NOTICE: The Processors Wiki will End-of-Life in December of 2020. It is recommended to download any files or other content you may need that are hosted on processors.wiki.ti.com. The site is now set to read only.

Debugging JTAG Connectivity Problems legacy

From Texas Instruments Wiki
Jump to: navigation, search

Contents

Strategy for debugging JTAG connectivity problems
- If using Spectrum Digital emulators
- If using BlackHawk emulators

Check your hardware JTAG connection

Check your software configuration

Location of files and utilities
- Code Composer Studio v5 and greater
- DBGJTAG GUI
- Code Composer Studio v4
- Code Composer Studio v3

Useful tests
- Emulator not plugged in
- Detect the length of the scan chain
- How to debug whether the JTAG scan path is broken
- JTAG scan chain integrity test
- How can I find out more about emulator error messages?

Strategy for debugging JTAG connectivity problems

1. Check that you are using high quality cables for connections. For example, with USB 2.0 emulators, please use a cable which is certified for USB 2.0 High Speed operation. (Poor cable quality has resulted in a failure to connect, unstable connection, etc.)

2. Determine whether the emulator is correctly setup in Windows (i.e. is the USB driver, etc. working) by checking in the Windows System Devices control panel. If this is not right, then it is possible there is a simple glitch (solved by disconnecting and connecting to a different USB port) or perhaps something more serious such as invalid or non-existing device drivers. The device drivers of most popular models of JTAG debuggers (Blackhawk and Spectrum Digital) are

5. If using devices that are able to run embedded Linux, frequently this OS prevents the JTAG debugger to properly connect to the main processor (typically a Cortex A). In this case, to make sure the JTAG connectivity is working fine, it is recommended to either halt the boot process at the u-boot prompt or to prevent Linux from running altogether.

> **Note:** recent Linux kernels disable JTAG clock entirely during its boot process, thus requiring a kernel rebuild and/or modifications.

6. If using SoC and multicore devices, it is always a good idea to manually launch the target configuration and connect to each core individually instead of clicking on the *Debug Active Project* button. For details please check GSG:Connecting to slave cores in SoC devices

> **Note:** For some C6000 and SoC devices, you can inspect the status of each core individually by using the ICEPICK. Check the short clip **Using ICEPICK** at the Quick Tips page.

If using Spectrum Digital emulators

Spectrum Digital contains a great troubleshooting page that covers the SDConfig usage at:

> http://support.spectrumdigital.com/guides/JTAG_Emulator_guide/
> http://support.spectrumdigital.com/guides/troubleshooting_guide/
> **Note:** Do not use Spectrum Digital utilities to debug XDS100 emulators

If using BlackHawk emulators

Blackhawk has a good troubleshooting guide and a FAQ at:

> http://www.blackhawk-dsp.com/support/get.aspx?name=EMULATOR&type=appnote&subtype=TROUBLESHOOT
> http://www.blackhawk-dsp.com/support/FAQ.aspx
> **Note:** Do not use Blackhawk utilities to debug XDS100 emulators

Check your hardware JTAG connection

1. Check your schematic to see whether the JTAG header is correctly connected on the PCB. Details can be found on the JTAG Connectors page.

> If you are using one of the newer boards that do not have the JTAG connector populated (BeagleBone, AM3359 ICE, AM3358 SK, etc.), keep in mind they usually need additional hardware changes other than simply populating the JTAG header. An example for the Beaglebone can be found here (http://circuitco.com/support/index.php?title=BeagleBone#Optional_JTAG_Header).

2. Is the voltage the same on all your JTAG pins?

3. Does your emulator support the target I/O voltage? There are some older products which cannot handle newer targets with 1.8V I/O as they were designed to operate with 3.3V and 5V targets. Contact your emulator manufacturer for details.

