver installed with HDT. When source code is not available, AMD BIOSDBG still has features that are useful.

1.2 Supported CPUs

AMD BIOSDBG supports debugging of AMD family 10h and newer CPUs. Support will be added to AMD BIOSDBG updates for new silicon.

1.3 Project build settings for source level debugging

In some cases BIOS projects will already support source level debugging. If the build process creates .pdb files, then debug information for 32-bit or 64-bit modules is present. Debug information for 16-bit code can be confirmed by the presence of 'Line numbers for' in .map files. Here are command line tool options for building with the debug information needed by AMD BIOSDBG:

Code Type	Compile flags	Link flags
16-bit	/Zd	/linenumbers
32-bit	/Zi	/debug
64-bit	/Zi	/debug

Source level debugging of 16-bit code requires a line number section in the 16-bit link map files (*.map). Source level debugging of 32 and 64 bit code requires program database files (*.pdb). At startup, AMD BIOSDBG searches the project directory for executable files with extension *dll* or *efi*. If a pdb file for the executable file is found, AMD BIOSDBG loads it for use during the debug session. Directory names containing the string 'tools' or 'compilers' are not searched. Files with extension *map* are searched looking for line number sections used for debugging 16-bit code.

By default, AMD BIOSDBG uses the location of the BIOS image to define the top level directory of the project. This directory is the search starting point for map and pdb files.

Notes: To enable the debug info to generate pdb files in UEFI code base, the following may need to be enabled:

- ***EFI_SYMBOLIC_DEBUG*** for Insyde and Phoenix codes
- ***DEBUG_MODE*** and ***DEBUG_INFO*** for AMI Aptio code

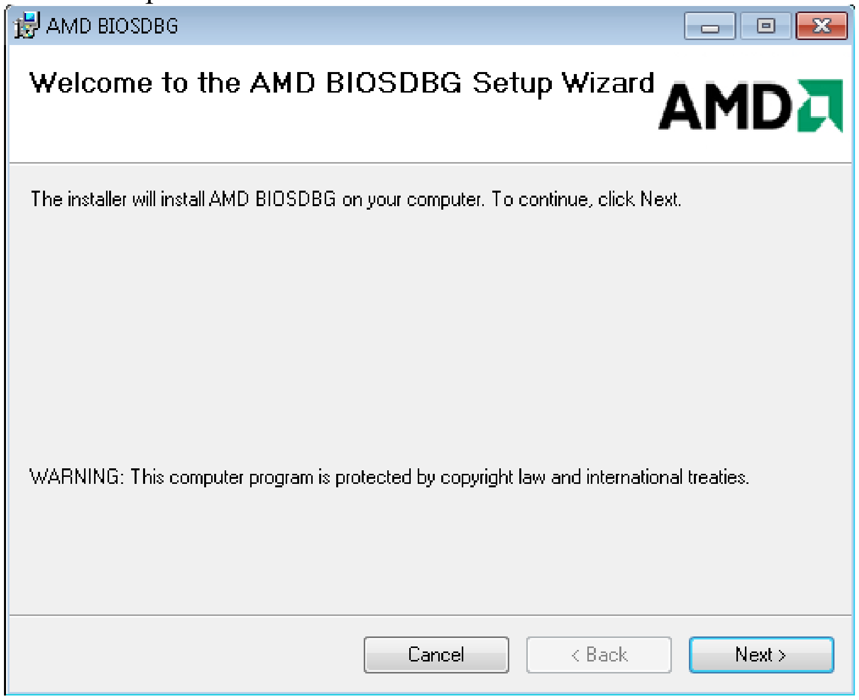
2 Installation

2.1 System Requirements

AMD BIOSDBG runs on Windows 32-bit or 64-bit system when debugging with Purple Possum or Wombat, and only runs on Windows 64-bit systems when debugging SimNow ™.

2.2 Installation Procedure

Click "setup.exe":



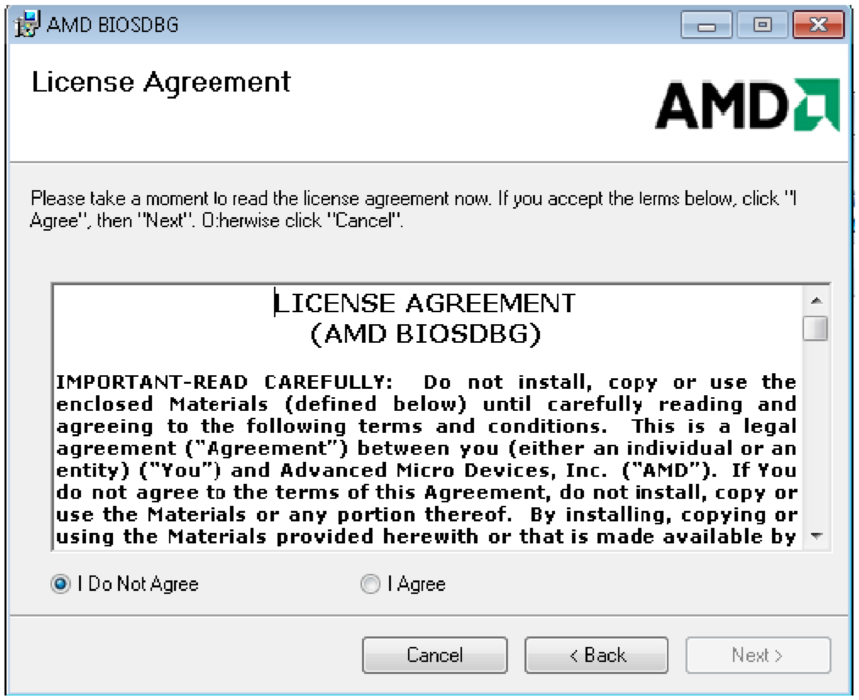
AMD BIOSDBG

Welcome to the AMD BIOSDBG Setup Wizard

The installer will install AMD BIOSDBG on your computer. To continue, click Next.

WARNING: This computer program is protected by copyright law and international treaties.

Click "Next".



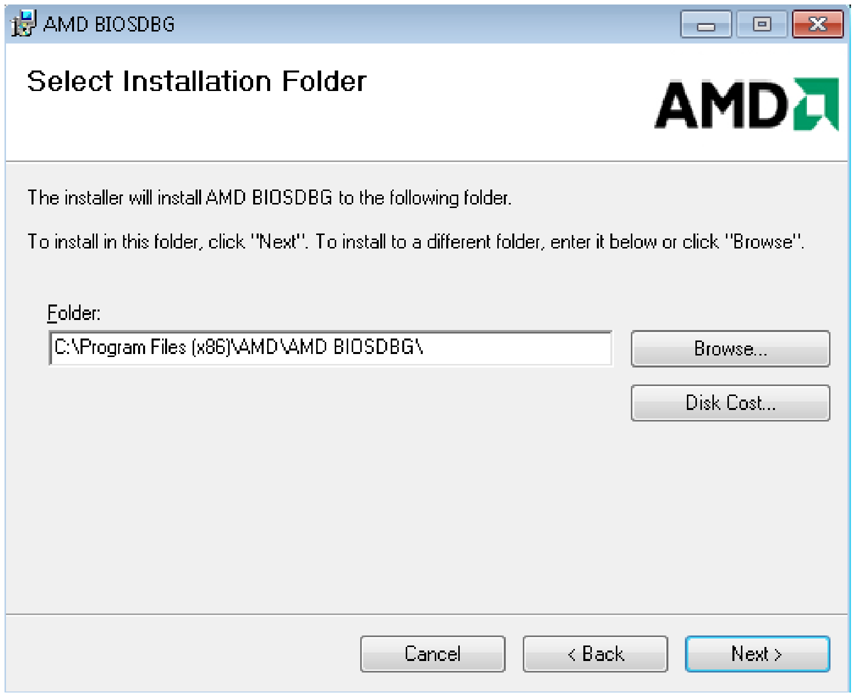
Please take a moment to read the license agreement now. If you accept the terms below, click "I Agree", then "Next". Otherwise click "Cancel".

LICENSE AGREEMENT
(AMD BIOSDBG)

IMPORTANT-READ CAREFULLY: Do not install, copy or use the enclosed Materials (defined below) until carefully reading and agreeing to the following terms and conditions. This is a legal agreement ("Agreement") between you (either an individual or an entity) ("You") and Advanced Micro Devices, Inc. ("AMD"). If You do not agree to the terms of this Agreement, do not install, copy or use the Materials or any portion thereof. By installing, copying or using the Materials provided herewith or that is made available by

[✓ I Do Not Agree] I Agree

The "License Agreement" must be agreed before you can install and use AMD BIOSDBG. Click "I Agree" and click "Next" for further installation.

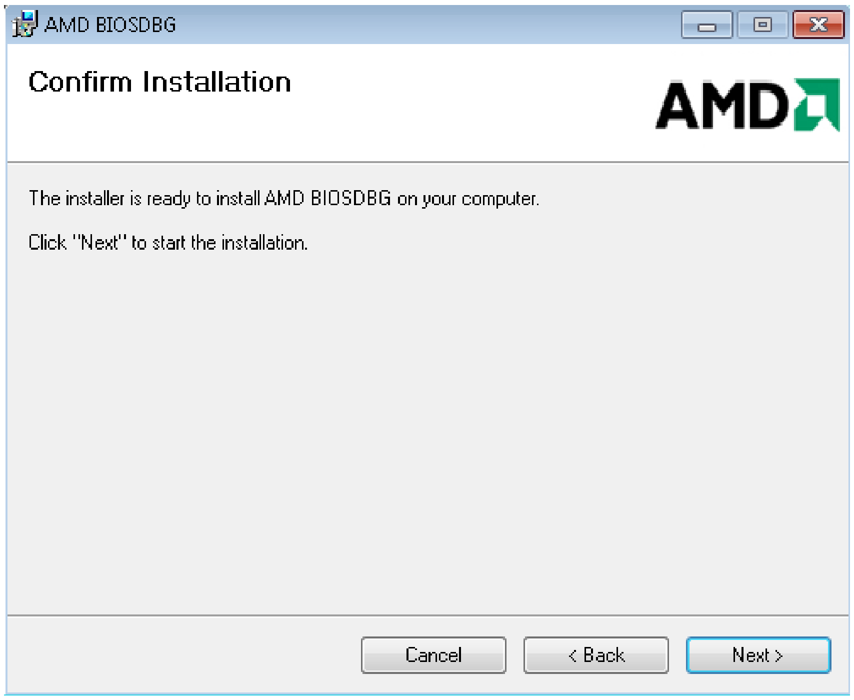


The installer will install AMD BIOSDBG to the following folder.

To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".

Folder: C:\Program Files (x86)\AMD\AMD BIOSDBG\

Change the installation folder if it is desired to install to a different folder.



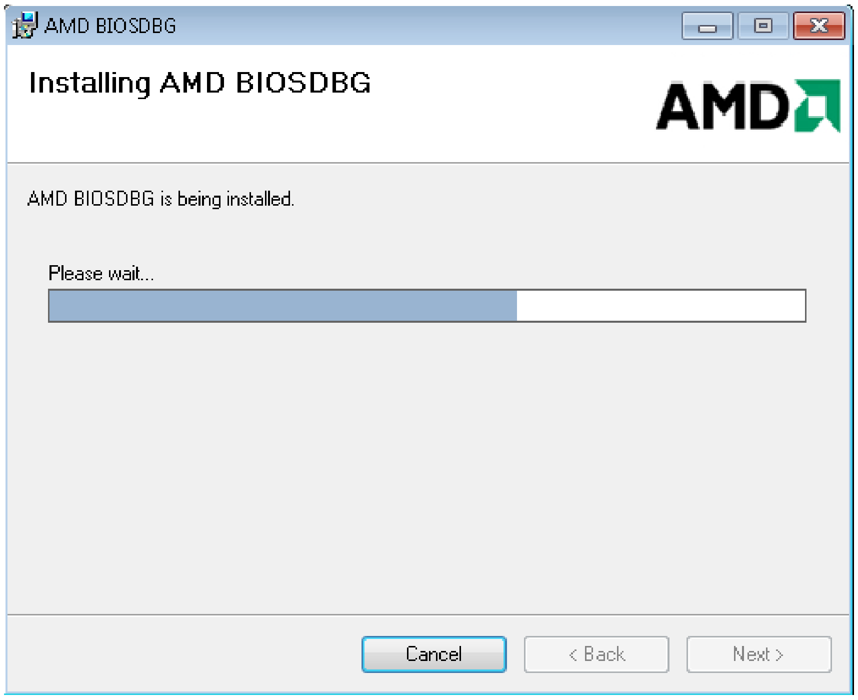
AMD BIOSDBG

Confirm Installation

The installer is ready to install AMD BIOSDBG on your computer.

Click "Next" to start the installation.

Confirm installation; click "Next" to continue.



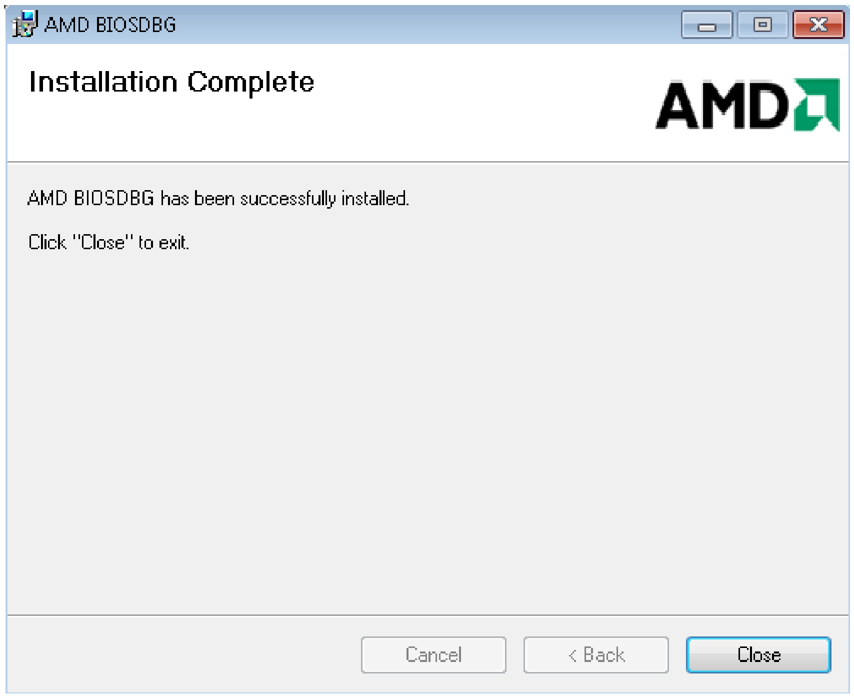
AMD BIOSDBG

Installing AMD BIOSDBG

AMD BIOSDBG is being installed.

Please wait...

Wait for the installation to complete.



AMD BIOSDBG has been successfully installed.

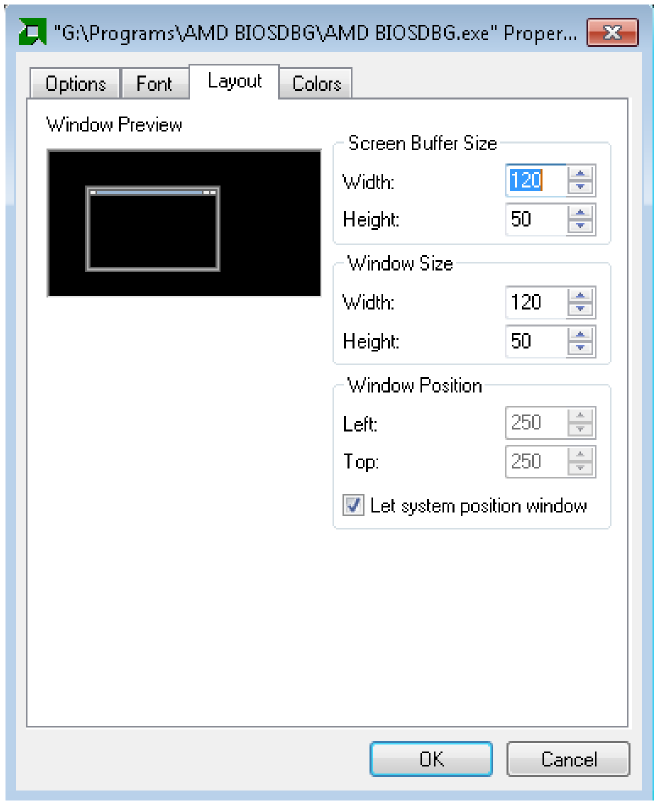
Click "Close" to exit.

When installation has completed, click “Close”.

3 Program startup

3.1 Preparation

It's ideal to set the prompt window size and screen buffer size to 120 * 50.If the active code page is not US English, it can be set by "chcp 437".


"G:\Programs\AMD BIOSDBG\AMD BIOSDBG.exe" Proper...

Options | Font | Layout | Colors

Window Preview

Screen Buffer Size
Width: 120
Height: 50

Window Size
Width: 120
Height: 50

Window Position
Left: 250
Top: 250
[✓] Let system position window

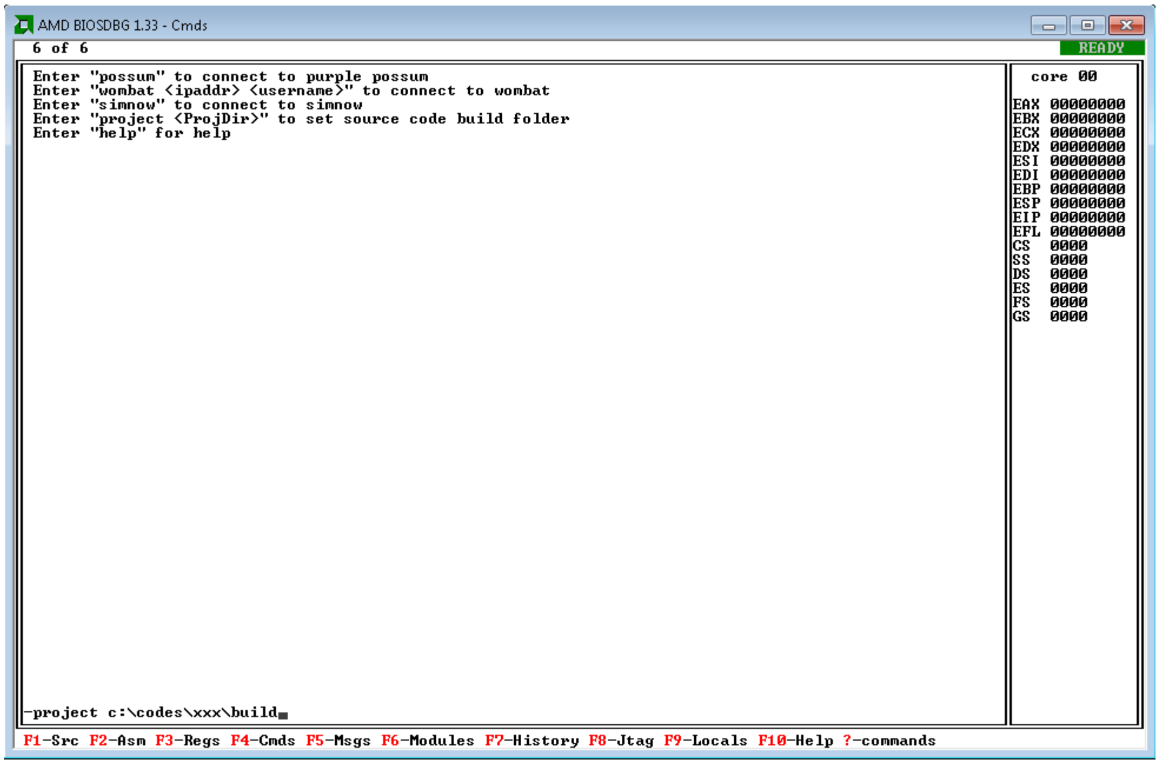
OK | Cancel

3.2 Startup for SimNow™

Start AMD BIOSDBG at any time after SimNow™ is started. Set "Project <ProjDir>" in the startup window and then enter "Simnow" to start SimNow™ debugging. AMD BIOSDBG must be run on the same machine as SimNow™. AMD BIOSDBG will find and connect to SimNow™ (AMD BIOSDBG supports only a single SimNow™ session). AMD BIOSDBG be exited and started while SimNow™ is running with no effect on the SimNow™ session. If the simulation has not started, AMD BIOSDBG executes the 'memdevice.load' command. This works around a problem where SimNow™ uses an old BIOS image even after the image has been updated.

AMD BIOSDBG locates the main BIOS image by executing the SimNow™ 'memdevice.getinitfile' command. By default, AMD BIOSDBG uses the path that SimNow™ loaded the BIOS image from as the project directory. This path will be searched when looking for map, pdb, and source files. If it is not possible to place the BIOS image in the top level of the project, then use the -project= command line option to supply the project top level directory.

