Application Note

C2000™ MCU JTAG Connectivity Debug

Matthew Pate, Matt Kukucka, Benjamin Collier, and Nabil Saheb

ABSTRACT

The Joint Test Action Group (JTAG) protocol is a primary means of communicating with a microcontroller (MCU) during product development, emulation, and application debug. All of Texas Instruments (TI) C2000™ devices support JTAG emulation and the C2000 evaluation products, such as controlCARDs and LaunchPad™ Development Kits, incorporate onboard JTAG Emulation. The this application report provides a brief overview of JTAG implementation and explains the steps used to resolve common JTAG connectivity errors when using Code Composer Studio™ software.

If you are having problems using JTAG to communicate with the C2000™ device, TI **strongly recommends** that users follow this debug guide before seeking help through a platform such as TI's E2E™ forum. As part of the screening process, TI support personnel asks about your progress with this guide.

Table of Contents

Trademarks

1 What Is JTAG?

JTAG is named after the group that formed the industry standard for boundary scan testing of printed circuit boards (PCBs). The results of the JTAG effort were later codified in the Institute of Electrical and Electronics Engineers (IEEE) standard: IEEE Std 1149.1. Soon after introduction, the standard became very widely used, which resulted in several additional standards, including the implementation of the on-chip Test Access Port (TAP). This made JTAG the most adapted means for embedded systems development, debug, and testing. JTAG uses a 5-pin implementation in most systems:

- Test Data In (TDI)
- Test Data Out (TDO)
- Test Clock (TCK)
- Test Mode Select (TMS)
- Test Reset (TRSTn)

The necessary biasing of these pins on custom board designs is found in the device-specific data sheet or from reference designs from TI.

2 Common JTAG Debug Probes

Table 2-1 lists some common debug probes for the C2000 ecosystem. For the majority of C2000 devices, it is recommended to use the newer, lower cost XDS110 (www.ti.com/tool/TMDSEMU110-U).

Table 2-1. Common JTAG Debug Probes Used With C2000™ MCUs

	XDS100v1 and XDS100v2	XDS110	XDS200	XDS560v2
Price/Speed	+	+	++	+++
Features	• USB Interface • Flash Programming • Built in Debug Probe on many older C2000 EVMs	• Successor to the XDS100 • USB 2.0 Interface • Flash Programming • Isolated debug (XDS110ISO) • Built in Debug Probe for many new TI C2000 EVMs	• USB 2.0 High Speed Interface (480Mbps) • Flash Programming • Core and System Trace	• USB Interface • Flash Programming • Code Trace Options (high-bandwidth)
Manufacturers	• Texas Instruments • Blackhawk	• Texas Instruments	• Blackhawk	• Texas Instruments • Blackhawk

See also *C2000™ real-time microcontrollers design and development*.

3 Debug Steps for LaunchPad™ Development Kits and controlCARDs

Although all LaunchPad™ Development Kits and controlCARDs are slightly different, there are a few common things to check when users cannot connect to the LaunchPad or controlCARD.

3.1 LaunchPad™ Development Kits

There are a few common reasons that a user is unable to connect to a LaunchPad. To determine which of these steps are relevant, follow the instructions in *How to test your JTAG connection with CCS* to *Test Connection* to the LaunchPad. Before using the following steps, disconnect any BoosterPack™ Plug-in Modules or other external hardware from the LaunchPad.

1. If the log from the *Test Connection* indicates that the *Test Connection* cannot communicate with the debug probe, try the following steps:
   a. Change the USB cable and make sure that the correct debug probe is selected in the *Target Configuration* file.
   b. Check the hardware switches and jumpers on the LaunchPad to see if any can disable the JTAG connection. This information is in the LaunchPad User's Guide, which is usually linked on the TI.com store page for the LaunchPad.
   c. If the *Test Connection* log still shows that the *Test Connection* cannot communicate with the debug probe, there is a possibility that the LaunchPad has been damaged. This is especially likely if the debug probe is not visible in the device manager of the PC.
2. If the log from *Test Connection* mentions that the *Test Connection* cannot circulate bits or that the IR or DR paths are broken, try the following steps:
   a. Check the hardware switches on the LaunchPad to see if any disable the JTAG connection. This information is in the LaunchPad User's Guide, which is usually linked on the TI.com store page for the LaunchPad.
   b. Some LaunchPads are designed to work only with 2-pin cJTAG or only with 4-pin JTAG. This is set in the *Advanced* tab of the target configuration. Make sure to set this correctly.
   c. If the *Test Connection* log still indicates that the *Test Connection* cannot circulate bits, there is a possibility that the LaunchPad was damaged.
3. If the log from *Test Connection* denotes that the test passed, but code still cannot be loaded to the device, try the following steps:
   a. Use the switches on the LaunchPad to put the device in wait boot mode. If you are then able to connect by *following the manual launch* steps, then there was likely something in the previously flashed code that was preventing the connection.
   b. If a connection is still not possible, or the connection worked but code cannot be loaded, it is possible that the device is locked.