4. Review the Emulation FAQ

5. If experiencing sudden target disconnects, see Emulation Connect/Disconnect for details.

6. Check for Electro-Magnetic Interference (EMI). If you are working with electronics that are near motors or other electrical components that could induce currents, then shielding and isolation may be important. There is an isolation adapter here: [1] (https://estore.ti.com/Emulators-C22.aspx). This typically manifests itself as the connection to CCS becoming unstable or intermittent. This would mean the connection could randomly drop out.

Check your software configuration

1. Check the JTAG clock settings. Due to PCB and device design the speed of the connection may have to be limited to guarantee reliability.

Check the short clip **Changing the Target Configuration Properties** at the Quick Tips page
If you are using a target with an ARM9 processor, you should review the section on Adaptive Clocking.
If you are using a XDS560 or XDS560v2 emulator you can configure its clock settings. Check their pages at XDS560 and XDS560v2 System Trace.
If you are using a Stellaris device with a XDS emulator, please check the page Stellaris Emulator Compatibility
If you are using a TMS570 development board, please check the following two threads:

How to properly set the speed when using a Spectrum Digital XDS510USB emulator (http://e2e.ti.com/support/microcontrollers/hercules/f/312/t/169975.aspx)
How to properly correct a bug on the Debug Access Port designator (all emulators) (http://e2e.ti.com/support/development_tools/code_composer_studio/f/81/p/161988/596913.aspx#596913)

2. DSK/EVM specfic files, such as GEL files used in CCSetup or the target configuration file (.ccxml), are not always provided by TI and are generally not supplied by the emulator vendor. They are usually provided by the board manufacturer. These files also need to be installed and usually accompany the DSK/EVM on CD, labeled something like "EVM target content". Please check with your board manufacturer.

3. Add GEL files to the emulator setup for each CPU if not defined. If you change emulators, having the GEL files specified in a configuration file can cause problems because of a mismatch in emulator and board manufacturer. Unless the emulator vendor is the same vendor that built the target hardware, they are not usually included or defined with driver updates. You can also email your emulator provider for the imports if they are not present.

4. Install the latest emulator drivers for the revision of CCS that you are using. You can check the emulator 3rd party site for the latest drivers (CCSv4 and CCSv5 have options to install device drivers for XDS100, MSP430, ICDI, Blackhawk and Spectrum Digital). As TI releases new devices and DSK/EVM target boards you will see driver updates that include these imports either in the service release, chip support package. Most of the time, a new emulator driver is not needed, just a new CCS import configuration file (.ccs) or target configuration file (.ccxml).

5. If using CCSv4 and CCSv5 you may want to see Troubleshooting CCS topic to inspect IDE and Debugger issues.

6. Update CCS to the latest release. Support for certain devices require CCS to be updated to the correct level.

7. For CCS v3.3 issues related to emulation drivers and service releases, you may want to see the CCS 3.3#Third_Party_Emulation_Drivers topic.

Location of files and utilities

- The configuration file <ccBoard0.dat> is located inside a directory named *.TI* in the user area. Typically it is located at:

  Windows XP systems:
  C:\Documents and Settings\<username>\Local Settings\Application Data\.TI<

  Windows Vista and 7 systems:
  C:\Users\<username>\AppData\Local\.TI

  Linux systems:
  /home/<username>/.TI

  Change the <username> above to the name of your user.
  To help locating the configuration file <ccBoard0.dat> you can use the OS command-line utilities to scan through the TI temporary directories.

On Windows: To find the file, simply open a command prompt, change to the directory and search for the <ccBoardo.dat> file.

```
C:\> cd C:\Users\user\AppData\Local\.TI
C:\Users\user\AppData\Local\.TI> dir ccBoard0.dat /s
Volume in drive C is USER
Volume Serial Number is ABCD-EFGH

Directory of C:\Users\user\AppData\Local\.TI\2079738214\0\0\BrdDat

12/10/2011  11:10             2.198 ccBoard0.dat
               1 File(s)          2.198 bytes
```

Note: if you have multiple installs of CCS in your system or use CCS Cloud, there will be multiple files found. CCSv5.x and prior will be

On Linux To find the file, simply open a terminal, change to the directory and search for the <ccBoardo.dat> file.

user@host:~$ cd .TI
user@host:~/.TI$ find * -name ccBoard0.dat
2079738214/0/0/BrdDat/ccBoard0.dat
user@host:~/.TI$
Note: if you have multiple installs of CCS in your system or use CCS Cloud, there will be multiple files found.