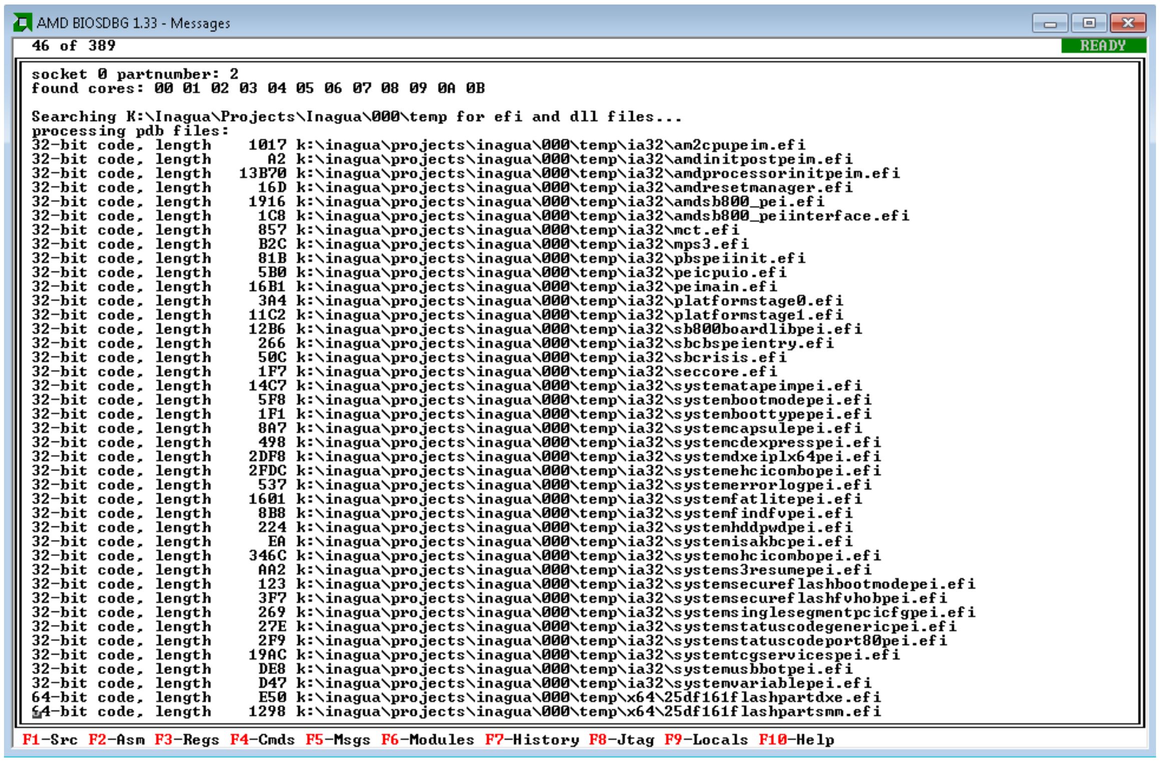
Set project folder to "c:\code\xxx\build":


AMD BIOSDBG 1.33 - Cmds
6 of 6
READY
F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F10-Help ?-commands

```
Enter "possum" to connect to purple possum
Enter "wombat <ipaddr> <username>" to connect to wombat
Enter "simnow" to connect to simnow
Enter "project <ProjDir>" to set source code build folder
Enter "help" for help

-project c:\codes\xxx\build
```

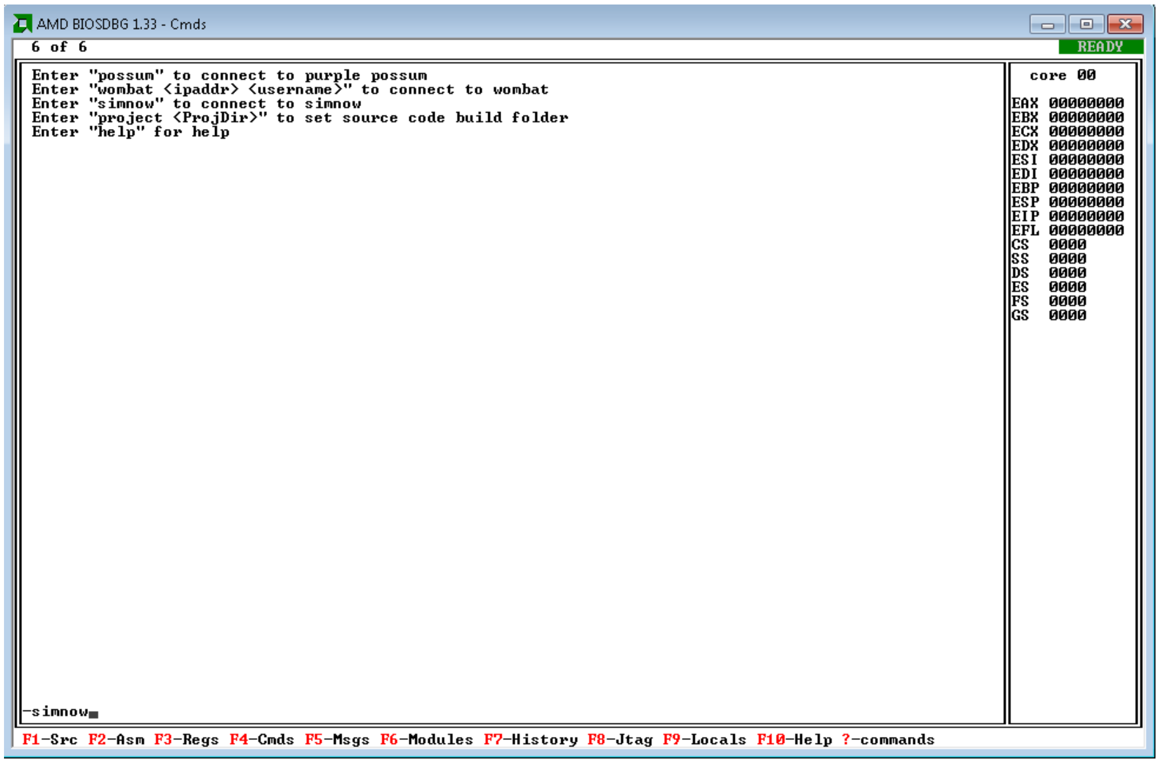
```
  core 00

EAX 00000000
EBX 00000000
ECX 00000000
EDX 00000000
ESI 00000000
EDI 00000000
EBP 00000000
ESP 00000000
EIP 00000000
EFL 00000000
CS  0000
SS  0000
DS  0000
ES  0000
FS  0000
GS  0000
```

If the project folder is set correctly, AMD BIOSDBG will search all the .efi and .dll files with relative .pdb files to get code information:



```
AMD BIOSDBG 1.33 - Messages
 46 of 389                                                                                READY

 socket 0 partnumber: 2
 found cores: 00 01 02 03 04 05 06 07 08 09 0A 0B

 Searching K:\Inagua\Projects\Inagua\000\temp for efi and dll files...
 processing pdb files:
 32-bit code, length    1017 k:\inagua\projects\inagua\000\temp\ia32\am2cpupeim.efi
 32-bit code, length      A2 k:\inagua\projects\inagua\000\temp\ia32\amdinitpostpeim.efi
 32-bit code, length   13B70 k:\inagua\projects\inagua\000\temp\ia32\amdprocessorinitpeim.efi
 32-bit code, length     16D k:\inagua\projects\inagua\000\temp\ia32\amdresetmanager.efi
 32-bit code, length    1916 k:\inagua\projects\inagua\000\temp\ia32\amdsb800_pei.efi
 32-bit code, length     1C8 k:\inagua\projects\inagua\000\temp\ia32\amdsb800_peiinterface.efi
 32-bit code, length     857 k:\inagua\projects\inagua\000\temp\ia32\mct.efi
 32-bit code, length     B2C k:\inagua\projects\inagua\000\temp\ia32\mps3.efi
 32-bit code, length     81B k:\inagua\projects\inagua\000\temp\ia32\pbspeiinit.efi
 32-bit code, length     5B0 k:\inagua\projects\inagua\000\temp\ia32\peicpuio.efi
 32-bit code, length    16B1 k:\inagua\projects\inagua\000\temp\ia32\peimain.efi
 32-bit code, length     3A4 k:\inagua\projects\inagua\000\temp\ia32\platformstage0.efi
 32-bit code, length    11C2 k:\inagua\projects\inagua\000\temp\ia32\platformstage1.efi
 32-bit code, length    12B6 k:\inagua\projects\inagua\000\temp\ia32\sb800boardlibpei.efi
 32-bit code, length     266 k:\inagua\projects\inagua\000\temp\ia32\sbcbspeientry.efi
 32-bit code, length     50C k:\inagua\projects\inagua\000\temp\ia32\sbcrisis.efi
 32-bit code, length     1F7 k:\inagua\projects\inagua\000\temp\ia32\seccore.efi
 32-bit code, length    14C7 k:\inagua\projects\inagua\000\temp\ia32\systematapeimpei.efi
 32-bit code, length     5F8 k:\inagua\projects\inagua\000\temp\ia32\systembootmodepei.efi
 32-bit code, length     1F1 k:\inagua\projects\inagua\000\temp\ia32\systemboottypepei.efi
 32-bit code, length     8A7 k:\inagua\projects\inagua\000\temp\ia32\systemcapsulepei.efi
 32-bit code, length     498 k:\inagua\projects\inagua\000\temp\ia32\systemcdexpresspei.efi
 32-bit code, length    2DF8 k:\inagua\projects\inagua\000\temp\ia32\systemdxeiplx64pei.efi
 32-bit code, length    2FDC k:\inagua\projects\inagua\000\temp\ia32\systemehcicombopei.efi
 32-bit code, length     537 k:\inagua\projects\inagua\000\temp\ia32\systemerrorlogpei.efi
 32-bit code, length    1601 k:\inagua\projects\inagua\000\temp\ia32\systemfatlitepei.efi
 32-bit code, length     8B8 k:\inagua\projects\inagua\000\temp\ia32\systemfindfvpei.efi
 32-bit code, length     224 k:\inagua\projects\inagua\000\temp\ia32\systemhddpwdpei.efi
 32-bit code, length      EA k:\inagua\projects\inagua\000\temp\ia32\systemisakbcpei.efi
 32-bit code, length    346C k:\inagua\projects\inagua\000\temp\ia32\systemohcicombopei.efi
 32-bit code, length     AA2 k:\inagua\projects\inagua\000\temp\ia32\systems3resumepei.efi
 32-bit code, length     123 k:\inagua\projects\inagua\000\temp\ia32\systemsecureflashbootmodepei.efi
 32-bit code, length     3F7 k:\inagua\projects\inagua\000\temp\ia32\systemsecureflashfvhobpei.efi
 32-bit code, length     269 k:\inagua\projects\inagua\000\temp\ia32\systemsinglesegmentpcicfgpei.efi
 32-bit code, length     27E k:\inagua\projects\inagua\000\temp\ia32\systemstatuscodegenericpei.efi
 32-bit code, length     2F9 k:\inagua\projects\inagua\000\temp\ia32\systemstatuscodeport80pei.efi
 32-bit code, length    19AC k:\inagua\projects\inagua\000\temp\ia32\systemtcgservicespei.efi
 32-bit code, length     DE8 k:\inagua\projects\inagua\000\temp\ia32\systemusbbotpei.efi
 32-bit code, length     D47 k:\inagua\projects\inagua\000\temp\ia32\systemvariablepei.efi
 64-bit code, length     E50 k:\inagua\projects\inagua\000\temp\x64\25df161flashpartdxe.efi
 64-bit code, length    1298 k:\inagua\projects\inagua\000\temp\x64\25df161flashpartsmm.efi
 F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F10-Help
```

Enter "Simnow" to start SimNow™ debugging:



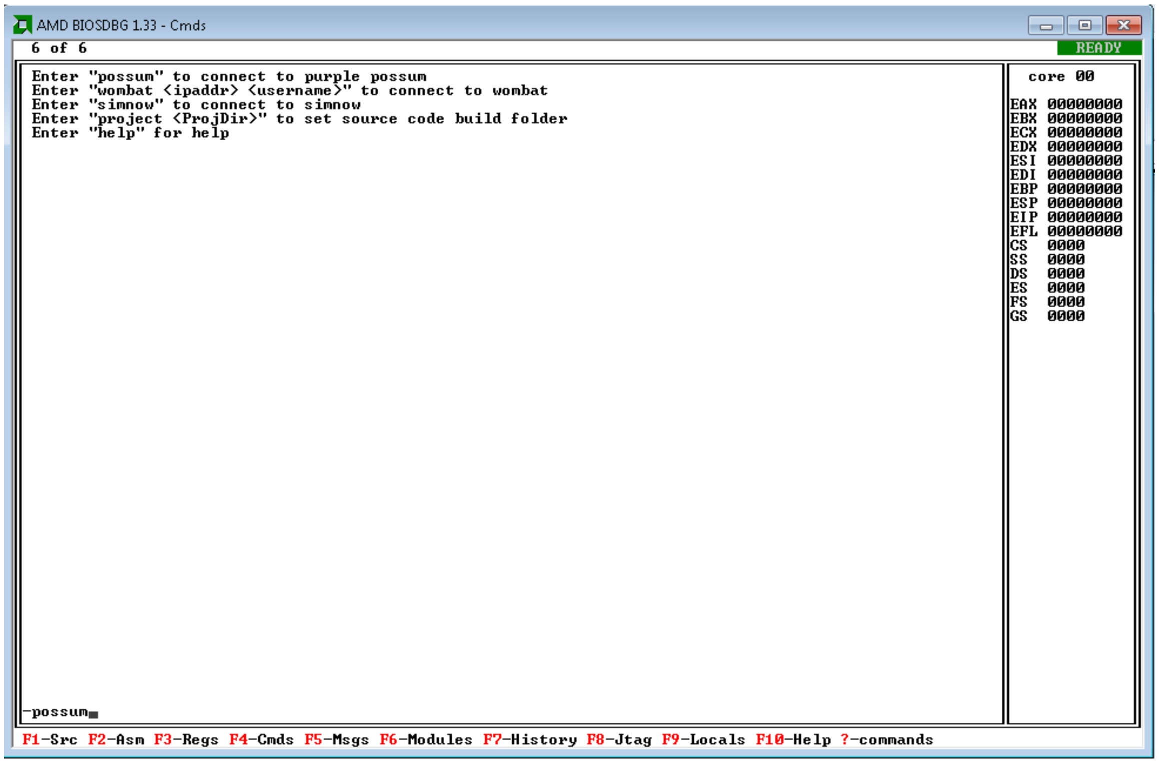
```
AMD BIOSDBG 1.33 - Cmds
 6 of 6                                                                                   READY
 Enter "possum" to connect to purple possum                                           core 00
 Enter "wombat <ipaddr> <username>" to connect to wombat
 Enter "simnow" to connect to simnow                                                 EAX 00000000
 Enter "project <ProjDir>" to set source code build folder                           EBX 00000000
 Enter "help" for help                                                               ECX 00000000
                                                                                     EDX 00000000
                                                                                     ESI 00000000
                                                                                     EDI 00000000
                                                                                     EBP 00000000
                                                                                     ESP 00000000
                                                                                     EIP 00000000
                                                                                     EFL 00000000
                                                                                     CS  0000
                                                                                     SS  0000
                                                                                     DS  0000
                                                                                     ES  0000
                                                                                     FS  0000
                                                                                     GS  0000

-simnow
 F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F10-Help ?-commands
```

3.3 Startup for Purple Possum

Start AMD BIOSDBG and set "Project <ProjDir>" in the startup window and then enter "possum". Passing the project path on the command line allows AMD BIOSDBG to locate the bios binary used as the default for the lpcload and reload commands (the default bios binary is the first *.rom file found in the top level of the project path). Passing the project path on the command line is also a requirement for source level debugging when Purple Possum is used.

If using the LPC flash emulator feature of the Purple Possum, load the bios image using the lpcload or reload command. These commands can work with no arguments if the project path was passed on the command line. With some systems, AMD BIOSDBG will report an error when started up before the Purple Possum rom emulator is initialized. This error can be ignored. The LPC flash emulator can be loaded even when the target is unavailable. After the bios image is loaded, it is often useful to execute the reset command. The reset command will stop the target at the reset vector.

If possible connect the second Purple Possum relay output to the target board cold reset input. This will prevent AMD BIOSDBG from prompting for the reset button to be pressed each time the reset command is entered.



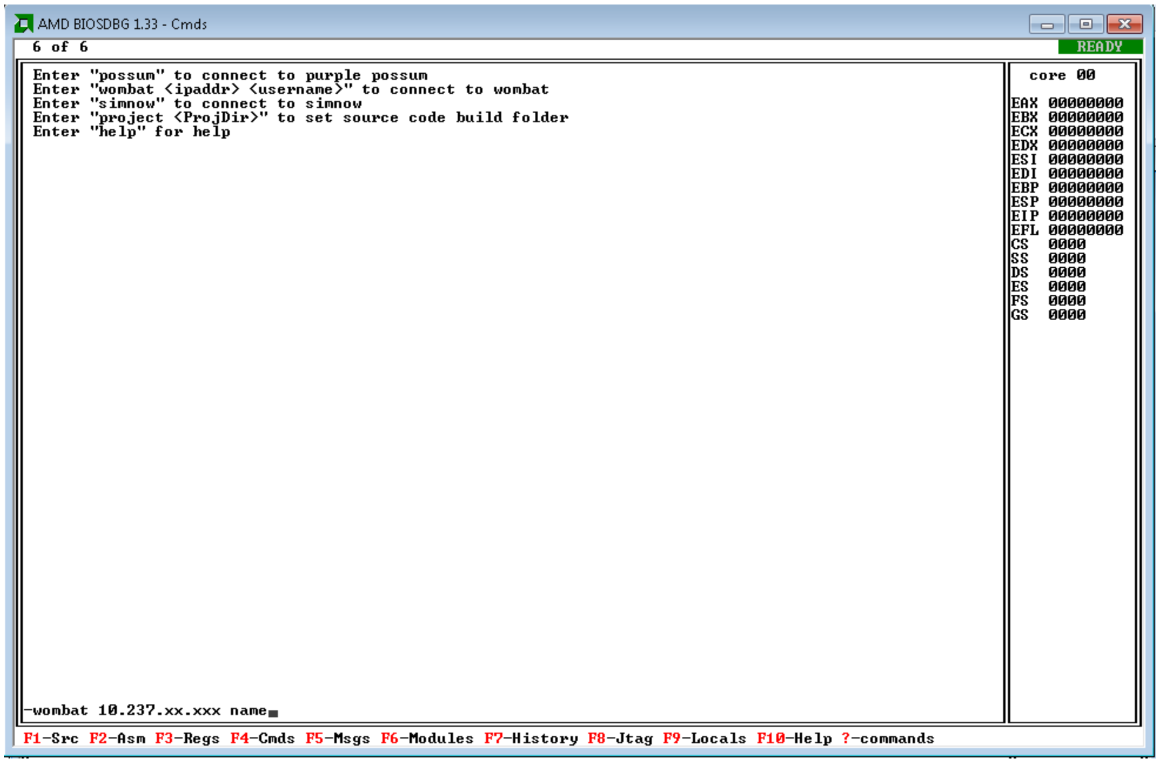
```
Enter "possum" to connect to purple possum
Enter "wombat <ipaddr> <username>" to connect to wombat
Enter "simnow" to connect to simnow
Enter "project <ProjDir>" to set source code build folder
Enter "help" for help

-possum
```

```
 core 00

EAX 00000000
EBX 00000000
ECX 00000000
EDX 00000000
ESI 00000000
EDI 00000000
EBP 00000000
ESP 00000000
EIP 00000000
EFL 00000000
CS  0000
SS  0000
DS  0000
ES  0000
FS  0000
GS  0000
```

F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F10-Help ?-commands

3.4 Startup for Wombat

Start AMD BIOSDBG, set “Project <ProjDir>” in the startup window and then enter "wombat <ipAddr> <userName>".



```
Enter "possum" to connect to purple possum
Enter "wombat <ipaddr> <username>" to connect to wombat
Enter "simnow" to connect to simnow
Enter "project <ProjDir>" to set source code build folder
Enter "help" for help

-wombat 10.237.xx.xxx name
```

```
 core 00

EAX 00000000
EBX 00000000
ECX 00000000
EDX 00000000
ESI 00000000
EDI 00000000
EBP 00000000
ESP 00000000
EIP 00000000
EFL 00000000
CS  0000
SS  0000
DS  0000
ES  0000
FS  0000
GS  0000
```

F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F10-Help ?-commands

4 Keyboard commands

Function key assignments are listed at the bottom of the window. Additional keys:

Navigation keys	
Key	Function
<Ctrl> Home, <Ctrl> End	Move to top or bottom of buffer
PgUp, PgDn	Move up or down one page in buffer
Enter	Move to rip location (code or asm view)
Enter	Open file (modules view)
Enter	View details (locals view)
Arrow keys	Move within window or buffer
Mouse wheel	Move within window or buffer
Mouse single click	Move within window
Home, end	Move to start or end of line
Tab, BackTab	Move left or right by 8 columns

Special keys	
Key	Function
F1	Switch to source code window
F2	Switch to disassembly window
F3	Switch to register window
F4	Switch to command window
F5	Switch to debug message window
F6	Switch to modules window
F7	Switch to execution history window
F8	Switch to jtag debug message window
<Shift>F8	Switch to pipe debug window
F9	Switch to locals windows
<Shift>F9	Switch to globals windows
F10	Show html help file in browser
<Ctrl>N	Set next instruction to execute (F1 or F2 view)
<Ctrl>L	Clear the screen
<Alt>X	Exit application
O	Step over (F1 or F2 view)
I	Step into (F1 or F2 view)
G	Go (start or continue simulation) (F1 or F2 view)
<Ctrl>G	Run to cursor (F1 or F2 view)
<Ctrl>C	Break into execution (F2 or F3 view)
B	Toggle SimNow™ execution breakpoint
H	Toggle hardware breakpoint (HDT only)
<Ctrl>E	Edit the source file in the F1 display
Esc	F4 command view: toggle between command entry and navigation mode F9 locals view: up one level
?	Display help text (F4 command view)

5 Breakpoints

AMD BIOSDBG retains the breakpoint settings for each project by writing them to the ini file. Use the help command (?) on the F4 command window to see breakpoint options available from a command line. Execution breakpoints are most easily set from a source code view or disassembly view. To set a breakpoint on a source code line or disassembled instruction, move the cursor to the breakpoint location and press B (SimNow™) or H (Purple Possum or Wombat). Use the same method to remove the breakpoint. If the source file visible in the F1 Source display is not the file where the breakpoint is needed, press F6 to open the Modules window. Move the cursor to the desired source file and press Enter. This will open the source file and allow breakpoints to be set. The F6 modules list may be large. Press the first letter of the file name to cycle through files that begin with that letter. If the file name does not appear in the Modules list, the module load address may not be known. This happens with compressed UEFI code that is loaded at an address determined at run time. In order for a module to be present in the F6 modules list, AMD BIOSDBG must get control when that module is executing. This may possible by using <Ctrl>C to break into execution. Other methods are single stepping into the code, stopping on a port 80 write, or stopping at a jmp $.