DBGJTAG GUI

The DBGJTAG utility has a GUI that can be downloaded from the page below:

DBGJTAG_Graphical_User_Interface

Code Composer Studio v4

1.If using CCSv4.1.2 or older:

- The <dbgjtag.exe> utility is typically located at:

  <CCS_INSTALL_DIR>\ccsv4\common\uscif\dbgjtag.exe

- The configuration file <ccBoard0.dat> is located at:

supplied with CCS - you just have to be sure to install their support during CCS install. However, if you use a JTAG debugger from a different vendor, you will have to reference their documentation to make sure the driver is properly installed.

> For XDS100, please check section 2.5 (troubleshooting) of the XDS100 page.
> For XDS200, please check section 8 (troubleshooting) of the XDS200 page.
> For XDS560v2, please check section 5.3 (troubleshooting) of the XDS560v2_System_Trace page

3. Determine whether your JTAG connection is working at the lowest level. See section 4 below for details on how to use the *Test connection* (CCSv5 and newer) or DBGJTAG (older releases) to confirm that the JTAG connection is good (or other 3rd party utility such as SDCONFIG) If this fails or has errors, then the problem is related to the software communicating to the emulator. This could be because there is a hardware problem, the configuration in CCS is setup incorrectly, or the software is not installed correctly.

4. Check your Code Composer configuration and launch the debugger manually instead of using the *Debug Active Project* button. Check the following videos:

- The quick tip **Easily launch the debugger without a project** at youtube:

- The short clips **Launching a project target configuration manually** or **Launching a shared target configuration manually** at the Quick Tips page.

The reason is to perform a connection step-by-step and precisely know where the issues are happening:

- If the issue happens during the Debugger Launch phase, check the Troubleshooting CCS page.
- If the issue happens during connect phase, keep reading.
- If the issue happens during loading, the most common issue is trouble writing to memory. This can be caused by a GEL file not properly initializing the EMIF peripheral - keep reading. Another cause can be due to a mismatch between the memory configuration of the application and the physical memory - in this case the application needs to be fixed.
- If the issue happens while running to main(), the problem is with the application code (bad initialization of the device, watchdog timers not being refreshed, invalid memory accesses, etc.) - in this case the application needs to be fixed.

<CCS_INSTALL_DIR>\ccsv4\DebugServer\bin\win32\BrdDat\ccBoard0.dat

Therefore a typical command would be issued as:
"C:\Program Files\Texas Instruments\ccsv4\common\uscif\dbgjtag.exe" -f "C:\Program Files\Texas Instruments\ccsv4\DebugServer\bin\win32\BrdDat\ccBoard0.dat" -rv -S pathlength

Notes:

Change the path to the location appropriate to your installation.
To help locating the configuration file <ccBoard0.dat> check the page Locating the board configuration

2.If using CCSv4.1.3 and newer:

- The <dbgjtag.exe> utility is located in the same place as above
- The configuration file <ccBoard0.dat> is inside a directory named *.TI* in the user area. Typically it is located at:

Windows XP systems
C:\Documents and Settings\<username>\Local Settings\Application Data\.TI<

Windows Vista and 7 systems:
C:\Users\<username>\AppData\Local\.TI

Notes This change is required to accomodate the requirements of Windows Vista and 7 that prevent access to sensitive areas of the system.

Code Composer Studio v3

- The <dbgjtag.exe> utility is typically located at:

<CCS_INSTALL_DIR>\cc\bin\dbgjtag.exe

- The configuration file is located at:

<CCS_INSTALL_DIR>\cc\bin\brddat\ccbrd0.dat

Therefore a typical command would be issued as:
C:\CCStudio_v3.3\cc\bin\dbgjtag.exe -f brddat\ccbrd0.dat -rv -S pathlength

Useful tests

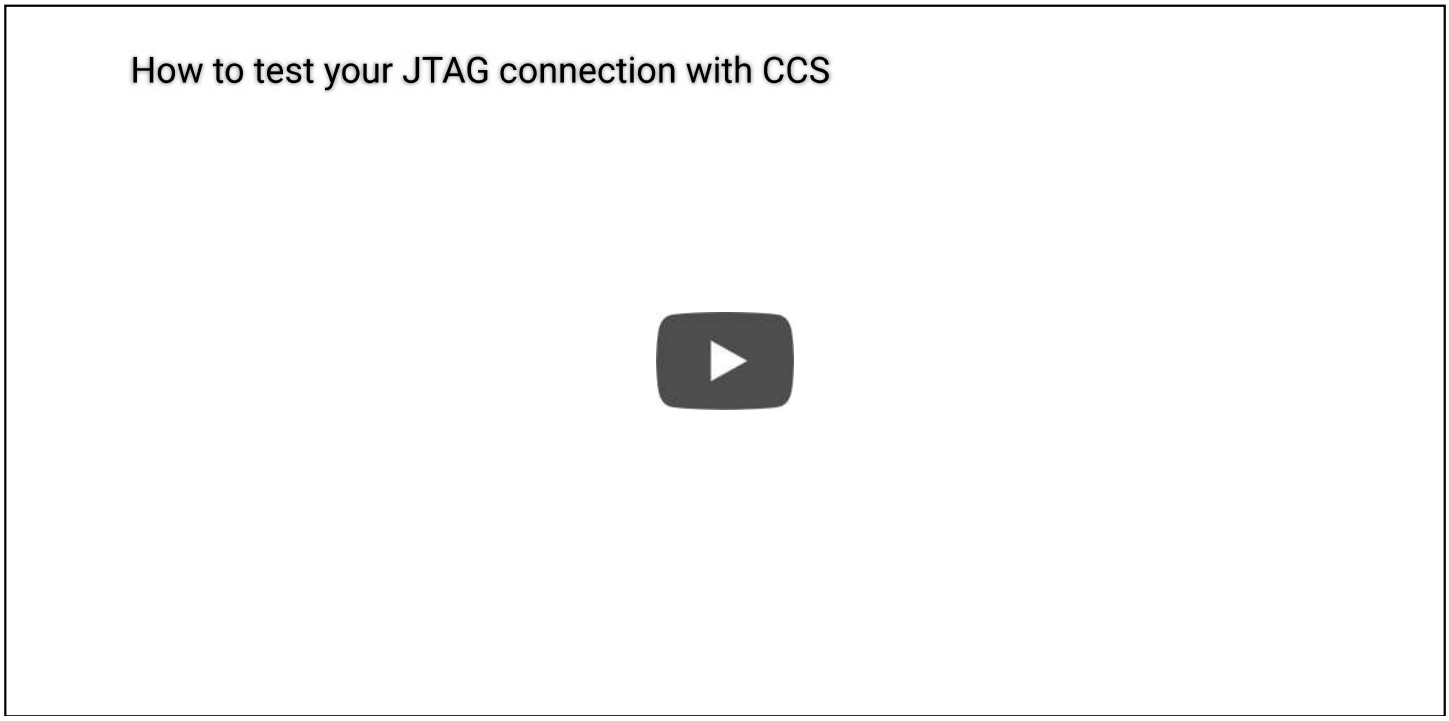
Emulator not plugged in

- The purpose of this test is to verify that the emulator is plugged in propery to the PC and is recognized by the software. For an illustration of a physical set-up, see the picture on the side.