5.1 Breakpoints (SimNow™)

AMD BIOSDBG manages the SimNow™ execution breakpoints. The B key toggles SimNow™ execution breakpoints in the source code or disassembly window. AMD BIOSDBG does not set or clear other SimNow™ breakpoint types, such as I/O breakpoints. To set an I/O breakpoint, use the SimNow™ debugger window (SimNow™ GUI, View, Show debugger). A SimNow™ limitation is that breakpoints cannot be set on an individual core. Instead, the breakpoint will be active only for the selected core. To hit a breakpoint on an AP core, use the core command to select the desired AP core. Hardware breakpoints are not used in SimNow™ mode because SimNow™ has no breakpoint redirect feature.

5.2 Breakpoints (Purple Possum or Wombat)

The H key toggles hardware execution breakpoints in the source code or disassembly window. If all 4 hardware breakpoints are enabled for the selected core, then the H key is ignored. Use the bl (breakpoint list) command to see how many hardware breakpoints are in use. The B key sets a software breakpoint and has no function if the code memory is not writable. Note: software breakpoints are not well tested. Use hardware breakpoints if possible. Breakpoint types other than execution breakpoints must be entered from the F4 command window. Use the HDT command to see the available breakpoint commands.

A Appendix

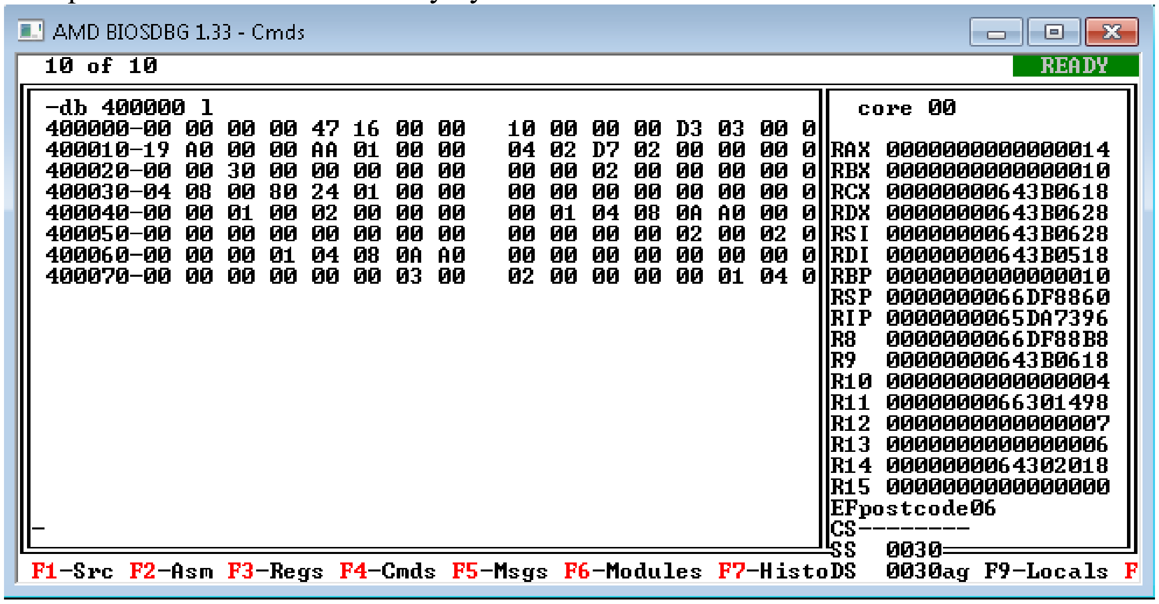
A.1 Commands in F4 View

A.1.1 Dump memory

Usage:
d[b|w|d|q] [range] [l]

Function:
Dump memory, b|w|d|q defaults to last used, l=linear

Example:
Dump linear address 0x400000 by byte:



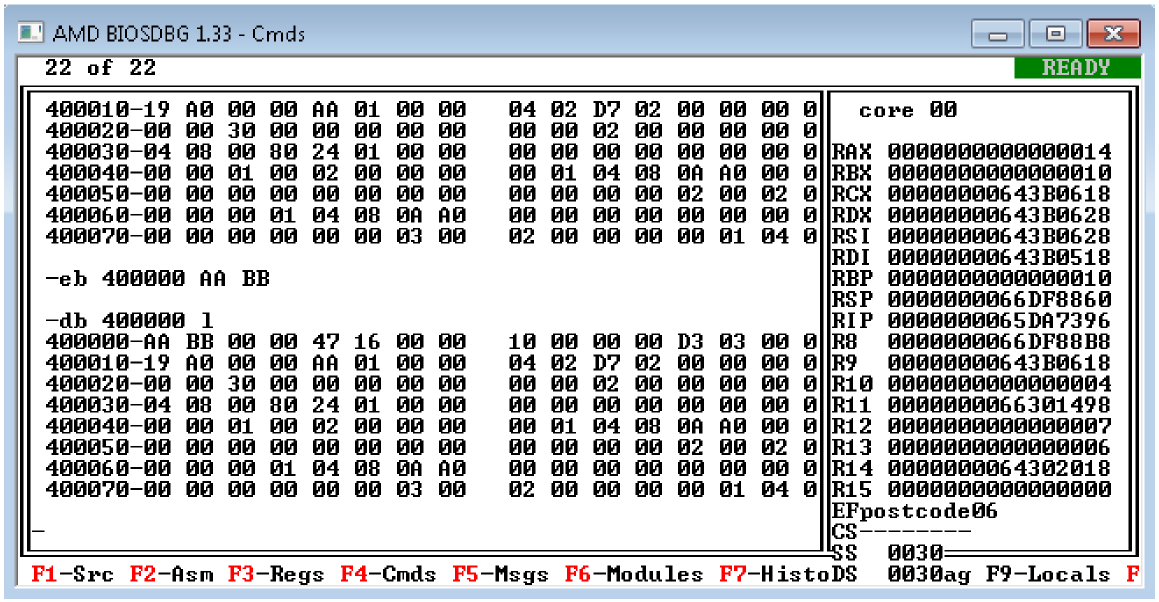
```
AMD BIOSDBG 1.33 - Cmds
  10 of 10                                                          READY
 -db 400000 l                                              core 00
 400000-00 00 00 00 47 16 00 00   10 00 00 00 D3 03 00 0
 400010-19 A0 00 00 AA 01 00 00   04 02 D7 02 00 00 00 0 RAX 0000000000000014
 400020-00 00 30 00 00 00 00 00   00 00 02 00 00 00 00 0 RBX 0000000000000010
 400030-04 08 00 80 24 01 00 00   00 00 00 00 00 00 00 0 RCX 00000000643B0618
 400040-00 00 01 00 02 00 00 00   00 01 04 08 0A A0 00 0 RDX 00000000643B0628
 400050-00 00 00 00 00 00 00 00   00 00 00 00 02 00 02 0 RSI 00000000643B0628
 400060-00 00 00 01 04 08 0A A0   00 00 00 00 00 00 00 0 RDI 00000000643B0518
 400070-00 00 00 00 00 00 03 00   02 00 00 00 00 01 04 0 RBP 0000000000000010
                                                         RSP 0000000066DF8860
                                                         RIP 0000000065DA7396
                                                         R8  0000000066DF88B8
                                                         R9  00000000643B0618
                                                         R10 0000000000000004
                                                         R11 0000000066301498
                                                         R12 0000000000000007
                                                         R13 0000000000000006
                                                         R14 0000000064302018
                                                         R15 0000000000000000
                                                         EFpostcode06
_                                                        CS---------
                                                         SS  0030
F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-HistoDS  0030ag F9-Locals F
```

A.1.2 Modify memory

Usage:
e[b|w|d|q] address x x x...

Function:
Enter one or more values into memory start with address

Example:
Modify address 0x400000 to 0xAA, and modify address 0x400001 to 0xBB:



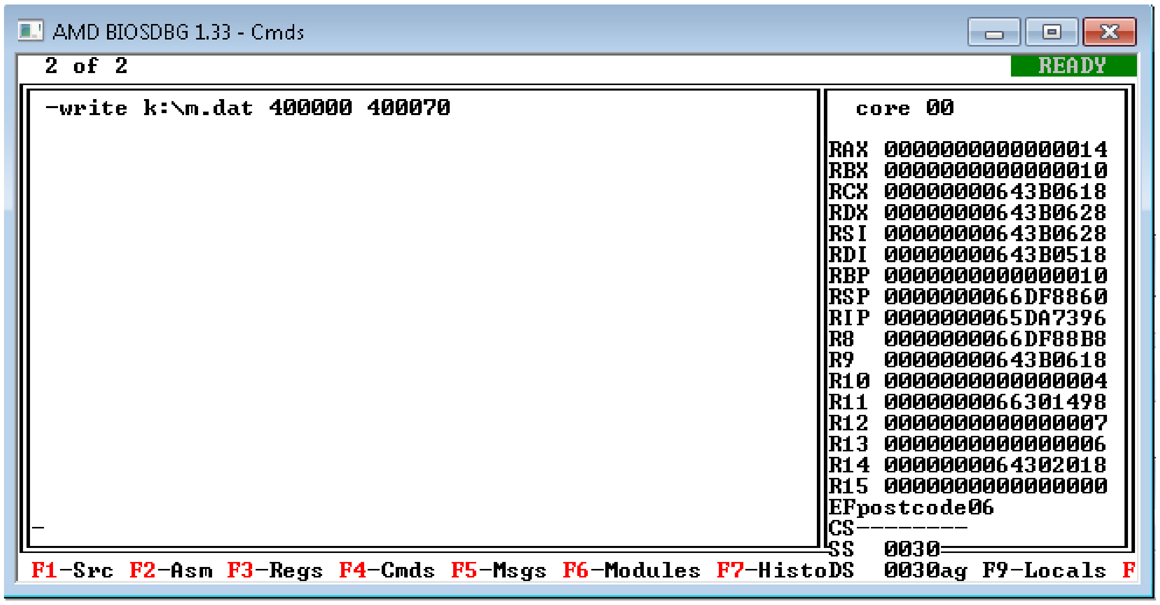
```
AMD BIOSDBG 1.33 - Cmds
 22 of 22                                                              READY
 400010-19 A0 00 00 AA 01 00 00   04 02 D7 02 00 00 00 0   core 00
 400020-00 00 30 00 00 00 00 00   00 00 02 00 00 00 00 0
 400030-04 08 00 80 24 01 00 00   00 00 00 00 00 00 00 0 RAX 0000000000000014
 400040-00 00 01 00 02 00 00 00   00 01 04 08 0A A0 00 0 RBX 0000000000000010
 400050-00 00 00 00 00 00 00 00   00 00 00 00 02 00 02 0 RCX 00000000643B0618
 400060-00 00 00 01 04 08 0A A0   00 00 00 00 00 00 00 0 RDX 00000000643B0628
 400070-00 00 00 00 00 00 03 00   02 00 00 00 00 01 04 0 RSI 00000000643B0628
                                                         RDI 00000000643B0518
 -eb 400000 AA BB                                        RBP 0000000000000010
                                                         RSP 0000000066DF8860
 -db 400000 l                                            RIP 0000000065DA7396
 400000-AA BB 00 00 47 16 00 00   10 00 00 00 D3 03 00 0 R8  0000000066DF88B8
 400010-19 A0 00 00 AA 01 00 00   04 02 D7 02 00 00 00 0 R9  00000000643B0618
 400020-00 00 30 00 00 00 00 00   00 00 02 00 00 00 00 0 R10 0000000000000004
 400030-04 08 00 80 24 01 00 00   00 00 00 00 00 00 00 0 R11 0000000066301498
 400040-00 00 01 00 02 00 00 00   00 01 04 08 0A A0 00 0 R12 0000000000000007
 400050-00 00 00 00 00 00 00 00   00 00 00 00 02 00 02 0 R13 0000000000000006
 400060-00 00 00 01 04 08 0A A0   00 00 00 00 00 00 00 0 R14 0000000064302018
 400070-00 00 00 00 00 00 03 00   02 00 00 00 00 01 04 0 R15 0000000000000000
                                                         EFpostcode06
_                                                        CS---------
                                                         SS  0030
 F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-HistoDS  0030ag F9-Locals F
```

A.1.3 Write memory context to file

Usage:
write <filename> startAddr endAddr

Function:
Dump memory from "startAddr" to "endAddr" to file "filename"

Example:
Dump memory 0x400000 – 0x400070 to k:\m.dat:



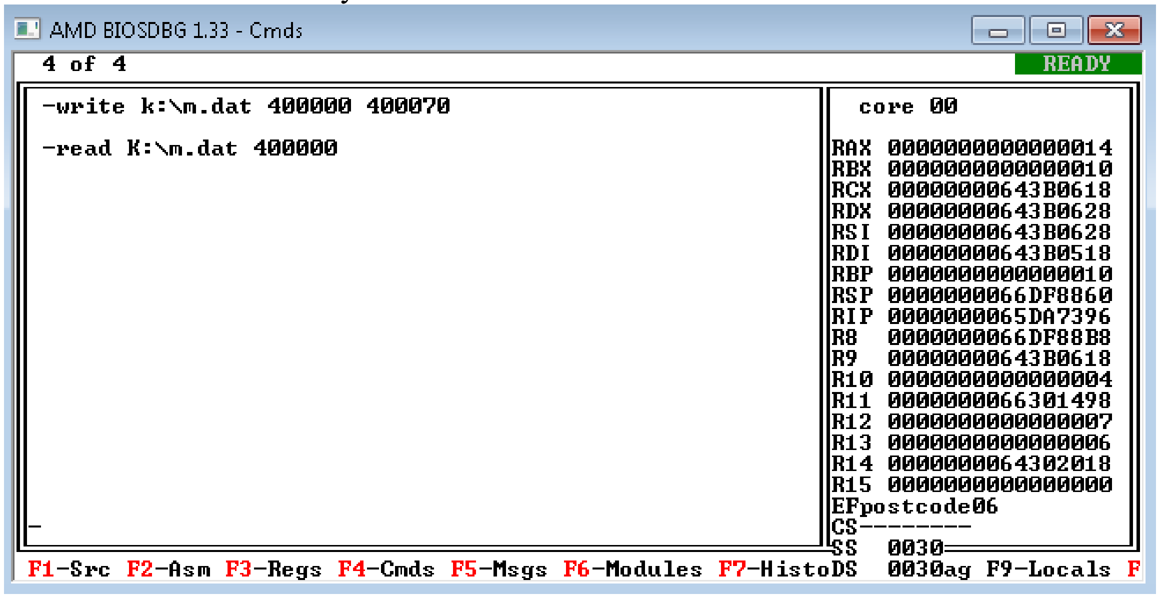
```
AMD BIOSDBG 1.33 - Cmds
 2 of 2                                                                READY
 -write k:\m.dat 400000 400070                             core 00

                                                         RAX 0000000000000014
                                                         RBX 0000000000000010
                                                         RCX 00000000643B0618
                                                         RDX 00000000643B0628
                                                         RSI 00000000643B0628
                                                         RDI 00000000643B0518
                                                         RBP 0000000000000010
                                                         RSP 0000000066DF8860
                                                         RIP 0000000065DA7396
                                                         R8  0000000066DF88B8
                                                         R9  00000000643B0618
                                                         R10 0000000000000004
                                                         R11 0000000066301498
                                                         R12 0000000000000007
                                                         R13 0000000000000006
                                                         R14 0000000064302018
                                                         R15 0000000000000000
                                                         EFpostcode06
_                                                        CS---------
                                                         SS  0030
 F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-HistoDS  0030ag F9-Locals F
```

A.1.4 Read memory context from file

Usage:
read <filename> address

Function:
Read context from <filename> into memory start from "address"

Example:
Read k:\m.dat into memory start from 0x400000:



```
 4 of 4

 -write k:\m.dat 400000 400070

 -read K:\m.dat 400000
```

```
   core 00

RAX 0000000000000014
RBX 0000000000000010
RCX 00000000643B0618
RDX 00000000643B0628
RSI 00000000643B0628
RDI 00000000643B0518
RBP 0000000000000010
RSP 0000000066DF8860
RIP 0000000065DA7396
R8  0000000066DF88B8
R9  00000000643B0618
R10 0000000000000004
R11 0000000066301498
R12 0000000000000007
R13 0000000000000006
R14 0000000064302018
R15 0000000000000000
EFpostcode06
CS--------
SS  0030
```

```
 F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-HistoDS  0030ag F9-Locals F
```

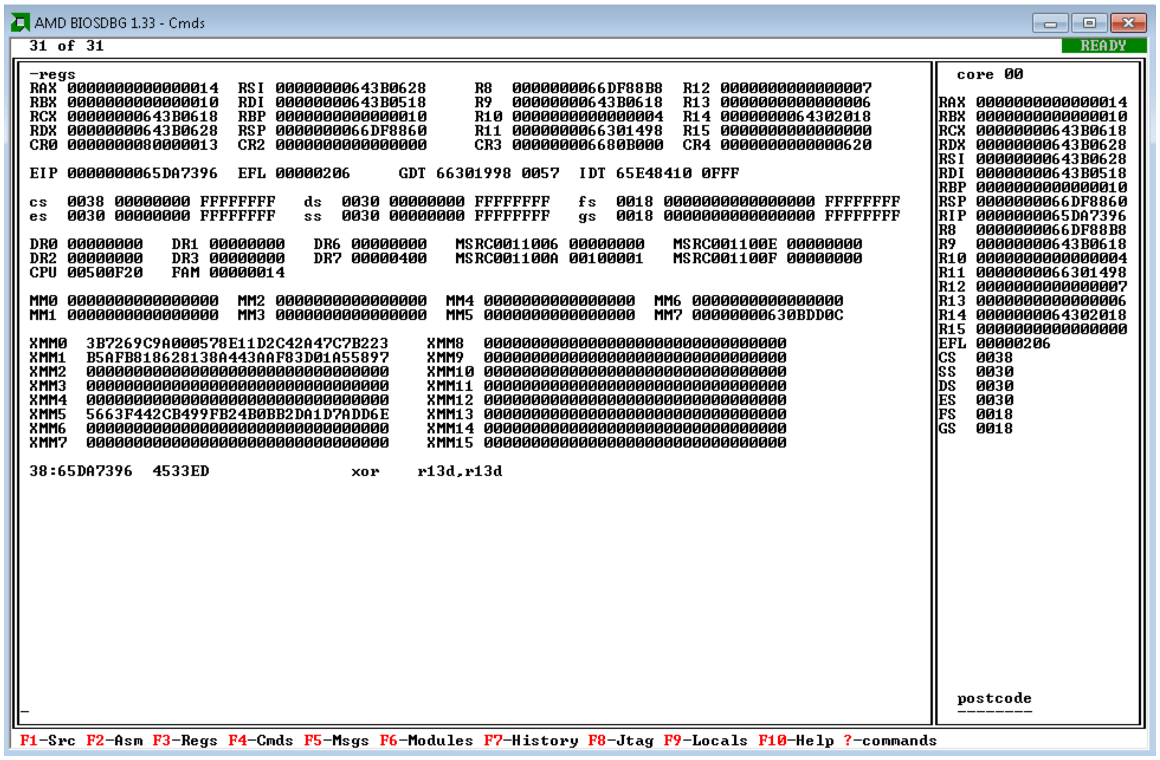
A.1.5 Dump or modify general purpose registers

A.1.5.1 Dump general purpose registers

Usage:
regs

Function:
Dump general purpose registers

Example:
Dump general purpose registers:


AMD BIOSDBG 1.33 - Cmds
31 of 31
READY
F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F10-Help ?-commands

```
-regs
RAX 0000000000000014  RSI 00000000643B0628     R8  0000000066DF88B8  R12 0000000000000007
RBX 0000000000000010  RDI 00000000643B0518     R9  00000000643B0618  R13 0000000000000006
RCX 00000000643B0618  RBP 0000000000000010     R10 0000000000000004  R14 0000000064302018
RDX 00000000643B0628  RSP 0000000066DF8860     R11 0000000066301498  R15 0000000000000000
CR0 0000000080000013  CR2 0000000000000000     CR3 000000006680B000  CR4 0000000000000620

EIP 0000000065DA7396  EFL 00000206      GDT 66301998 0057  IDT 65E48410 0FFF

cs  0038 00000000 FFFFFFFF   ds  0030 00000000 FFFFFFFF   fs  0018 0000000000000000 FFFFFFFF
es  0030 00000000 FFFFFFFF   ss  0030 00000000 FFFFFFFF   gs  0018 0000000000000000 FFFFFFFF

DR0 00000000   DR1 00000000   DR6 00000000   MSRC0011006 00000000   MSRC001100E 00000000
DR2 00000000   DR3 00000000   DR7 00000400   MSRC001100A 00100001   MSRC001100F 00000000
CPU 00500F20   FAM 00000014

MM0 0000000000000000  MM2 0000000000000000  MM4 0000000000000000  MM6 0000000000000000
MM1 0000000000000000  MM3 0000000000000000  MM5 0000000000000000  MM7 00000000630BDD0C

XMM0  3B7269C9A000578E11D2C42A47C7B223    XMM8  00000000000000000000000000000000
XMM1  B5AFB818628138A443AAF83D01A55897    XMM9  00000000000000000000000000000000
XMM2  00000000000000000000000000000000    XMM10 00000000000000000000000000000000
XMM3  00000000000000000000000000000000    XMM11 00000000000000000000000000000000
XMM4  00000000000000000000000000000000    XMM12 00000000000000000000000000000000
XMM5  5663F442CB499FB24B0BB2DA1D7ADD6E    XMM13 00000000000000000000000000000000
XMM6  00000000000000000000000000000000    XMM14 00000000000000000000000000000000
XMM7  00000000000000000000000000000000    XMM15 00000000000000000000000000000000

38:65DA7396  4533ED              xor    r13d,r13d

_
```

```
  core 00

RAX 0000000000000014
RBX 0000000000000010
RCX 00000000643B0618
RDX 00000000643B0628
RSI 00000000643B0628
RDI 00000000643B0518
RBP 0000000000000010
RSP 0000000066DF8860
RIP 0000000065DA7396
R8  0000000066DF88B8
R9  00000000643B0618
R10 0000000000000004
R11 0000000066301498
R12 0000000000000007
R13 0000000000000006
R14 0000000064302018
R15 0000000000000000
EFL 00000206
CS  0038
SS  0030
DS  0030
ES  0030
FS  0018
GS  0018

  postcode
  --------
```

A.1.5.2 Modify general purpose registers

Usage:
rax x

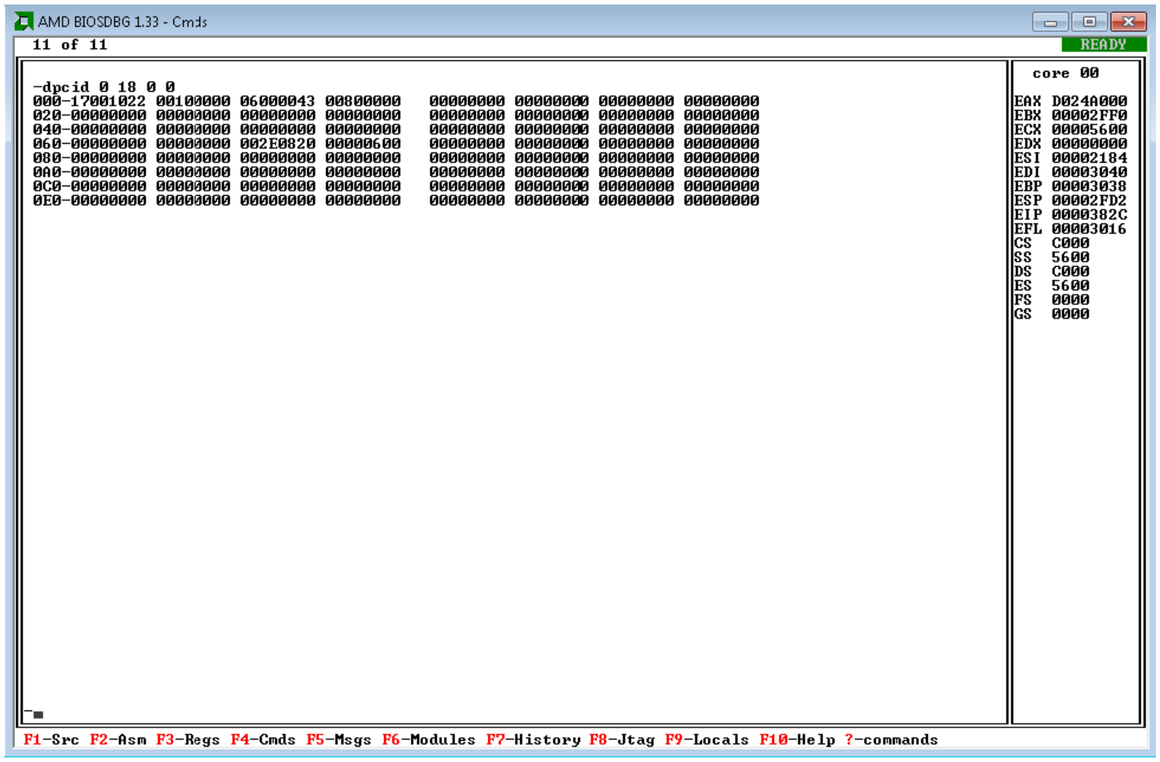
Function:
Modify RAX to 'x', other registers are similar to change values.

A.1.6 Dump PCI registers

Usage:
dpci[b|w|d] b d f [range]
or
dpci[b|w|d]

Function:
Dump PCI register for bus, device, function offset
or
Dump PCI registers space for all devices found

Example:
Dump all PCI registers of device 18, function 0:



```
11 of 11

-dpcid 0 18 0 0
000-17001022 00100000 06000043 00800000   00000000 00000000 00000000 00000000
020-00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
040-00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
060-00000000 00000000 002E0820 00000600   00000000 00000000 00000000 00000000
080-00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
0A0-00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
0C0-00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
0E0-00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000

-
```

```
  core 00

EAX D024A000
EBX 00002FF0
ECX 00005600
EDX 00000000
ESI 00002184
EDI 00003040
EBP 00003038
ESP 00002FD2
EIP 0000382C
EFL 00003016
CS  C000
SS  5600
DS  C000
ES  5600
FS  0000
GS  0000
```

F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F10-Help ?-commands

A.1.7 Modify PCI registers

Usage:

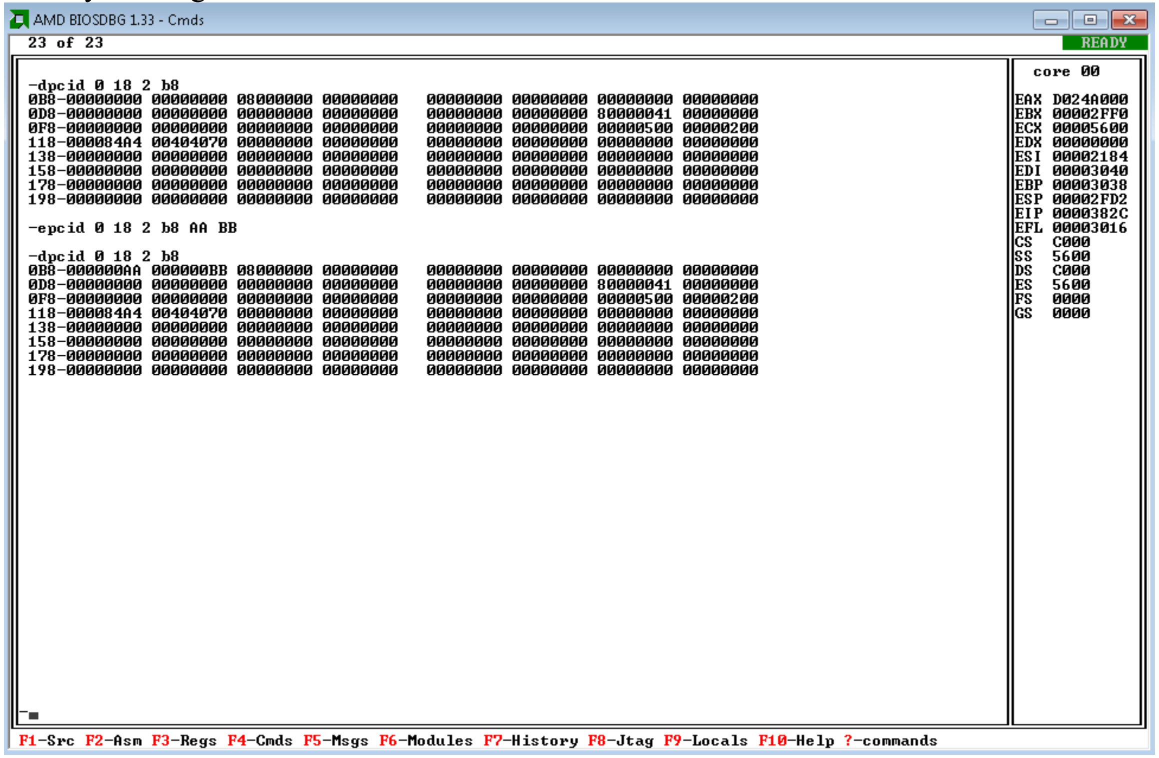
epci[b|w|d] b d f o x x x...

Function:

Modify PCI register for bus, device, function offset to value

Example:

Modify PCI register D18F2XB8to 0xAA and D18F2XBC to 0xBB:



```
-dpcid 0 18 2 b8
0B8-00000000 00000000 08000000 00000000   00000000 00000000 00000000 00000000
0D8-00000000 00000000 00000000 00000000   00000000 00000000 80000041 00000000
0F8-00000000 00000000 00000000 00000000   00000000 00000000 00000500 00000200
118-000084A4 00404070 00000000 00000000   00000000 00000000 00000000 00000000
138-00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
158-00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
178-00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
198-00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000

-epcid 0 18 2 b8 AA BB

-dpcid 0 18 2 b8
0B8-000000AA 000000BB 08000000 00000000   00000000 00000000 00000000 00000000
0D8-00000000 00000000 00000000 00000000   00000000 00000000 80000041 00000000
0F8-00000000 00000000 00000000 00000000   00000000 00000000 00000500 00000200
118-000084A4 00404070 00000000 00000000   00000000 00000000 00000000 00000000
138-00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
158-00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
178-00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000
198-00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000

-
```

```
  core 00

EAX D024A000
EBX 00002FF0
ECX 00005600
EDX 00000000
ESI 00002184
EDI 00003040
EBP 00003038
ESP 00002FD2
EIP 0000382C
EFL 00003016
CS  C000
SS  5600
DS  C000
ES  5600
FS  0000
GS  0000
```

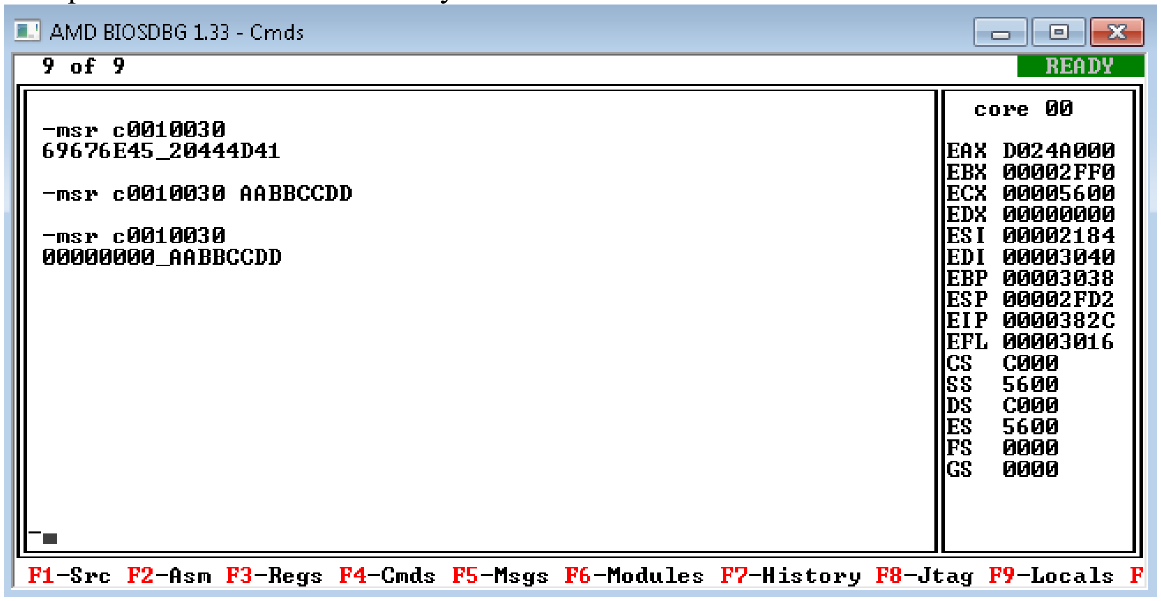
F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F10-Help ?-commands

A.1.8 Dump or modify MSR registers

Usage:
msr x [new value]

Function:
Dump or modify MSR x

Example:
Dump MSR C0010030 and modify it to 0xAABBCCDD:


AMD BIOSDBG 1.33 - Cmds
9 of 9
READY
F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F

```
-msr c0010030
69676E45_20444D41

-msr c0010030 AABBCCDD

-msr c0010030
00000000_AABBCCDD
```

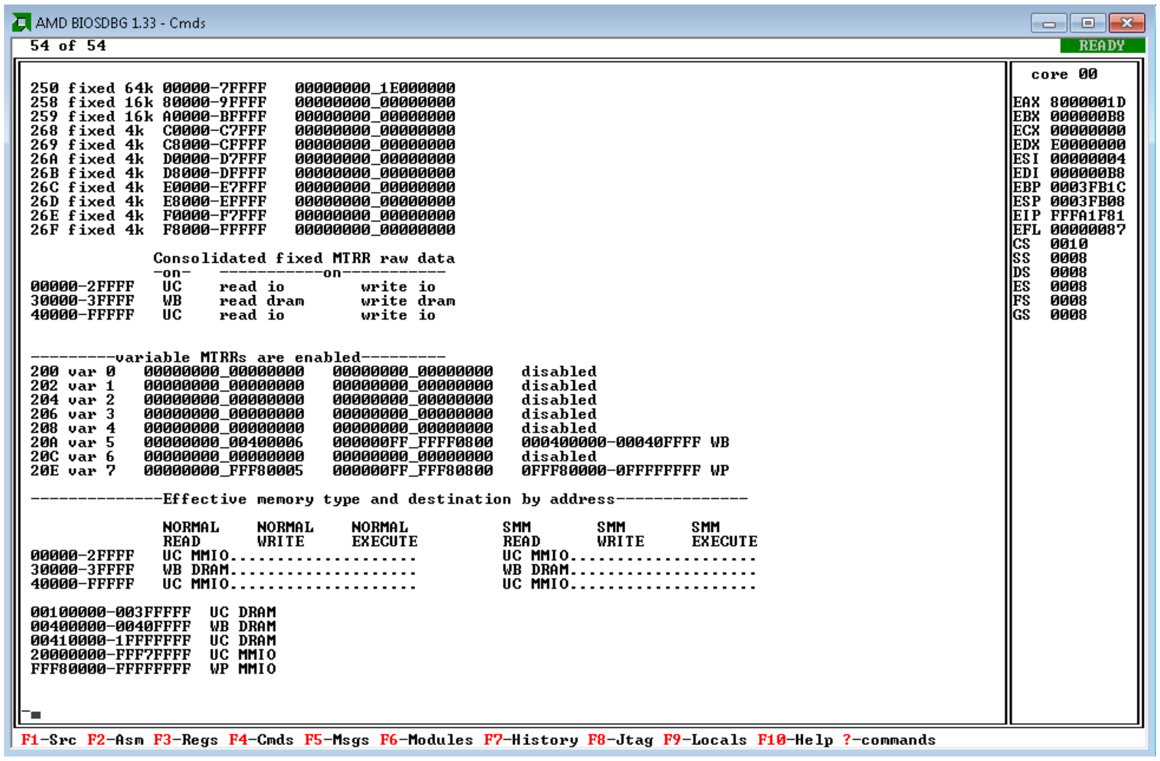
```
 core 00

EAX D024A000
EBX 00002FF0
ECX 00005600
EDX 00000000
ESI 00002184
EDI 00003040
EBP 00003038
ESP 00002FD2
EIP 0000382C
EFL 00003016
CS  C000
SS  5600
DS  C000
ES  5600
FS  0000
GS  0000
```

A.1.9 Dump MTRR registers

Usage:
mtrr

Function:
Dump MTRR registers and resulting memory attributes (lower 4GB)

Example:
Dump MTRR registers:



```
 54 of 54

 250 fixed 64k 00000-7FFFF   00000000_1E000000
 258 fixed 16k 80000-9FFFF   00000000_00000000
 259 fixed 16k A0000-BFFFF   00000000_00000000
 268 fixed 4k  C0000-C7FFF   00000000_00000000
 269 fixed 4k  C8000-CFFFF   00000000_00000000
 26A fixed 4k  D0000-D7FFF   00000000_00000000
 26B fixed 4k  D8000-DFFFF   00000000_00000000
 26C fixed 4k  E0000-E7FFF   00000000_00000000
 26D fixed 4k  E8000-EFFFF   00000000_00000000
 26E fixed 4k  F0000-F7FFF   00000000_00000000
 26F fixed 4k  F8000-FFFFF   00000000_00000000

              Consolidated fixed MTRR raw data
              -on-   -----------on-----------
 00000-2FFFF   UC     read io         write io
 30000-3FFFF   WB     read dram       write dram
 40000-FFFFF   UC     read io         write io


 ---------variable MTRRs are enabled---------
 200 var 0   00000000_00000000   00000000_00000000   disabled
 202 var 1   00000000_00000000   00000000_00000000   disabled
 204 var 2   00000000_00000000   00000000_00000000   disabled
 206 var 3   00000000_00000000   00000000_00000000   disabled
 208 var 4   00000000_00000000   00000000_00000000   disabled
 20A var 5   00000000_00400006   000000FF_FFFF0800   000400000-00040FFFF WB
 20C var 6   00000000_00000000   00000000_00000000   disabled
 20E var 7   00000000_FFF80005   000000FF_FFF80800   0FFF80000-0FFFFFFFF WP

 --------------Effective memory type and destination by address--------------

              NORMAL    NORMAL    NORMAL          SMM       SMM       SMM
              READ      WRITE     EXECUTE         READ      WRITE     EXECUTE
 00000-2FFFF   UC MMIO....................         UC MMIO....................
 30000-3FFFF   WB DRAM....................         WB DRAM....................
 40000-FFFFF   UC MMIO....................         UC MMIO....................

 00100000-003FFFFF  UC DRAM
 00400000-0040FFFF  WB DRAM
 00410000-1FFFFFFF  UC DRAM
 20000000-FFF7FFFF  UC MMIO
 FFF80000-FFFFFFFF  WP MMIO


-■
 F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F10-Help ?-commands
```

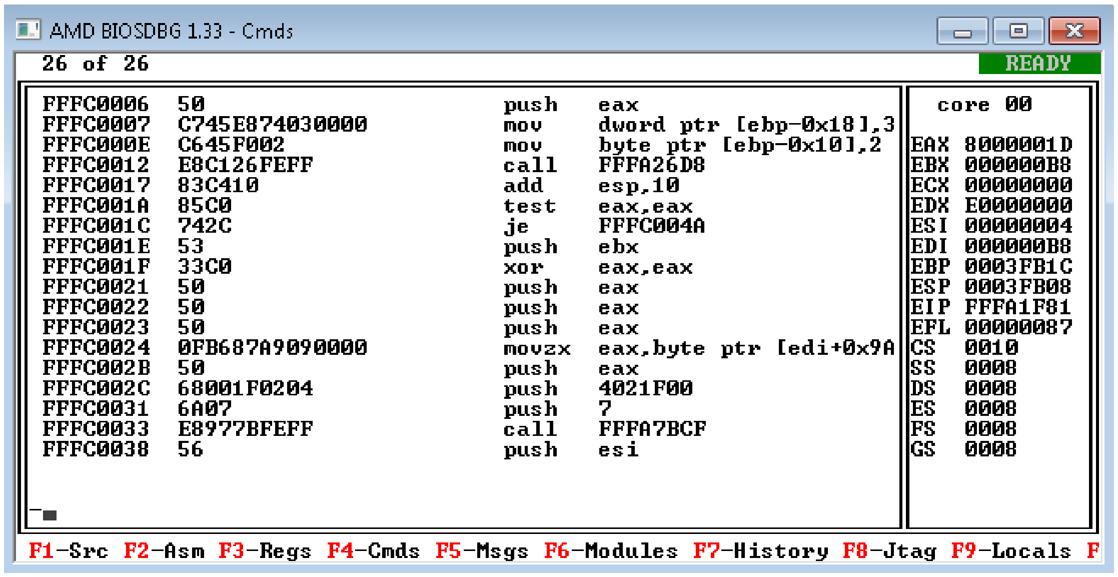
```
   core 00

EAX 8000001D
EBX 000000B8
ECX 00000000
EDX E0000000
ESI 00000004
EDI 000000B8
EBP 0003FB1C
ESP 0003FB08
EIP FFFA1F81
EFL 00000087
CS  0010
SS  0008
DS  0008
ES  0008
FS  0008
GS  0008
```

A.1.10 Unassemble memory

Usage:
u[16|32|64] [range]

Function:
Unassemble memory, 16|32|64 defaults to last used

Example:
Unassemble memory 0xFFFC0000 in 16 bits mode:



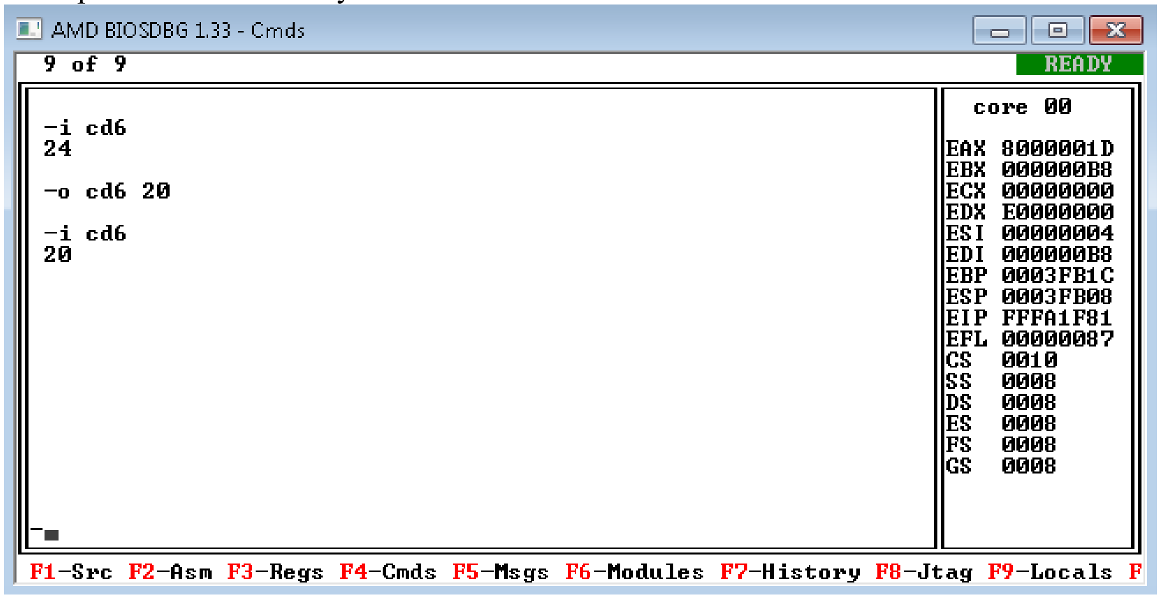
```
 26 of 26                                                    READY

 FFFC0006  50                      push   eax                      core 00
 FFFC0007  C745E874030000          mov    dword ptr [ebp-0x18],3
 FFFC000E  C645F002                mov    byte ptr [ebp-0x10],2   EAX 8000001D
 FFFC0012  E8C126FEFF              call   FFFA26D8                EBX 000000B8
 FFFC0017  83C410                  add    esp,10                  ECX 00000000
 FFFC001A  85C0                    test   eax,eax                 EDX E0000000
 FFFC001C  742C                    je     FFFC004A                ESI 00000004
 FFFC001E  53                      push   ebx                     EDI 000000B8
 FFFC001F  33C0                    xor    eax,eax                 EBP 0003FB1C
 FFFC0021  50                      push   eax                     ESP 0003FB08
 FFFC0022  50                      push   eax                     EIP FFFA1F81
 FFFC0023  50                      push   eax                     EFL 00000087
 FFFC0024  0FB687A9090000          movzx  eax,byte ptr [edi+0x9A  CS  0010
 FFFC002B  50                      push   eax                     SS  0008
 FFFC002C  68001F0204              push   4021F00                 DS  0008
 FFFC0031  6A07                    push   7                       ES  0008
 FFFC0033  E8977BFEFF              call   FFFA7BCF                FS  0008
 FFFC0038  56                      push   esi                     GS  0008

 -

 F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F
```

A.1.11 Read or write I/O port

Usage:
i[b|w|d] port
or
o[b|w|d] port value

Function:
Input from I/O port, b|w|d defaults to last used
or
Output to I/O port, b|w|d defaults to last used

Example:
Read port CD6 and modify it to 0x20:


AMD BIOSDBG 1.33 - Cmds
READY
F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F

```
 9 of 9

 -i cd6
 24

 -o cd6 20

 -i cd6
 20

-
```

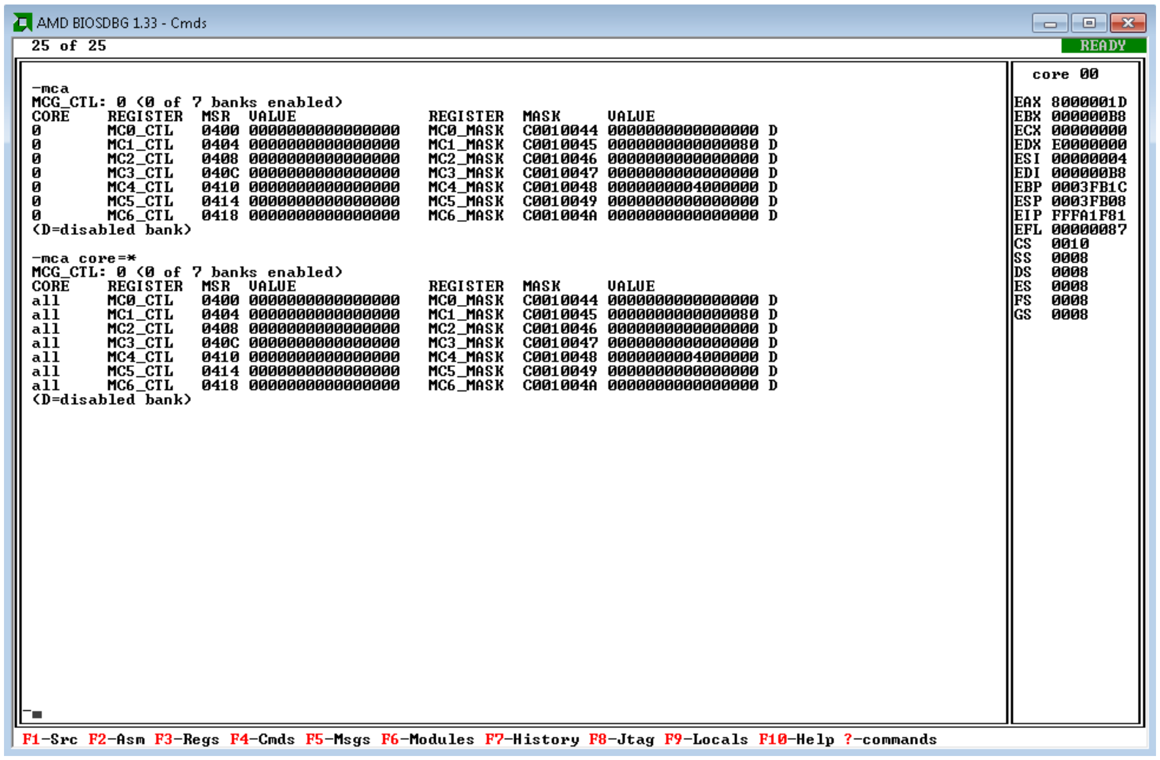
```
  core 00

EAX 8000001D
EBX 000000B8
ECX 00000000
EDX E0000000
ESI 00000004
EDI 000000B8
EBP 0003FB1C
ESP 0003FB08
EIP FFFA1F81
EFL 00000087
CS  0010
SS  0008
DS  0008
ES  0008
FS  0008
GS  0008
```

A.1.12 Read MCA errors

Usage:
mca [core=x | core=*]

Function:
Show MCA errors for current core, core x, or all cores

Example:
Show MCA errors for current core and all cores



```
25 of 25

-mca
MCG_CTL: 0 (0 of 7 banks enabled)
CORE     REGISTER  MSR  VALUE                 REGISTER  MASK     VALUE
0        MC0_CTL   0400 0000000000000000      MC0_MASK  C0010044 0000000000000000 D
0        MC1_CTL   0404 0000000000000000      MC1_MASK  C0010045 0000000000000080 D
0        MC2_CTL   0408 0000000000000000      MC2_MASK  C0010046 0000000000000000 D
0        MC3_CTL   040C 0000000000000000      MC3_MASK  C0010047 0000000000000000 D
0        MC4_CTL   0410 0000000000000000      MC4_MASK  C0010048 0000000004000000 D
0        MC5_CTL   0414 0000000000000000      MC5_MASK  C0010049 0000000000000000 D
0        MC6_CTL   0418 0000000000000000      MC6_MASK  C001004A 0000000000000000 D
(D=disabled bank)

-mca core=*
MCG_CTL: 0 (0 of 7 banks enabled)
CORE     REGISTER  MSR  VALUE                 REGISTER  MASK     VALUE
all      MC0_CTL   0400 0000000000000000      MC0_MASK  C0010044 0000000000000000 D
all      MC1_CTL   0404 0000000000000000      MC1_MASK  C0010045 0000000000000080 D
all      MC2_CTL   0408 0000000000000000      MC2_MASK  C0010046 0000000000000000 D
all      MC3_CTL   040C 0000000000000000      MC3_MASK  C0010047 0000000000000000 D
all      MC4_CTL   0410 0000000000000000      MC4_MASK  C0010048 0000000004000000 D
all      MC5_CTL   0414 0000000000000000      MC5_MASK  C0010049 0000000000000000 D
all      MC6_CTL   0418 0000000000000000      MC6_MASK  C001004A 0000000000000000 D
(D=disabled bank)

-
```

```
  core 00

EAX 8000001D
EBX 000000B8
ECX 00000000
EDX E0000000
ESI 00000004
EDI 000000B8
EBP 0003FB1C
ESP 0003FB08
EIP FFFA1F81
EFL 00000087
CS  0010
SS  0008
DS  0008
ES  0008
FS  0008
GS  0008
```

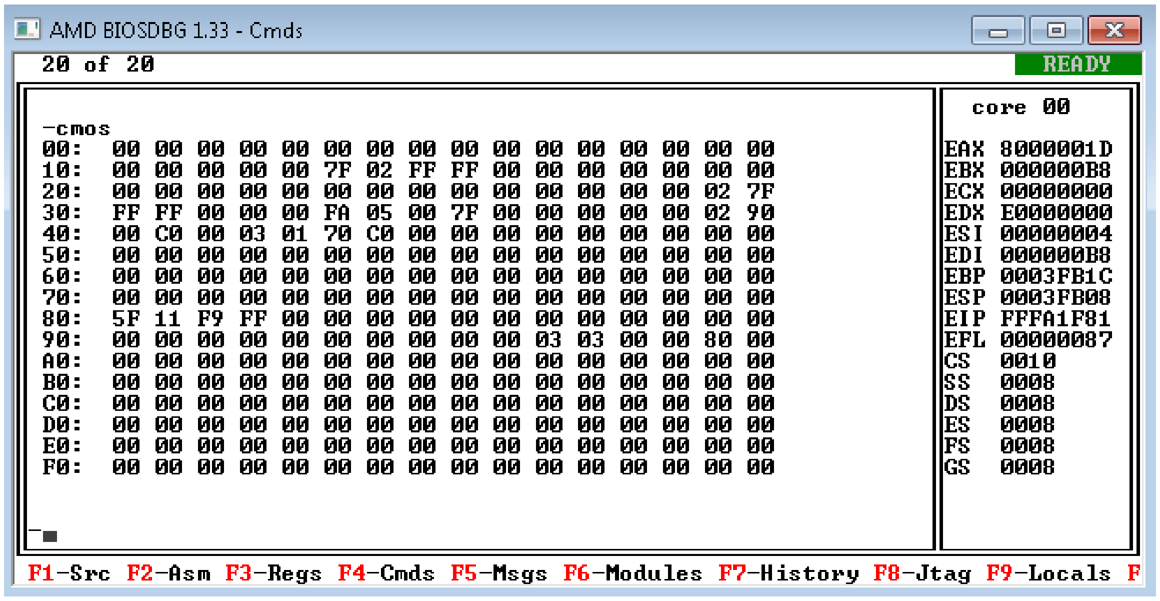
```
F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F10-Help ?-commands
```

A.1.13 Read CMOS values

Usage:
cmos

Function:
Read CMOS values

Example:
Read CMOS values:


AMD BIOSDBG 1.33 - Cmds
READY
F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F

```
 20 of 20

-cmos
00:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10:  00 00 00 00 00 7F 02 FF FF 00 00 00 00 00 00 00
20:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 7F
30:  FF FF 00 00 00 FA 05 00 7F 00 00 00 00 00 02 90
40:  00 C0 00 03 01 70 C0 00 00 00 00 00 00 00 00 00
50:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
60:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
70:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
80:  5F 11 F9 FF 00 00 00 00 00 00 00 00 00 00 00 00
90:  00 00 00 00 00 00 00 00 00 00 03 03 00 00 80 00
A0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
B0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
E0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
F0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

-
```

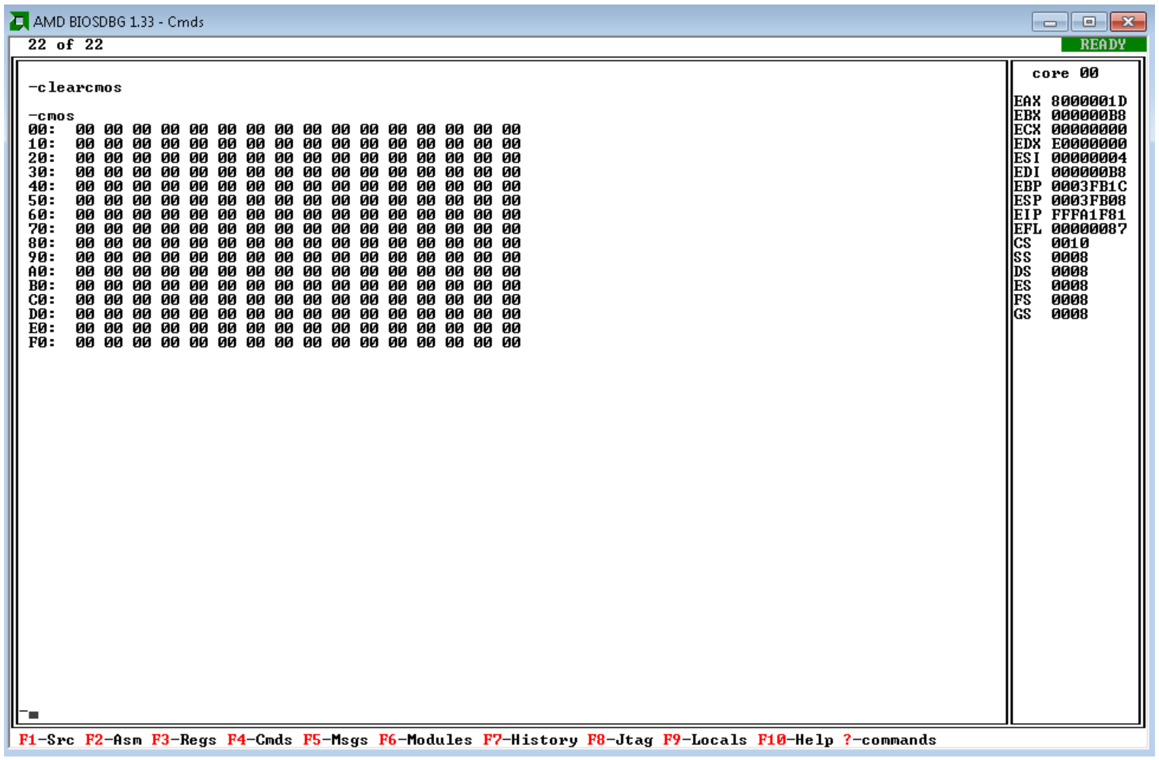
```
  core 00

EAX 8000001D
EBX 000000B8
ECX 00000000
EDX E0000000
ESI 00000004
EDI 000000B8
EBP 0003FB1C
ESP 0003FB08
EIP FFFA1F81
EFL 00000087
CS  0010
SS  0008
DS  0008
ES  0008
FS  0008
GS  0008
```

A.1.14 Clear CMOS

Usage:
clearcmos

Function:
Clear CMOS to 0

Example:
Clear CMOS to 0:


AMD BIOSDBG 1.33 - Cmds

```
 22 of 22                                                   READY

 -clearcmos                                                  core 00

 -cmos                                                       EAX 8000001D
 00:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        EBX 000000B8
 10:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        ECX 00000000
 20:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        EDX E0000000
 30:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        ESI 00000004
 40:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        EDI 000000B8
 50:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        EBP 0003FB1C
 60:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        ESP 0003FB08
 70:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        EIP FFFA1F81
 80:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        EFL 00000087
 90:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        CS  0010
 A0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        SS  0008
 B0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        DS  0008
 C0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        ES  0008
 D0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        FS  0008
 E0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00        GS  0008
 F0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

-

F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F10-Help ?-commands
```

A.1.15 Read Pstate

Usage:

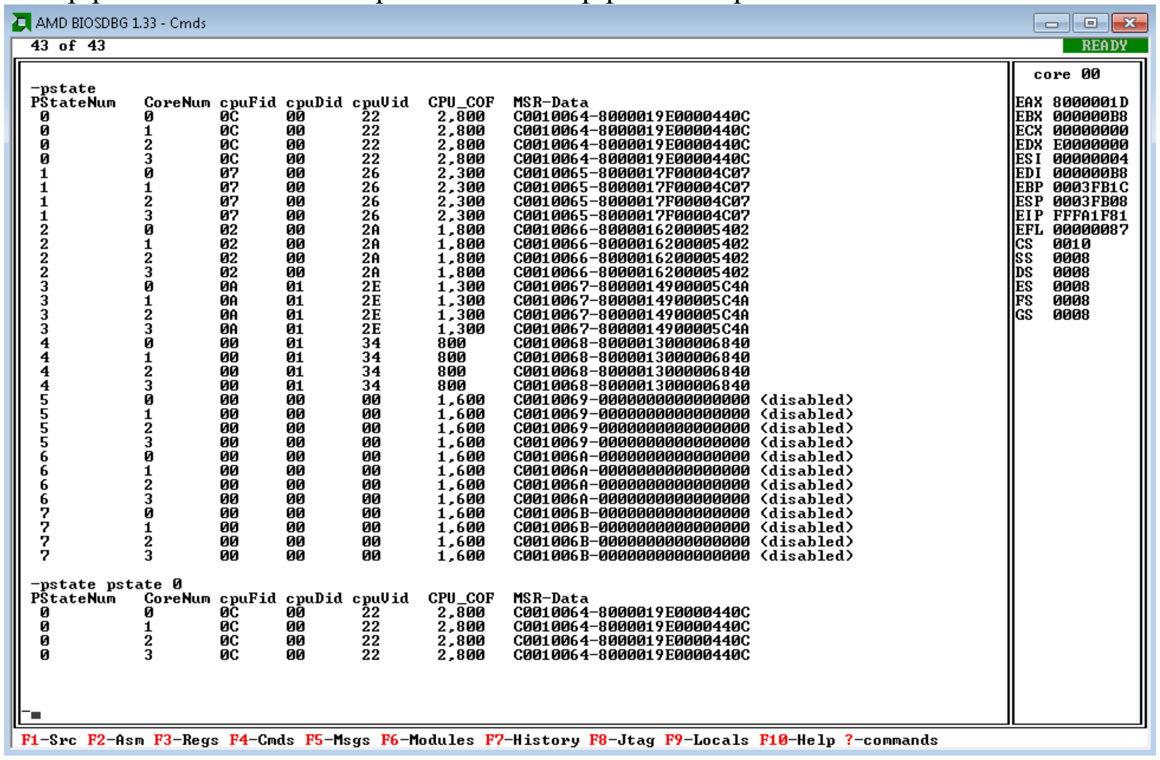
pstate [core #coreNum] | [pstate #pstateNum]

Function:

Dump pstate table, for specific core or specific pstate number

Example:

Dump pstate for all cores all pstates and dump pstate for pstate0 for all cores:



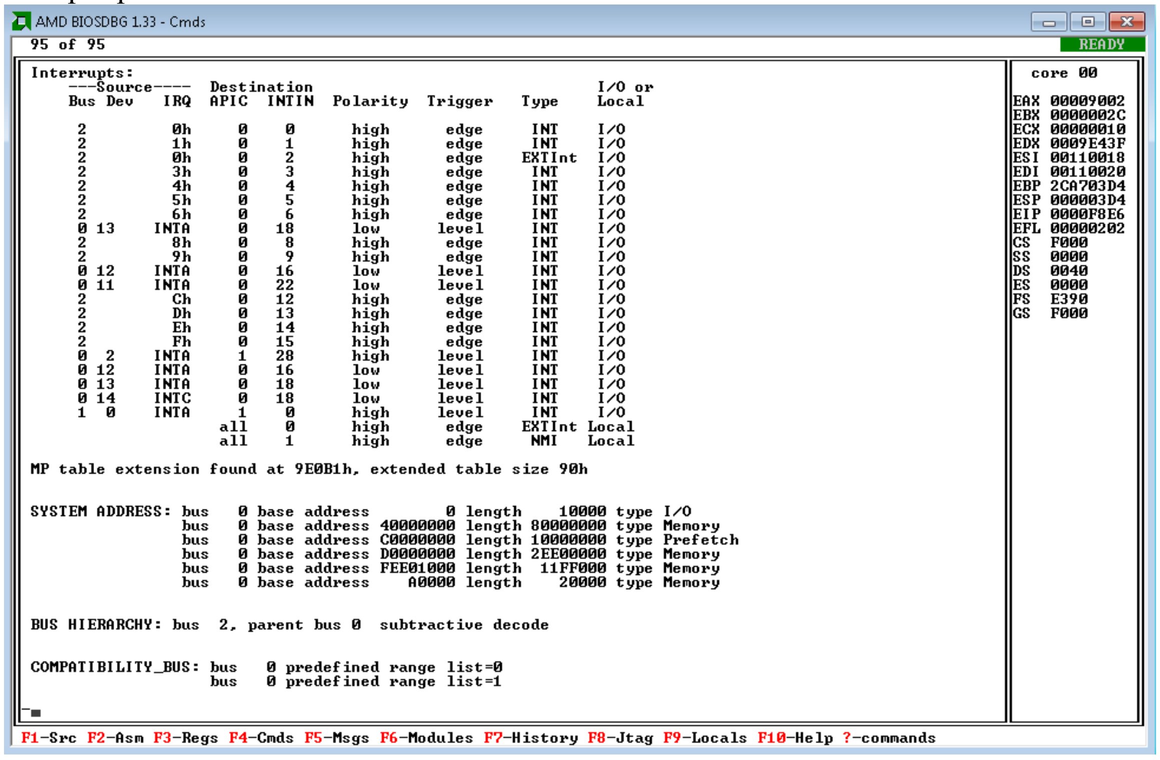
```
AMD BIOSDBG 1.33 - Cmds
 43 of 43                                                                         READY

 -pstate                                                                           core 00
 PStateNum     CoreNum cpuFid cpuDid cpuVid   CPU_COF   MSR-Data
  0            0       0C     00      22       2,800    C0010064-8000019E0000440C                 EAX 8000001D
  0            1       0C     00      22       2,800    C0010064-8000019E0000440C                 EBX 000000B8
  0            2       0C     00      22       2,800    C0010064-8000019E0000440C                 ECX 00000000
  0            3       0C     00      22       2,800    C0010064-8000019E0000440C                 EDX E0000000
  1            0       07     00      26       2,300    C0010065-8000017F00004C07                 ESI 00000004
  1            1       07     00      26       2,300    C0010065-8000017F00004C07                 EDI 000000B8
  1            2       07     00      26       2,300    C0010065-8000017F00004C07                 EBP 0003FB1C
  1            3       07     00      26       2,300    C0010065-8000017F00004C07                 ESP 0003FB08
  2            0       02     00      2A       1,800    C0010066-8000016200005402                 EIP FFFA1F81
  2            1       02     00      2A       1,800    C0010066-8000016200005402                 EFL 00000087
  2            2       02     00      2A       1,800    C0010066-8000016200005402                 CS  0010
  2            3       02     00      2A       1,800    C0010066-8000016200005402                 SS  0008
  3            0       0A     01      2E       1,300    C0010067-8000014900005C4A                 DS  0008
  3            1       0A     01      2E       1,300    C0010067-8000014900005C4A                 ES  0008
  3            2       0A     01      2E       1,300    C0010067-8000014900005C4A                 FS  0008
  3            3       0A     01      2E       1,300    C0010067-8000014900005C4A                 GS  0008
  4            0       00     01      34      800       C0010068-8000013000006840
  4            1       00     01      34      800       C0010068-8000013000006840
  4            2       00     01      34      800       C0010068-8000013000006840
  4            3       00     01      34      800       C0010068-8000013000006840
  5            0       00     00      00       1,600    C0010069-0000000000000000 (disabled)
  5            1       00     00      00       1,600    C0010069-0000000000000000 (disabled)
  5            2       00     00      00       1,600    C0010069-0000000000000000 (disabled)
  5            3       00     00      00       1,600    C0010069-0000000000000000 (disabled)
  6            0       00     00      00       1,600    C001006A-0000000000000000 (disabled)
  6            1       00     00      00       1,600    C001006A-0000000000000000 (disabled)
  6            2       00     00      00       1,600    C001006A-0000000000000000 (disabled)
  6            3       00     00      00       1,600    C001006A-0000000000000000 (disabled)
  7            0       00     00      00       1,600    C001006B-0000000000000000 (disabled)
  7            1       00     00      00       1,600    C001006B-0000000000000000 (disabled)
  7            2       00     00      00       1,600    C001006B-0000000000000000 (disabled)
  7            3       00     00      00       1,600    C001006B-0000000000000000 (disabled)

 -pstate pstate 0
 PStateNum     CoreNum cpuFid cpuDid cpuVid   CPU_COF   MSR-Data
  0            0       0C     00      22       2,800    C0010064-8000019E0000440C
  0            1       0C     00      22       2,800    C0010064-8000019E0000440C
  0            2       0C     00      22       2,800    C0010064-8000019E0000440C
  0            3       0C     00      22       2,800    C0010064-8000019E0000440C


-
F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F10-Help ?-commands
```

A.1.16 Dump MPS table

Usage:
dumpmp

Function:
Dump mps table

Example:
Dump mps table:



AMD BIOSDBG 1.33 - Cmds

```
 95 of 95                                                              READY

 Interrupts:                                                         core 00
     ---Source----  Destination                          I/O or
     Bus Dev    IRQ  APIC  INTIN  Polarity  Trigger   Type     Local     EAX 00009002
                                                                         EBX 0000002C
      2          0h    0     0      high      edge      INT     I/O      ECX 00000010
      2          1h    0     1      high      edge      INT     I/O      EDX 0009E43F
      2          0h    0     2      high      edge    EXTInt    I/O      ESI 00110018
      2          3h    0     3      high      edge      INT     I/O      EDI 00110020
      2          4h    0     4      high      edge      INT     I/O      EBP 2CA703D4
      2          5h    0     5      high      edge      INT     I/O      ESP 000003D4
      2          6h    0     6      high      edge      INT     I/O      EIP 0000F8E6
      0 13     INTA    0    18      low      level      INT     I/O      EFL 00000202
      2          8h    0     8      high      edge      INT     I/O      CS  F000
      2          9h    0     9      high      edge      INT     I/O      SS  0000
      0 12     INTA    0    16      low      level      INT     I/O      DS  0040
      0 11     INTA    0    22      low      level      INT     I/O      ES  0000
      2          Ch    0    12      high      edge      INT     I/O      FS  E390
      2          Dh    0    13      high      edge      INT     I/O      GS  F000
      2          Eh    0    14      high      edge      INT     I/O
      2          Fh    0    15      high      edge      INT     I/O
      0  2     INTA    1    28      high     level      INT     I/O
      0 12     INTA    0    16      low      level      INT     I/O
      0 13     INTA    0    18      low      level      INT     I/O
      0 14     INTC    0    18      low      level      INT     I/O
      1  0     INTA    1     0      high     level      INT     I/O
                     all     0      high      edge    EXTInt  Local
                     all     1      high      edge      NMI   Local

 MP table extension found at 9E0B1h, extended table size 90h


 SYSTEM ADDRESS: bus    0 base address        0 length    10000 type I/O
                 bus    0 base address 40000000 length 80000000 type Memory
                 bus    0 base address C0000000 length 10000000 type Prefetch
                 bus    0 base address D0000000 length 2EE00000 type Memory
                 bus    0 base address FEE01000 length  11FF000 type Memory
                 bus    0 base address    A0000 length    20000 type Memory


 BUS HIERARCHY: bus  2, parent bus 0  subtractive decode


 COMPATIBILITY_BUS: bus   0 predefined range list=0
                    bus   0 predefined range list=1

F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F10-Help ?-commands
```

A.1.17 Dump IO APIC

Usage:
ioapic [x]

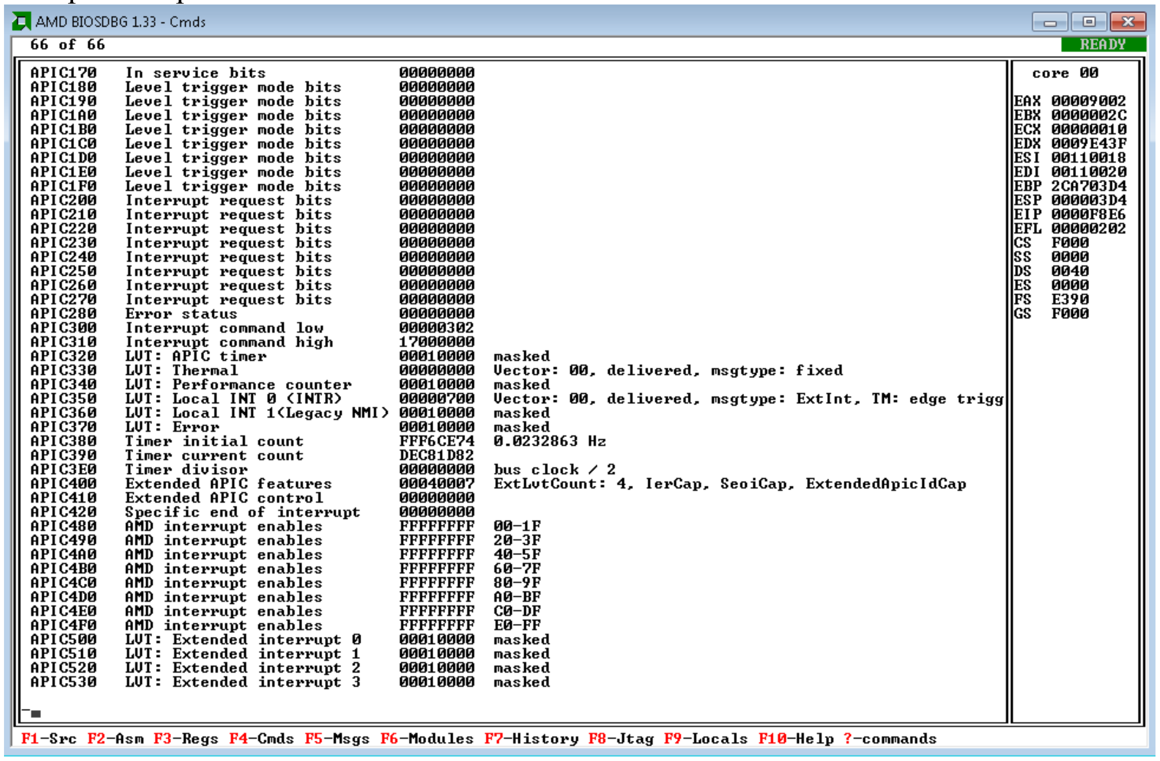
Function:
Dump io apic at address FEC00000 by default or at specific address [x]

A.1.18 Dump local APIC

Usage:
localapic [x]

Function:
Dump local apic at address FEE00000 by default or at specific address [x]

Example:
Dump local apic at address 0xFEE00000:



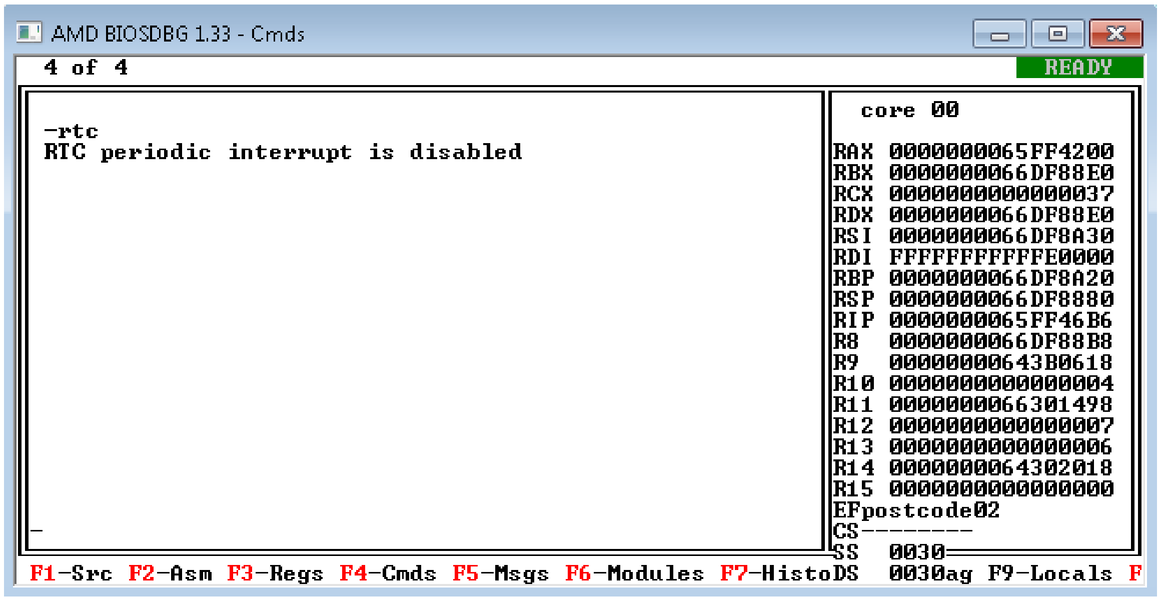
```
 66 of 66

APIC170   In service bits          00000000                                           core 00
APIC180   Level trigger mode bits  00000000
APIC190   Level trigger mode bits  00000000                                          EAX 00009002
APIC1A0   Level trigger mode bits  00000000                                          EBX 0000002C
APIC1B0   Level trigger mode bits  00000000                                          ECX 00000010
APIC1C0   Level trigger mode bits  00000000                                          EDX 0009E43F
APIC1D0   Level trigger mode bits  00000000                                          ESI 00110018
APIC1E0   Level trigger mode bits  00000000                                          EDI 00110020
APIC1F0   Level trigger mode bits  00000000                                          EBP 2CA703D4
APIC200   Interrupt request bits   00000000                                          ESP 000003D4
APIC210   Interrupt request bits   00000000                                          EIP 0000F8E6
APIC220   Interrupt request bits   00000000                                          EFL 00000202
APIC230   Interrupt request bits   00000000                                          CS  F000
APIC240   Interrupt request bits   00000000                                          SS  0000
APIC250   Interrupt request bits   00000000                                          DS  0040
APIC260   Interrupt request bits   00000000                                          ES  0000
APIC270   Interrupt request bits   00000000                                          FS  E390
APIC280   Error status             00000000                                          GS  F000
APIC300   Interrupt command low    00000302
APIC310   Interrupt command high   17000000
APIC320   LVT: APIC timer          00010000  masked
APIC330   LVT: Thermal             00000000  Vector: 00, delivered, msgtype: fixed
APIC340   LVT: Performance counter 00010000  masked
APIC350   LVT: Local INT 0 (INTR)  00000700  Vector: 00, delivered, msgtype: ExtInt, TM: edge trigg
APIC360   LVT: Local INT 1(Legacy NMI) 00010000  masked
APIC370   LVT: Error               00010000  masked
APIC380   Timer initial count      FFF6CE74  0.0232863 Hz
APIC390   Timer current count      DEC81D82
APIC3E0   Timer divisor            00000000  bus clock / 2
APIC400   Extended APIC features   00040007  ExtLvtCount: 4, IerCap, SeoiCap, ExtendedApicIdCap
APIC410   Extended APIC control    00000000
APIC420   Specific end of interrupt 00000000
APIC480   AMD interrupt enables    FFFFFFFF  00-1F
APIC490   AMD interrupt enables    FFFFFFFF  20-3F
APIC4A0   AMD interrupt enables    FFFFFFFF  40-5F
APIC4B0   AMD interrupt enables    FFFFFFFF  60-7F
APIC4C0   AMD interrupt enables    FFFFFFFF  80-9F
APIC4D0   AMD interrupt enables    FFFFFFFF  A0-BF
APIC4E0   AMD interrupt enables    FFFFFFFF  C0-DF
APIC4F0   AMD interrupt enables    FFFFFFFF  E0-FF
APIC500   LVT: Extended interrupt 0 00010000  masked
APIC510   LVT: Extended interrupt 1 00010000  masked
APIC520   LVT: Extended interrupt 2 00010000  masked
APIC530   LVT: Extended interrupt 3 00010000  masked

-
F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F10-Help ?-commands
```

A.1.19 Dump RTC info

Usage:
rtc

Function:
Dump RTC information

Example:
Dump RTC information:



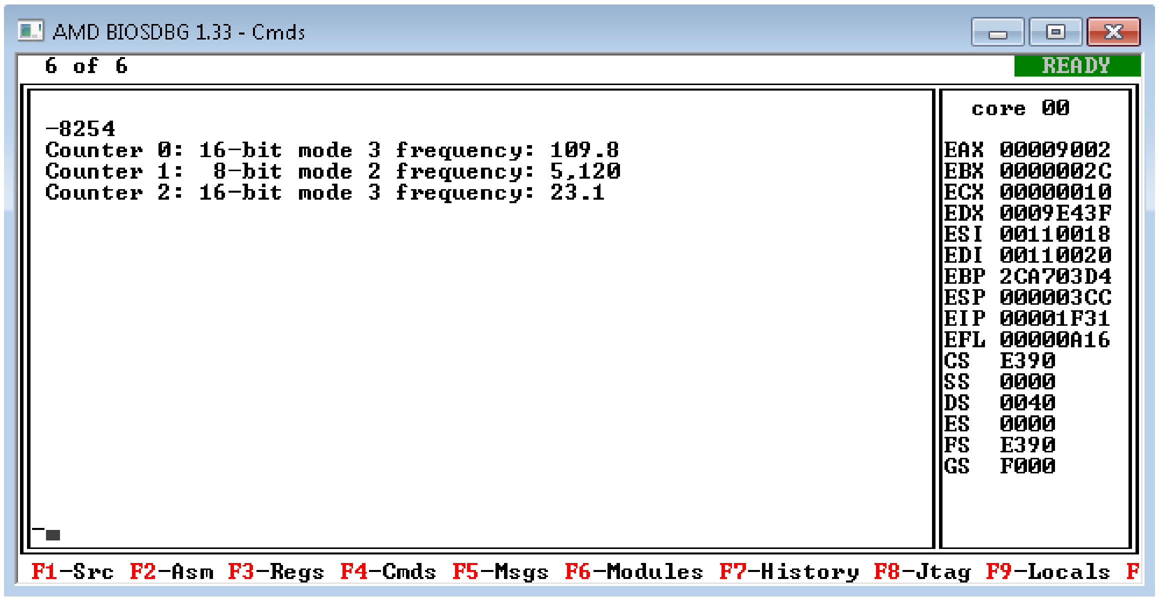
```
AMD BIOSDBG 1.33 - Cmds
 4 of 4                                                          READY
                                                 core 00
 -rtc
 RTC periodic interrupt is disabled             RAX 0000000065FF4200
                                                RBX 0000000066DF88E0
                                                RCX 0000000000000037
                                                RDX 0000000066DF88E0
                                                RSI 0000000066DF8A30
                                                RDI FFFFFFFFFFFE0000
                                                RBP 0000000066DF8A20
                                                RSP 0000000066DF8880
                                                RIP 0000000065FF46B6
                                                R8  0000000066DF88B8
                                                R9  00000000643B0618
                                                R10 0000000000000004
                                                R11 0000000066301498
                                                R12 0000000000000007
                                                R13 0000000000000006
                                                R14 0000000064302018
                                                R15 0000000000000000
                                                EFpostcode02
 _                                              CS---------
                                                SS  0030
F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-HistoDS  0030ag F9-Locals F
```

A.1.20 Dump 8254 PIT info

Usage:
8254

Function:
Dump 8254 PIT information

Example:
Dump 8254 PIT information:



AMD BIOSDBG 1.33 - Cmds
6 of 6
READY

```
-8254
Counter 0: 16-bit mode 3 frequency: 109.8
Counter 1:  8-bit mode 2 frequency: 5,120
Counter 2: 16-bit mode 3 frequency: 23.1

-
```

```
  core 00

EAX 00009002
EBX 0000002C
ECX 00000010
EDX 0009E43F
ESI 00110018
EDI 00110020
EBP 2CA703D4
ESP 000003CC
EIP 00001F31
EFL 00000A16
CS   E390
SS   0000
DS   0040
ES   0000
FS   E390
GS   F000
```

F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F

A.1.21 Target to specific core

A.1.21.1 Display misc core info

Usage:
cores

Function:
Display misc core info

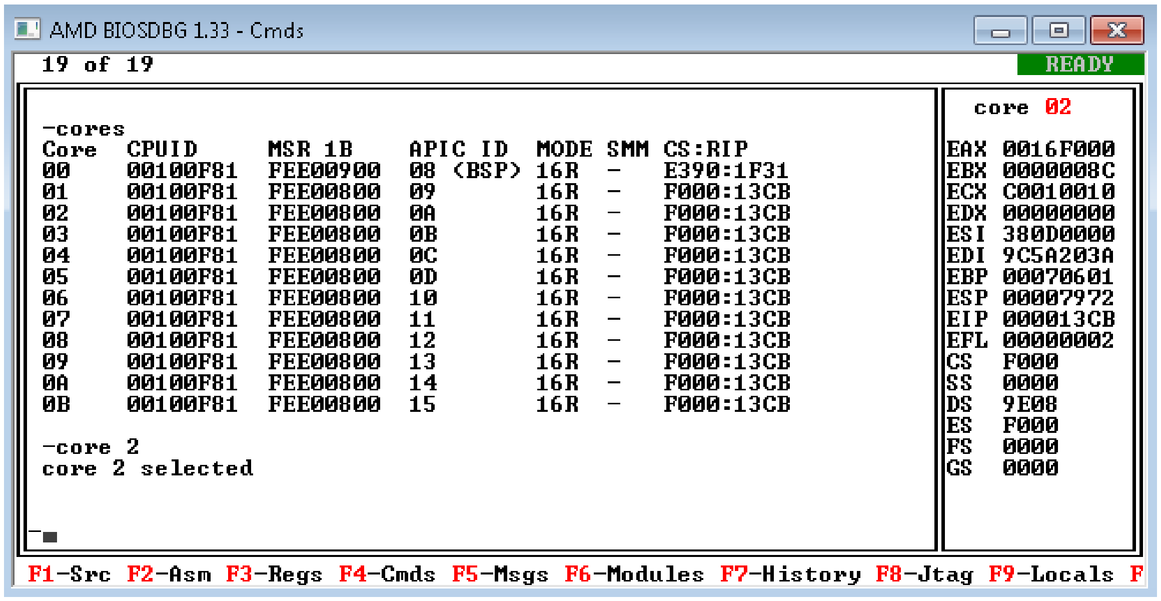
Example:
Display misc core info:
See below picture.