Code Composer Studio v5 and greater

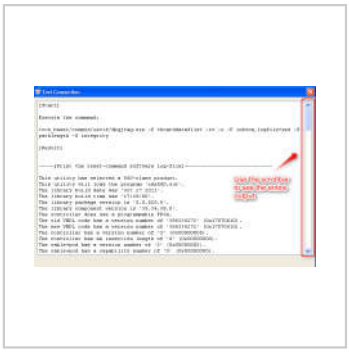
Code Composer Studio v5 (and greater) features a useful *Test Connection* button at the target configuration editor that automatically executes various low-level JTAG tests on the configured XDS based connection. This turns it unnecessary to open a Command Prompt or Shell and look for the utility and the <ccBoardo.dat> files as in the previous versions, however if you need to perform specific tests, the location of the utilities is still provided.

To get this button simply create and save a target configuration (.ccxml) in the target configuration editor and click on the *Test Connection* button.



- **Note:** This button does NOT work for Spectrum Digital XDS510 emulators and all non-XDS based connections (MSP-FET430UIF, Stellaris ICDI, etc)

Test Connection Button

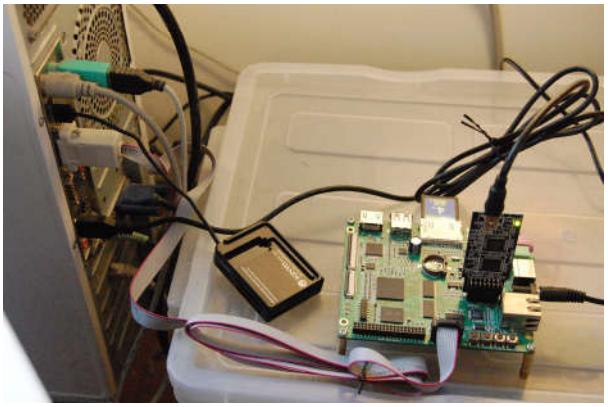


Test Connection in action

- The `dbgjtag` utility is typically located at:

Windows:
CCSv5: <CCS_INSTALL_DIR>\ccsv5\ccs_base\common\uscif\dbgjtag.exe
CCSv6: <CCS_INSTALL_DIR>\ccsv6\ccs_base\common\uscif\dbgjtag.exe
Linux:
CCSv5: <CCS_INSTALL_DIR>/ccsv5/ccs_base/common/uscif/dbgjtag
CCSv6: <CCS_INSTALL_DIR>/ccsv6/ccs_base/common/uscif/dbgjtag

- We will use Dbgjtag to do this test. Dbgjtag can be used with TI emulators and all XDS100 emulators. **IMPORTANT!** All other Spectrum Digital emulators **must** use the utility called "SDconfig", which can perform very similar functions. Please refer to the section If using Spectrum Digital emulators.
- In this example, the emulation SW is unable to access the emulator because it is not plugged in or the operating system drivers have a problem. (Please keep your emulator plugged into the PC for the test.)
- The option "-f brddat\ccbrd0.dat" selects the emulator currently configured in CCS setup.
- The option "-rv" attempts to reset the emulator



```
C:\CCStudio_v3.3\cc\bin>dbgjtag -f brddat\ccbrd0.dat -rv

-------------------------------------------------------------------------

[Print the reset-command software log-file]-------------------------

This utility has selected an XDS560 class product.
This utility will load the program 'bh560usbM.out'.
This utility will operate on port address '0'.
Connect failure

An error occurred while soft opening the controller.


-------------------------------------------------------------------------

[An error has occurred and this utility has aborted]-----------------

This error is generated by TI's USCIF driver.

The value is '-250' (0xffffff06).
The title is 'SC_ERR_ECOM_EMUNAME'.

The explanation is:
An attempt to access the named emulator via USCIF ECOM has failed.
```