A.1.21.2 Select specific core

Usage:
core x

Function:
Display core selection and select core x for debugging

Example:
Select core 2:



```
 19 of 19

-cores
Core   CPUID      MSR 1B     APIC ID  MODE SMM  CS:RIP
00     00100F81   FEE00900   08 (BSP) 16R  -    E390:1F31
01     00100F81   FEE00800   09       16R  -    F000:13CB
02     00100F81   FEE00800   0A       16R  -    F000:13CB
03     00100F81   FEE00800   0B       16R  -    F000:13CB
04     00100F81   FEE00800   0C       16R  -    F000:13CB
05     00100F81   FEE00800   0D       16R  -    F000:13CB
06     00100F81   FEE00800   10       16R  -    F000:13CB
07     00100F81   FEE00800   11       16R  -    F000:13CB
08     00100F81   FEE00800   12       16R  -    F000:13CB
09     00100F81   FEE00800   13       16R  -    F000:13CB
0A     00100F81   FEE00800   14       16R  -    F000:13CB
0B     00100F81   FEE00800   15       16R  -    F000:13CB

-core 2
core 2 selected

-
```

```
  core 02
EAX 0016F000
EBX 0000008C
ECX C0010010
EDX 00000000
ESI 380D0000
EDI 9C5A203A
EBP 00070601
ESP 00007972
EIP 000013CB
EFL 00000002
CS  F000
SS  0000
DS  9E08
ES  F000
FS  0000
GS  0000
```

F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F

A.1.22 Reset, Load / Reload, Go and Stop, Sleep and Relay command

A.1.22.1 Reset

Usage:
reset

Function:
Reset target and stop at reset vector.

A.1.22.2 Reload BIOS image

Usage:
reset

Function:
Reload the BIOS image file.

A.1.22.3 Load LPC rom

Usage:
lpcload <filename>

Function:
Load project bios image ”filename” into rom emulator.

A.1.22.4 Save LPC rom

Usage:
lpcsave <filename>

Function:
Save rom emulator contents to disk file.

A.1.22.5 Run target

Usage:
go

Function:
Command line go command, resume target execution.

A.1.22.6 Stop target

Usage:
stop

Function:
Command line stop command, stop target execution.

A.1.22.7 Sleep

Usage:
sleep x

Function:
Sleep x milliseconds (use ';' to combine with other commands).

A.1.22.8 Relay

Usage:
relay x [y]
Function:
Pulse relay x for 200 ms [or y ms, y is decimal]. Make sure relay connection is fine.

A.1.23 Breakpoints

See section 5.1 and 5.2 about how to set breakpoints in source.

A.1.23.1 Software breakpoints

Usage:
swbp addr=x

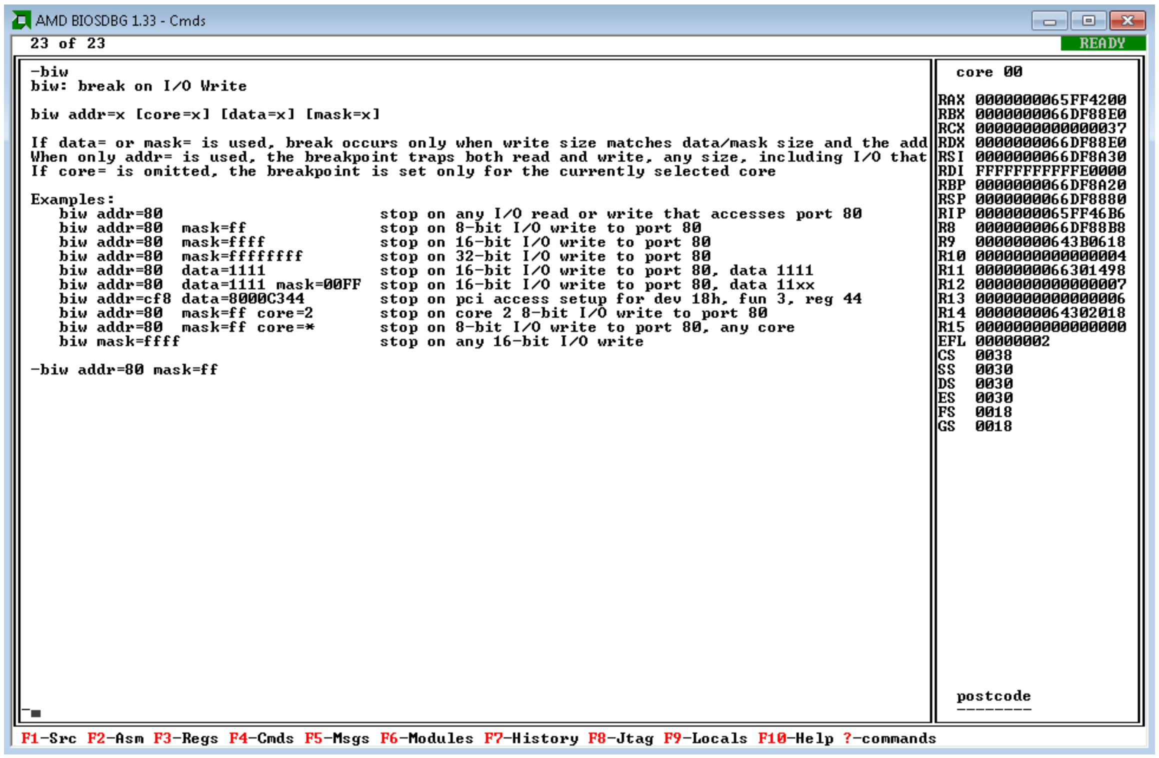
Function:
Add software breakpoint at address x (for SimNow™)

A.1.23.2 Hardware IO breakpoints

Usage:
biw

Function:
Display help for hardware I/O write breakpoint

Example:
Display hardware I/O breakpoint help info, set hardware IO breakpoint on any I/O read or write that access port 80:



```
-biw
biw: break on I/O Write

biw addr=x [core=x] [data=x] [mask=x]

If data= or mask= is used, break occurs only when write size matches data/mask size and the add
When only addr= is used, the breakpoint traps both read and write, any size, including I/O that
If core= is omitted, the breakpoint is set only for the currently selected core

Examples:
   biw addr=80                        stop on any I/O read or write that accesses port 80
   biw addr=80  mask=ff               stop on 8-bit I/O write to port 80
   biw addr=80  mask=ffff             stop on 16-bit I/O write to port 80
   biw addr=80  mask=ffffffff         stop on 32-bit I/O write to port 80
   biw addr=80  data=1111             stop on 16-bit I/O write to port 80, data 1111
   biw addr=80  data=1111 mask=00FF   stop on 16-bit I/O write to port 80, data 11xx
   biw addr=cf8 data=8000C344         stop on pci access setup for dev 18h, fun 3, reg 44
   biw addr=80  mask=ff core=2        stop on core 2 8-bit I/O write to port 80
   biw addr=80  mask=ff core=*        stop on 8-bit I/O write to port 80, any core
   biw mask=ffff                      stop on any 16-bit I/O write

-biw addr=80 mask=ff

-
```

```
  core 00

RAX 0000000065FF4200
RBX 0000000066DF88E0
RCX 0000000000000037
RDX 0000000066DF88E0
RSI 0000000066DF8A30
RDI FFFFFFFFFFFE0000
RBP 0000000066DF8A20
RSP 0000000066DF8880
RIP 0000000065FF46B6
R8  0000000066DF88B8
R9  00000000643B0618
R10 0000000000000004
R11 0000000066301498
R12 0000000000000007
R13 0000000000000006
R14 0000000064302018
R15 0000000000000000
EFL 00000002
CS  0038
SS  0030
DS  0030
ES  0030
FS  0018
GS  0018

  postcode
  --------
```

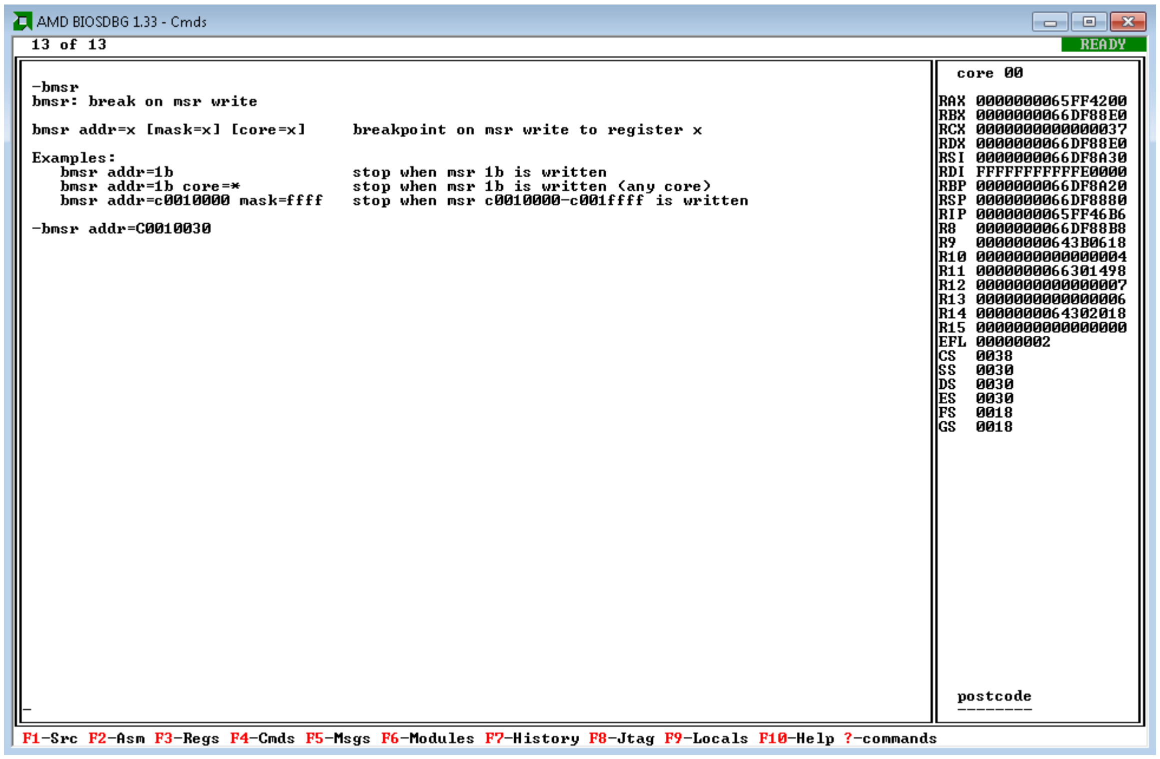
```
F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F10-Help ?-commands
```

A.1.23.3 Hardware MSR breakpoints

Usage:
bmsr

Function:
Display help for hardware MSR write breakpoint

Example:
Display hardware MSR write breakpoint help info, set break point to stop when MSR C001_0030 is written:



AMD BIOSDBG 1.33 - Cmds

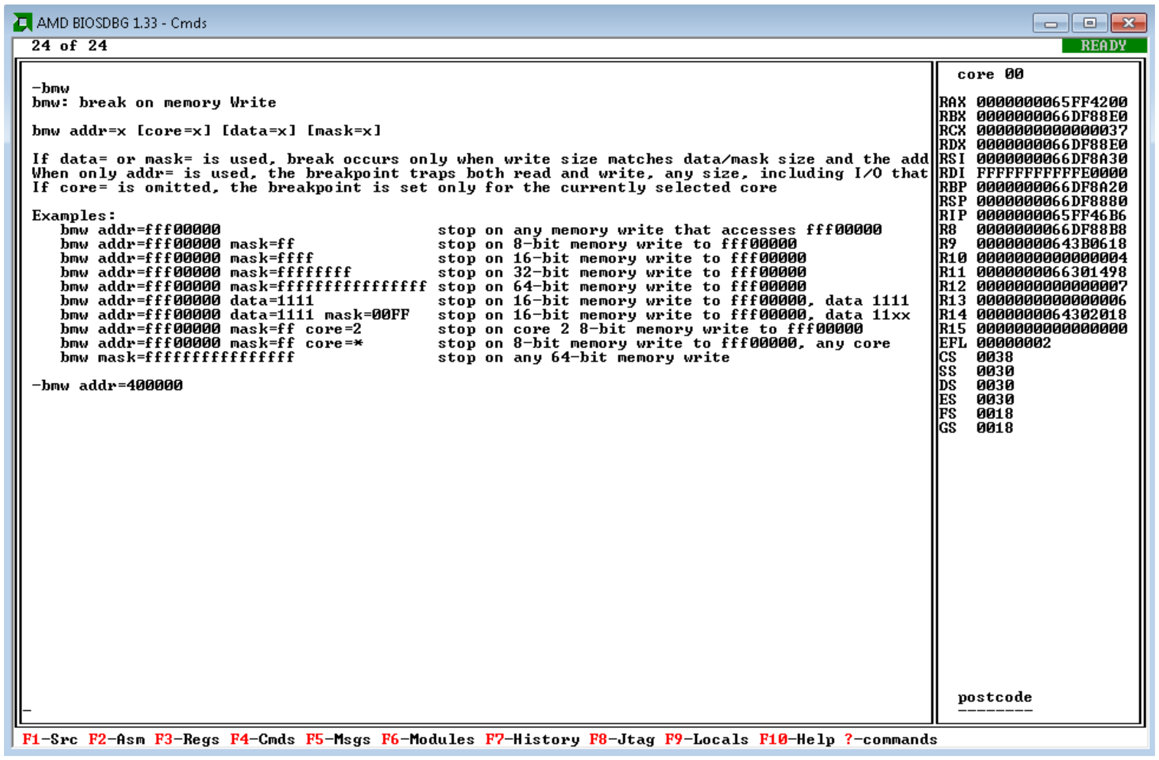
```
13 of 13                                                                        READY

-bmsr                                                                           core 00
bmsr: break on msr write
                                                                                RAX 0000000065FF4200
bmsr addr=x [mask=x] [core=x]     breakpoint on msr write to register x         RBX 000000066DF88E0
                                                                                RCX 0000000000000037
Examples:                                                                       RDX 000000066DF88E0
   bmsr addr=1b                   stop when msr 1b is written                   RSI 000000066DF8A30
   bmsr addr=1b core=*            stop when msr 1b is written (any core)        RDI FFFFFFFFFFFE0000
   bmsr addr=c0010000 mask=ffff   stop when msr c0010000-c001ffff is written    RBP 000000066DF8A20
                                                                                RSP 000000066DF8880
-bmsr addr=C0010030                                                             RIP 0000000065FF46B6
                                                                                R8  000000066DF88B8
                                                                                R9  00000000643B0618
                                                                                R10 0000000000000004
                                                                                R11 0000000066301498
                                                                                R12 0000000000000007
                                                                                R13 0000000000000006
                                                                                R14 0000000064302018
                                                                                R15 0000000000000000
                                                                                EFL 00000002
                                                                                CS  0038
                                                                                SS  0030
                                                                                DS  0030
                                                                                ES  0030
                                                                                FS  0018
                                                                                GS  0018

                                                                                postcode
_                                                                               --------
F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F10-Help ?-commands
```

A.1.23.4 Hardware memory write breakpoints

Usage:
bmw

Function:
Display help for hardware memory write breakpoint

Example:
Display hardware memory write breakpoint help info, set break point to stop when address 0x400000 is accessed:



```
24 of 24

-bmw
bmw: break on memory Write

bmw addr=x [core=x] [data=x] [mask=x]

If data= or mask= is used, break occurs only when write size matches data/mask size and the add
When only addr= is used, the breakpoint traps both read and write, any size, including I/O that
If core= is omitted, the breakpoint is set only for the currently selected core

Examples:
   bmw addr=fff00000                          stop on any memory write that accesses fff00000
   bmw addr=fff00000 mask=ff                  stop on 8-bit memory write to fff00000
   bmw addr=fff00000 mask=ffff                stop on 16-bit memory write to fff00000
   bmw addr=fff00000 mask=ffffffff            stop on 32-bit memory write to fff00000
   bmw addr=fff00000 mask=ffffffffffffffff    stop on 64-bit memory write to fff00000
   bmw addr=fff00000 data=1111                stop on 16-bit memory write to fff00000, data 1111
   bmw addr=fff00000 data=1111 mask=00FF      stop on 16-bit memory write to fff00000, data 11xx
   bmw addr=fff00000 mask=ff core=2           stop on core 2 8-bit memory write to fff00000
   bmw addr=fff00000 mask=ff core=*           stop on 8-bit memory write to fff00000, any core
   bmw mask=ffffffffffffffff                  stop on any 64-bit memory write

-bmw addr=400000

_
```

```
  core 00

RAX 0000000065FF4200
RBX 0000000066DF88E0
RCX 0000000000000037
RDX 0000000066DF88E0
RSI 0000000066DF8A30
RDI FFFFFFFFFFFE0000
RBP 0000000066DF8A20
RSP 0000000066DF8880
RIP 0000000065FF46B6
R8  0000000066DF88B8
R9  00000000643B0618
R10 0000000000000004
R11 0000000066301498
R12 0000000000000007
R13 0000000000000006
R14 0000000064302018
R15 0000000000000000
EFL 00000002
CS  0038
SS  0030
DS  0030
ES  0030
FS  0018
GS  0018

  postcode
  --------
```

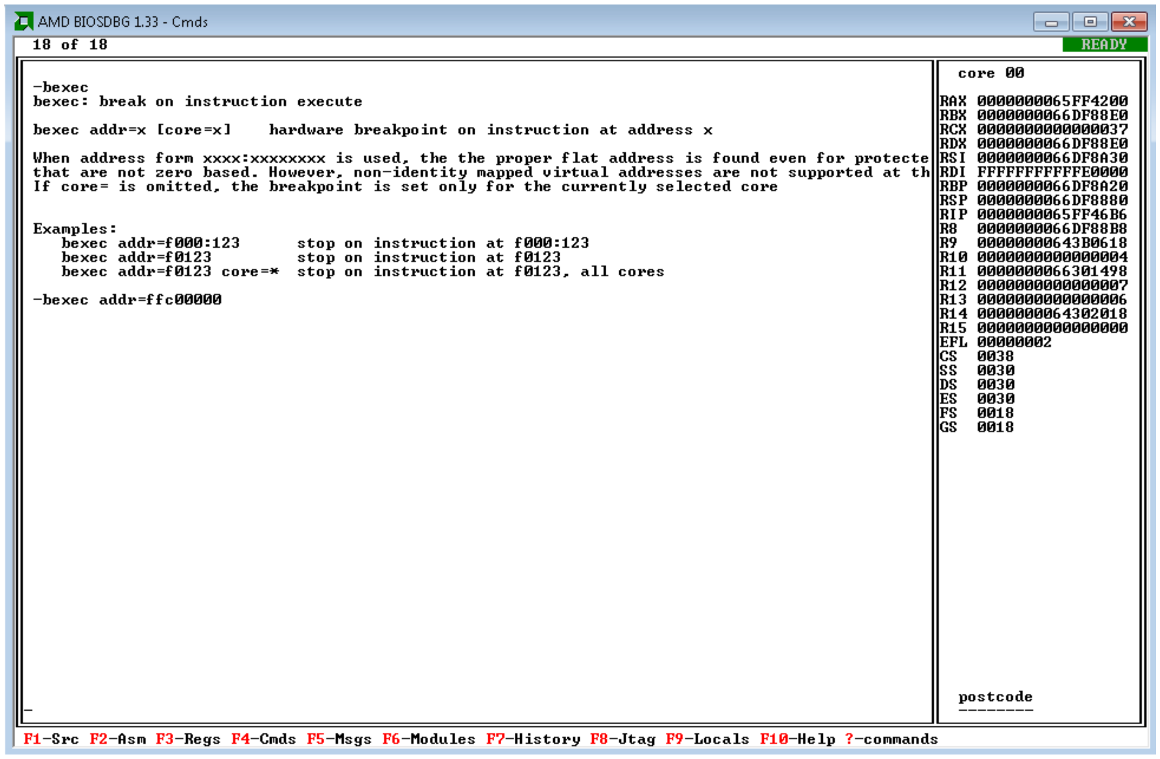
```
F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F10-Help ?-commands
```

A.1.23.5 Hardware instruction breakpoints

Usage:
bexec

Function:
Display help for hardware instruction breakpoint

Example:
Display hardware instruction breakpoint help info, set break point to stop on instruction 0xFFC00000:



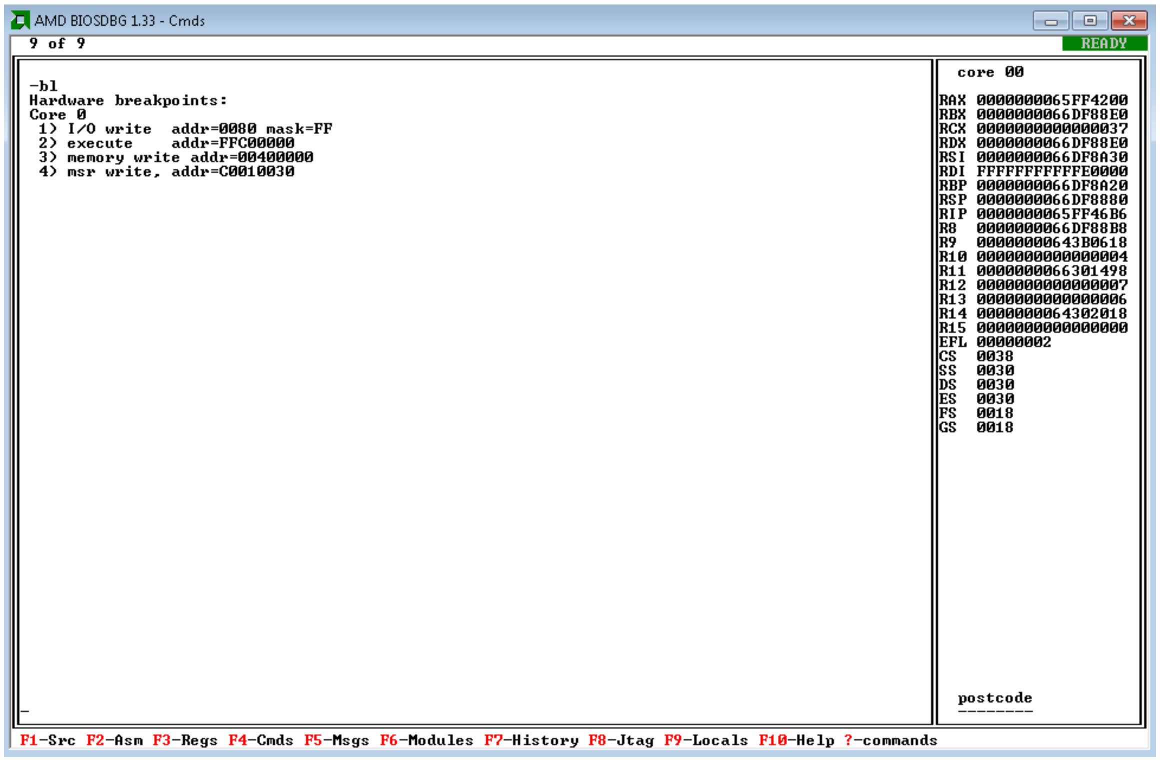
```
AMD BIOSDBG 1.33 - Cmds
 18 of 18                                                                                   READY

 -bexec                                                                                 core 00
 bexec: break on instruction execute
                                                                                      RAX 0000000065FF4200
 bexec addr=x [core=x]    hardware breakpoint on instruction at address x             RBX 0000000066DF88E0
                                                                                      RCX 0000000000000037
 When address form xxxx:xxxxxxxx is used, the the proper flat address is found even for protecte RDX 0000000066DF88E0
 that are not zero based. However, non-identity mapped virtual addresses are not supported at th RSI 0000000066DF8A30
 If core= is omitted, the breakpoint is set only for the currently selected core      RDI FFFFFFFFFFFE0000
                                                                                      RBP 0000000066DF8A20
                                                                                      RSP 0000000066DF8880
 Examples:                                                                            RIP 0000000065FF46B6
    bexec addr=f000:123      stop on instruction at f000:123                          R8  0000000066DF88B8
    bexec addr=f0123         stop on instruction at f0123                             R9  00000000643B0618
    bexec addr=f0123 core=*  stop on instruction at f0123, all cores                  R10 0000000000000004
                                                                                      R11 0000000066301498
 -bexec addr=ffc00000                                                                 R12 0000000000000007
                                                                                      R13 0000000000000006
                                                                                      R14 0000000064302018
                                                                                      R15 0000000000000000
                                                                                      EFL 00000002
                                                                                      CS  0038
                                                                                      SS  0030
                                                                                      DS  0030
                                                                                      ES  0030
                                                                                      FS  0018
                                                                                      GS  0018

                                                                                        postcode
 _                                                                                      --------
 F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F10-Help ?-commands
```

A.1.23.6 List breakpoints

Usage:
bl

Function:
List the breakpoints

Example:
List the breakpoints:



AMD BIOSDBG 1.33 - Cmds
READY
F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F10-Help ?-commands

```
 9 of 9

-bl
Hardware breakpoints:
Core 0
 1) I/O write   addr=0080 mask=FF
 2) execute     addr=FFC00000
 3) memory write addr=00400000
 4) msr write, addr=C0010030
```

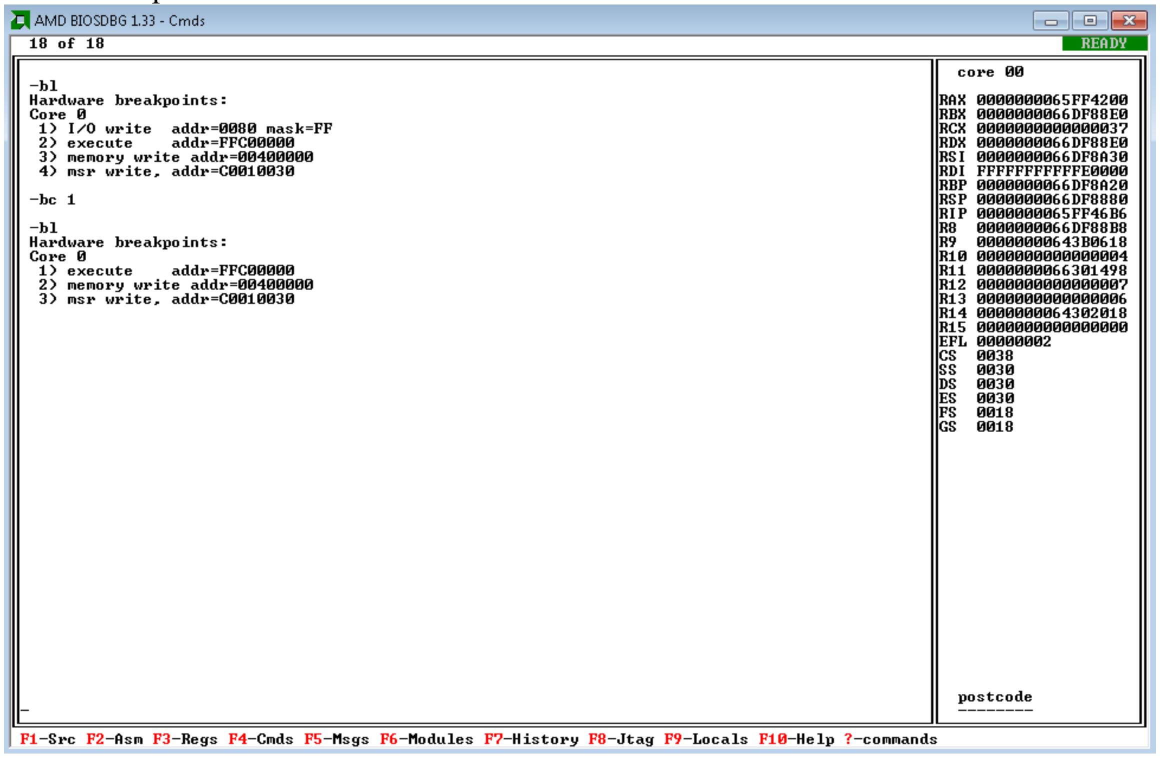
```
  core 00

RAX 0000000065FF4200
RBX 0000000066DF88E0
RCX 0000000000000037
RDX 0000000066DF88E0
RSI 0000000066DF8A30
RDI FFFFFFFFFFFE0000
RBP 0000000066DF8A20
RSP 0000000066DF8880
RIP 0000000065FF46B6
R8  0000000066DF88B8
R9  00000000643B0618
R10 0000000000000004
R11 0000000066301498
R12 0000000000000007
R13 0000000000000006
R14 0000000064302018
R15 0000000000000000
EFL 00000002
CS  0038
SS  0030
DS  0030
ES  0030
FS  0018
GS  0018

  postcode
  --------
```

A.1.23.7 Clear breakpoints

Usage:
bc [x | *]

Function:
Clear breakpoint x, or clear all breakpoints

Example:
Clear breakpoint 1:



```
 18 of 18

-bl
Hardware breakpoints:
Core 0
 1) I/O write  addr=0080 mask=FF
 2) execute    addr=FFC00000
 3) memory write addr=00400000
 4) msr write, addr=C0010030

-bc 1

-bl
Hardware breakpoints:
Core 0
 1) execute    addr=FFC00000
 2) memory write addr=00400000
 3) msr write, addr=C0010030
```

```
  core 00

RAX 0000000065FF4200
RBX 0000000066DF88E0
RCX 0000000000000037
RDX 0000000066DF88E0
RSI 0000000066DF8A30
RDI FFFFFFFFFFFE0000
RBP 0000000066DF8A20
RSP 0000000066DF8880
RIP 0000000065FF46B6
R8  0000000066DF88B8
R9  00000000643B0618
R10 0000000000000004
R11 0000000066301498
R12 0000000000000007
R13 0000000000000006
R14 0000000064302018
R15 0000000000000000
EFL 00000002
CS  0038
SS  0030
DS  0030
ES  0030
FS  0018
GS  0018

  postcode
  --------
```

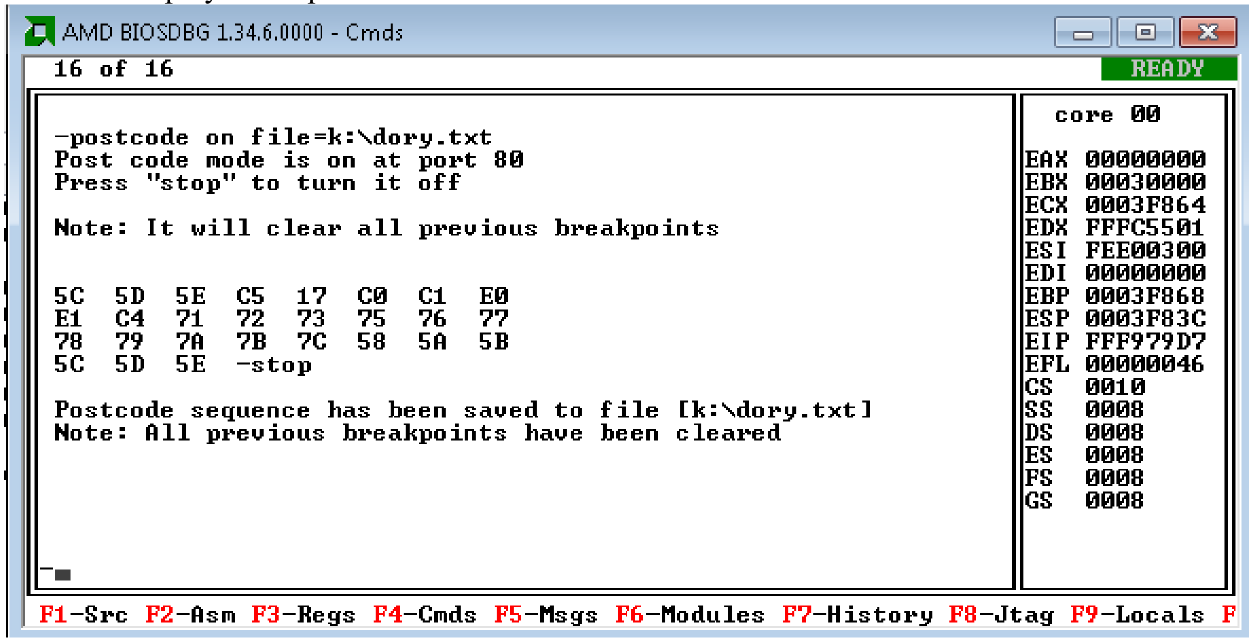
F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F10-Help ?-commands

A.1.24 Postcode display

Usage:
Postcode on [addr=xx (port 80 by default)] [file=saveFile]

Function:
Display postcode on addr continually, save to <saveFile> if it's defined

Example:
Postcode display on 80 port:


AMD BIOSDBG 1.34.6.0000 - Cmds
16 of 16
READY
F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F

```
-postcode on file=k:\dory.txt
Post code mode is on at port 80
Press "stop" to turn it off

Note: It will clear all previous breakpoints


5C   5D   5E   C5   17   C0   C1   E0
E1   C4   71   72   73   75   76   77
78   79   7A   7B   7C   58   5A   5B
5C   5D   5E   -stop

Postcode sequence has been saved to file [k:\dory.txt]
Note: All previous breakpoints have been cleared



-
```

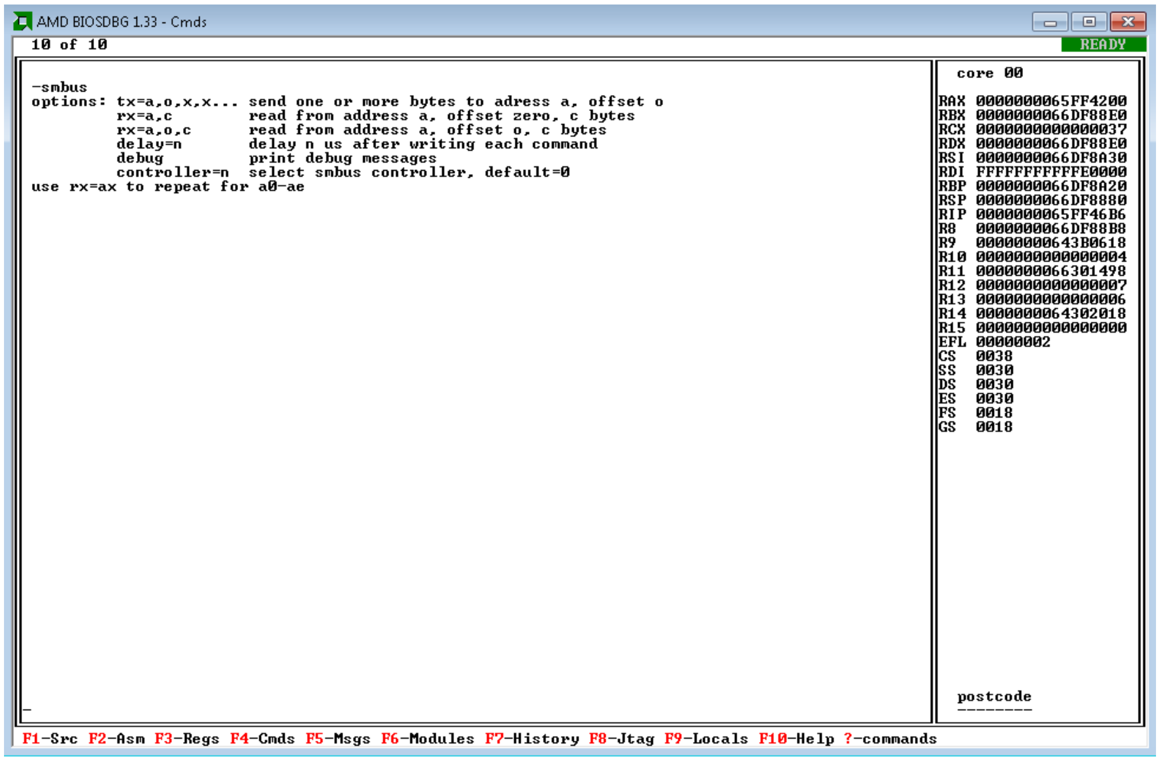
```
  core 00

EAX 00000000
EBX 00030000
ECX 0003F864
EDX FFFC5501
ESI FEE00300
EDI 00000000
EBP 0003F868
ESP 0003F83C
EIP FFF979D7
EFL 00000046
CS  0010
SS  0008
DS  0008
ES  0008
FS  0008
GS  0008
```

A.1.25 Smbus

Usage:
smbus

Function:
Smbus commands (good for DIMM SPD dump)

Example:
Smbus commands:



```
AMD BIOSDBG 1.33 - Cmds
 10 of 10                                                                         READY

 -smbus                                                                          core 00
 options: tx=a,o,x,x...  send one or more bytes to adress a, offset o
          rx=a,c         read from address a, offset zero, c bytes              RAX 0000000065FF4200
          rx=a,o,c       read from address a, offset o, c bytes                 RBX 0000000066DF88E0
          delay=n        delay n us after writing each command                  RCX 0000000000000037
          debug          print debug messages                                   RDX 0000000066DF88E0
          controller=n   select smbus controller, default=0                     RSI 0000000066DF8A30
 use rx=ax to repeat for a0-ae                                                  RDI FFFFFFFFFFFE0000
                                                                                RBP 0000000066DF8A20
                                                                                RSP 0000000066DF8880
                                                                                RIP 0000000065FF46B6
                                                                                R8  0000000066DF88B8
                                                                                R9  00000000643B0618
                                                                                R10 0000000000000004
                                                                                R11 0000000066301498
                                                                                R12 0000000000000007
                                                                                R13 0000000000000006
                                                                                R14 0000000064302018
                                                                                R15 0000000000000000
                                                                                EFL 00000002
                                                                                CS  0038
                                                                                SS  0030
                                                                                DS  0030
                                                                                ES  0030
                                                                                FS  0018
                                                                                GS  0018

                                                                                 postcode
 _                                                                               --------
 F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F10-Help ?-commands
```

A.1.26 Misc Commands

A.1.26.1 Esc

Usage:
<Esc>

Function:
Cancel command text if present, toggles command/navigate mode.

A.1.26.2 Clear screen

Usage:
<Ctrl>L or cls

Function:
Clear command screen (use <Ctrl>C to clear any screen)

A.1.26.3 Quit

Usage:
<Alt>X or q

Function:
Quit debugger application.

A.1.26.4 Comment

Usage:
#

Function:
Comment: the entire line is ignored.

A.1.26.5 Separator

Usage:
;

Function:
Separator for multiple commands on the same line

A.1.27 Dump SB/FCH Registers

Usage:
Dpmio / dpmio2 / dabcfg / daxcfg / daxindc / daxindp / drcindc / drcindp

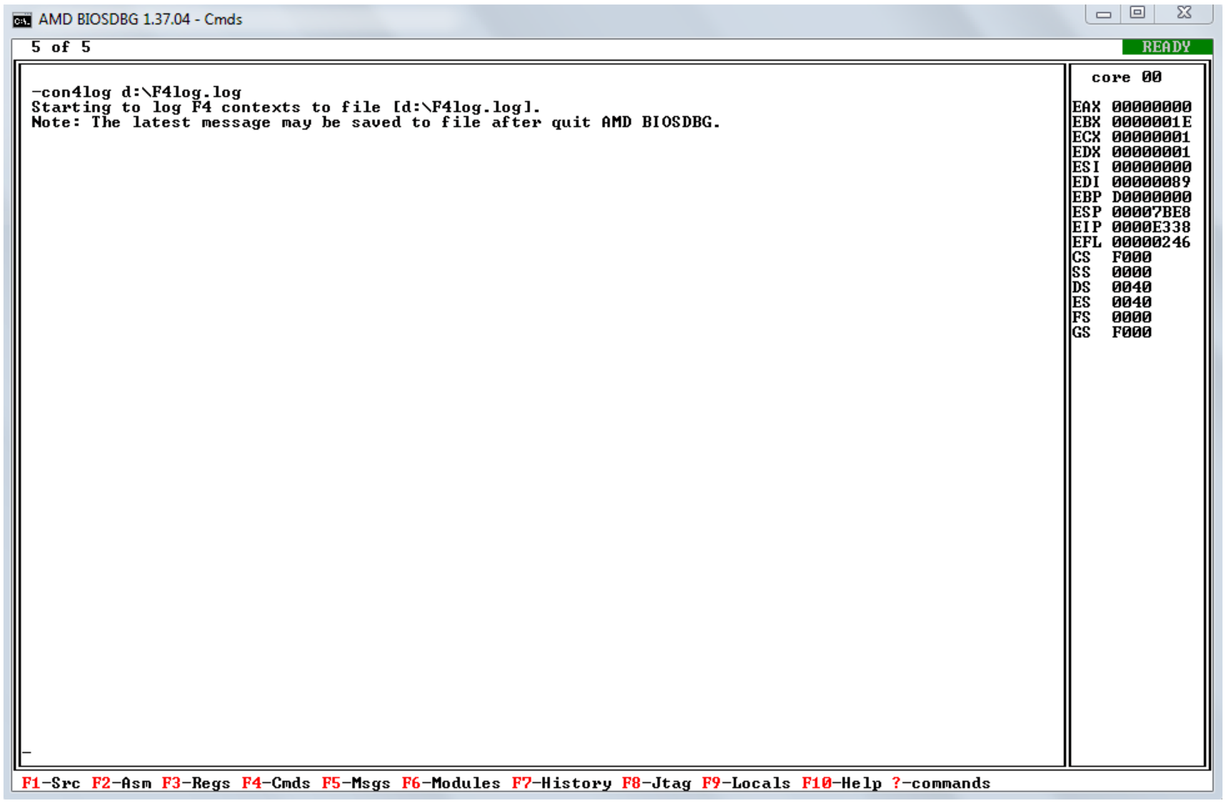
Function:
Dump PMIO / PMIO2 / ABCFG / AXCFG / AXINDC / AXINDP / RCINDC / RCINDP registers. This function is available after 1.37.

A.1.28 Log F4 contents to file

Usage:
Con4log <fileName>

Function:
Start to log F4 window messages to file <fileaName>. This function is available after 1.37.

Example:
Con4log d:\F4log.log


AMD BIOSDBG 1.37.04 - Cmds
5 of 5
READY

```
-con4log d:\F4log.log
Starting to log F4 contexts to file [d:\F4log.log].
Note: The latest message may be saved to file after quit AMD BIOSDBG.
```

```
core 00
EAX 00000000
EBX 0000001E
ECX 00000001
EDX 00000001
ESI 00000000
EDI 00000089
EBP D0000000
ESP 00007BE8
EIP 0000E338
EFL 00000246
CS  F000
SS  0000
DS  0040
ES  0040
FS  0000
GS  F000
```

F1-Src F2-Asm F3-Regs F4-Cmds F5-Msgs F6-Modules F7-History F8-Jtag F9-Locals F10-Help ?-commands

A.1.29 Dump acpi table data

Usage:
acpi

Function:
Dump the acpi tables data to files. Like FACP, SSDT, DSDT, HPET, BGRT …