Detect the length of the scan chain

- We will use Dbgjtag to do this test
- In this example, we want to figure out the legnth of the JTAG scan chain. The result is that the JTAG instruction register legnth is 38 bits and that the bypass register length is 1 bit.
- The option "-S pathlength" asks the dbgjtag program to find the length of the JTAG path.
- You can check lengths here

```
C:\CCStudio_v3.3\cc\bin>dbgjtag -f brddat\ccbrd0.dat -rv -S pathlength
-------------------------------------------------------------------------
```

```
[Print the reset-command software log-file]---------------------------- This utility has selected an XDS560 class
product.
This utility will load the program 'bh560usbM.out'.
This utility will operate on port address '0'.
The controller does use a programmable FPGA.
The old VHDL code has a version number of '1544' (0x0608).
The new VHDL code has a version number of '1544' (0x0608).
The emulator program is named 'bh560usbM.out'.
The emulator program is version '35.24.0.3'.
The controller has a version number of '4' (0x0004).
The controller has an insertion length of '0' (0x0000).
The cable+pod has a version number of '6' (0x0006).
The cable+pod has a capability number of '9' (0x0009).
The local memory has a base address of '0' (0x000000).
The local memory has a word capacity of '32768' (0x008000).
This utility will now attempt to reset the controller.
This utility has successfully reset the controller.


-------------------------------------------------------------------------------

[Print the reset-command hardware log-file]---------------------------- The scan-path will be reset by toggling the JTAG
TRST signal.
The software is configured to use all Nano-TBC VHDL features.
The controller type is the Nano-TBC VHDL.
The connection type is a 560-class revision-D multi-purpose cable.
The controller will be software reset via its configure register.
The controller will use rising-edge timing on output pins.
The controller may use rising edge timing on input pins.
The controller has a logic ONE on its EMU[0] input pin.
The controller has a logic ONE on its EMU[1] input pin.
The scan-path link-delay has been set to exactly '3' (0x0003).
The support logic has not previously detected a power-loss.


-------------------------------------------------------------------------------

[Perform the standard path-length test on the JTAG IR and DR]----------- This path-length test uses blocks of 512 32-bit
words.

The test for the JTAG IR instruction path-length succeeded.
The JTAG IR instruction path-length is 38 bits.

The test for the JTAG DR bypass path-length succeeded.
The JTAG DR bypass path-length is 1 bits.
```

How to debug whether the JTAG scan path is broken

- In this example, we want to figure whether the JTAG scan path is broken.
- The option "-S brokenpath" asks the dbgjtag program to check the JTAG scan chain for breaks.

```
C:\CCStudio_v3.3\cc\bin>dbgjtag -f brddat\ccbrd0.dat -rv -S brokenpath

[Print the reset-command software log-file]---------------------------- This utility has selected an XDS560 class product.
This utility will load the program 'bh560usbM.out'.
This utility will operate on port address '0'.
The controller does use a programmable FPGA.
The old VHDL code has a version number of '1544' (0x0608).
The new VHDL code has a version number of '1544' (0x0608).
The emulator program is named 'bh560usbM.out'.
The emulator program is version '35.24.0.3'.
The controller has a version number of '4' (0x0004).
The controller has an insertion length of '0' (0x0000).
```

```
The cable+pod has a version number of '6' (0x0006).
The cable+pod has a capability number of '9' (0x0009).
The local memory has a base address of '0' (0x000000).
The local memory has a word capacity of '32768' (0x008000).
This utility will now attempt to reset the controller.
This utility has successfully reset the controller.

-------------------------------------------------------------------------------

[Print the reset-command hardware log-file]---------------------------- The scan-path will be reset by toggling the JTAG TRST signal.
The software is configured to use all Nano-TBC VHDL features.
The controller type is the Nano-TBC VHDL.
The connection type is a 560-class revision-D multi-purpose cable.
The controller will be software reset via its configure register.
The controller will use rising-edge timing on output pins.
The controller may use rising edge timing on input pins.
The controller has a logic ONE on its EMU[0] input pin.
The controller has a logic ONE on its EMU[1] input pin.
The scan-path link-delay has been set to exactly '3' (0x0003).
The support logic has not previously detected a power-loss.

-------------------------------------------------------------------------------

[Perform the Broken Path scan-test on the JTAG IR]---------------------- This test will use blocks of 512 32-bit words.
This test will be applied just once.

Do a test using 0xFFFFFFFF.
Scan tests: 1, skipped: 0, failed: 0
Do a test using 0x00000000.
Scan tests: 2, skipped: 0, failed: 0
All of the values were scanned correctly.

The JTAG IR Broken Path scan-test has succeeded.

-------------------------------------------------------------------------------

[Perform the Broken Path scan-test on the JTAG DR]---------------------- This test will use blocks of 512 32-bit words.
This test will be applied just once.

Do a test using 0xFFFFFFFF.
Scan tests: 1, skipped: 0, failed: 0
Do a test using 0x00000000.
Scan tests: 2, skipped: 0, failed: 0
All of the values were scanned correctly.

The JTAG DR Broken Path scan-test has succeeded.
```

- The first run shows that the test succeeded, so the scan chain is OK.
- You can make the test run repetitively if you choose the "-S brokenpath,repeat=12" option to make it run 12 times. if repeat=0, then the test will run forever.

```
C:\CCStudio_v3.3\cc\bin>dbgjtag -f brddat\ccbrd0.dat -rv -S brokenpath

-------------------------------------------------------------------------------
```

```
[Print the reset-command software log-file]---------- This utility has selected an XDS560 class product.
This utility will load the program 'bh560usbM.out'.
This utility will operate on port address '0'.
The controller does use a programmable FPGA.
The old VHDL code has a version number of '1544' (0x0608).
The new VHDL code has a version number of '1544' (0x0608).

An error occurred while hard opening the controller.

---

[An error has occurred and this utility has aborted]---------- This error is generated by TI's USCIF driver.

The value is '-183' (0xffffff49).
The title is 'SC_ERR_CTL_CBL_BREAK_FAR'.

The explanation is:
The controller has detected a cable break far-from itself.
The user must connect the cable/pod to the target.
```

- In this example, you can see that the Dbgjtag software has detected that the scan chain is broken. In this case, the emulator is simply not connected to the taget, so it says that the cable is broken.
- In addition, it is important to note that this error is returned by the lowest level of the JTAG scan software layer and may be seen in a pop-up dialog when CCS is invoked. It may also be seen when debug utilities such as xdsreset, xdsprobe, dbgjtag are run from the DOS command prompt, as in the case of this example
- As one of the first steps in the debug startup sequence, the low level scan software layer checks the state of the TDIS pin (pin 4 on the TI 14pin and 20pin header) to determine if the emulator cable is connected to the target board. If the cable is not connected it will generate the SC_ERR_CTL_CBL_BREAK_FAR error.
- If the cable is physically connected to the target and the error is still observed, then see the following to further debug the issue: http://processors.wiki.ti.com/index.php/XDS_Target_Connection_Guide#TDIS_.28Target_Disconnect.29

JTAG scan chain integrity test

- You can test the integrity of the JTAG scan chain.
- Use the "-S integrity" option

```
C:\CCStudio_v3.3\cc\bin>dbgjtag -f brddat\ccbrd0.dat -rv -S integrity

---

[Print the reset-command software log-file]---------- This utility has selected an XDS560 class product.
This utility will load the program 'bh560usbM.out'.
This utility will operate on port address '0'.
The controller does use a programmable FPGA.
The old VHDL code has a version number of '1544' (0x0608).
The new VHDL code has a version number of '1544' (0x0608).
The emulator program is named 'bh560usbM.out'.
The emulator program is version '35.24.0.3'.
The controller has a version number of '4' (0x0004).
The controller has an insertion length of '0' (0x0000).
The cable+pod has a version number of '6' (0x0006).
The cable+pod has a capability number of '9' (0x0009).
The local memory has a base address of '0' (0x000000).
```

The local memory has a word capacity of '32768' (0x008000).
This utility will now attempt to reset the controller.
This utility has successfully reset the controller.

[Print the reset-command hardware log-file]---------- The scan-path will be reset by toggling the JTAG TRST signal.
The software is configured to use all Nano-TBC VHDL features.
The controller type is the Nano-TBC VHDL.
The connection type is a 560-class revision-D multi-purpose cable.
The controller will be software reset via its configure register.
The controller will use rising-edge timing on output pins.
The controller may use rising edge timing on input pins.
The controller has a logic ONE on its EMU[0] input pin.
The controller has a logic ONE on its EMU[1] input pin.
The scan-path link-delay has been set to exactly '3' (0x0003).
The support logic has not previously detected a power-loss.

[Perform the Integrity scan-test on the JTAG IR]---------- This test will use blocks of 512 32-bit words.
This test will be applied just once.

Do a test using 0xFFFFFFFF.
Scan tests: 1, skipped: 0, failed: 0
Do a test using 0x00000000.
Scan tests: 2, skipped: 0, failed: 0
Do a test using 0xFE03E0E2.
Scan tests: 3, skipped: 0, failed: 0
Do a test using 0x01FC1F1D.
Scan tests: 4, skipped: 0, failed: 0
Do a test using 0x5533CCAA.
Scan tests: 5, skipped: 0, failed: 0
Do a test using 0xAACC3355.
Scan tests: 6, skipped: 0, failed: 0
All of the values were scanned correctly.

The JTAG IR Integrity scan-test has succeeded.

[Perform the Integrity scan-test on the JTAG DR]---------- This test will use blocks of 512 32-bit words.
This test will be applied just once.

Do a test using 0xFFFFFFFF.
Scan tests: 1, skipped: 0, failed: 0
Do a test using 0x00000000.
Scan tests: 2, skipped: 0, failed: 0
Do a test using 0xFE03E0E2.
Scan tests: 3, skipped: 0, failed: 0
Do a test using 0x01FC1F1D.
Scan tests: 4, skipped: 0, failed: 0

```
Do a test using 0x5533CCAA.
Scan tests: 5, skipped: 0, failed: 0
Do a test using 0xAACC3355.
Scan tests: 6, skipped: 0, failed: 0
All of the values were scanned correctly.

The JTAG DR Integrity scan-test has succeeded.
```

How can I find out more about emulator error messages?

- You can use dbgjtag to find out more about error messages.
- For example, if I want to find out about error # -121, I could follow the sample below.

```
C:\CCStudio_v3.3\cc\bin>dbgjtag -E single,number=-121

[The explanation of the error number '-121']---------- This error is generated by TI's USCIF driver.

The value is '-121' (0xffffff87).
The title is 'SC_ERR_CMD_HANDLE'.

The explanation is:
A bad controller handle has been given to a function,
either before attempting to open the controller, or after
having opened the controller and ignored its error status.
Valid controller handles are generated when attempts
to open the controller return a clean error status.
```


Categories: Code Composer Studio v3 | Code Composer Studio v4 | Code Composer Studio v5 | Code Composer Studio v6 | DaVinci Debugging | Emulation | JTAG

Navigation menu

Personal tools

- Log in
- Request account

Namespaces

- Page
- Discussion

Variants

Views

- Read
- View source
- View history

More

Search

Search Texas Instruments Search Go

Navigation

- Main Page
- All pages
- All categories
- Recent changes
- Random page
- Help

Toolbox

- What links here
- Related changes
- Special pages
- Permanent link
- Page information