3.2 controlCARDs

Similarly to Launchpad Development Kits, there are a few common reasons that the connection to a controlCARD does not work. To determine which of these steps are relevant, see the *How to test your JTAG connection with CCS* link for *Test Connection* instructions. Before using the following the steps, disconnect any external hardware from the controlCARD.

1. If the log from *Test Connection* mentions that the host PC cannot communicate with the debug probe, try the following steps when using the onboard debug probe:
   a. Try changing the USB cable and make sure that the correct debug probe is selected in the *Target Configuration* file.
   b. Check the hardware switches on the controlCARD which select if JTAG signals are coming from the onboard debug probe or the docking station. This information is in the controlCARD User's Guide, which is usually linked on the TI.com store page for the controlCARD.
   c. If the *Test Connection* log still indicates that communication with the debug probe is not established, there is a possibility that the controlCARD is damaged.

2. If the *Test Connection* log indicates that communication with the debug probe is still not established, try the following steps if you are using a standalone debug probe:
   a. Try changing the USB cable to the debug probe.
   b. If the *Test Connection* log still indicates that communication with the debug probe is not established, there is a possibility that the debug probe is damaged.
3. If the log from the *Test Connection* indicates that the debug probe cannot circulate bits or that the IR or DR paths are broken, try the following steps:
   a. Check the hardware switches on the controlCARD to see if any disable the JTAG connection. This information is in the controlCARD User's Guide, which is usually linked on the TI.com store page for the controlCARD.
   b. If the *Test Connection* log still denotes that the debug probe cannot circulate bits, there is a possibility that the controlCARD is damaged.
4. If the log from *Test Connection* indicates that the test has passed, but code cannot load to the device, try the following steps:
   a. Use the switches on the controlCARD to put the device in wait boot mode. If the connection is successful using the *Manual Launch* instructions, then the error was likely something in the flashed code that was preventing the connection.
   b. If the connection still cannot be established, or if the connection succeeds but code cannot be loaded, the it is possible that the device is locked.

4 Common Error Codes

This section lists several common error codes that Code Composer Studio shows during a *Test Connection* operation, an attempted connection, or an attempt to load code to the device. **If the problem is not solved by the corresponding debug steps, search TI's E2E™ forum for any other threads with the same error code.**

See the *How to test your JTAG connection with CCS* link for *Test Connection* instructions, and see also the *Manual Launch* instructions to manually connect to the device and load code.

Use the following flow when debugging error codes:

1. *Test Connection* to verify that the JTAG signals are connected correctly
2. Connect to the device using the previous instructions for a manual launch
3. Load code to the device

4.1 Common Error Codes

Table 4-1 lists the common error codes and the associated debug steps.

Table 4-1. Common Error Codes Table

Error Message	Debug Steps

`This error is generated by TI's USCIF driver or utilities.`
`The value is '-233' (0xffffff17).`
`The title is 'SC_ERR_PATH_BROKEN'.`
`The explanation is: The JTAG IR and DR scan-paths cannot circulate bits, they may be broken.`
`An attempt to scan the JTAG scan-path has failed.`
`The target's JTAG scan-path appears to be broken with a stuck-at-ones or stuck-at-zero fault.`
	This error is usually reported when the JTAG signals are not connected correctly, though the error can also be caused by poor signal quality. Verify that the JTAG connections between the target and probe comply with the data sheet's recommendations.
	This also occurs if the debug probe is using a 4-pin JTAG when TDI and TDO are used as general-purpose inputs - outputs (GPIO) at runtime.
	This can also occur if pullup or pulldown resistors are too strong, so try removing them when debugging. Please compare with the data sheet's recommendations.
	This error sometimes occurs if the device is not booting correctly. Watch the power rails and XRSn with an oscilloscope to make sure that the device boots correctly and XRSn goes high. XRSn periodically rebooting is expected on unprogrammed devices due to the watchdog.
	See the buffered case section of the hardware design guide.

`Error connecting to the target: (Error -1015 @ 0x0) Device is not responding to the request. Device may be locked, or the debug probe connection may be unreliable. Unlock the device if possible (e.g. use wait in reset mode, and power-cycle the board.) If error persists, confirm configuration and/or try more reliable JTAG settings (e.g. lower TCLK).`	1. Verify that *Test Connection* in the target configuration passes. If the connection fails, follow the steps for that error code. 2. Set device to wait-boot mode. 3. Follow the *Manual Launch* instructions and connect to the device. 4. Verify that you are able to read PARTID in the memory browser. 5. Try again to program the device. 6. If applying these steps still fails to clear the error, check the following: Are there password locations on the device? On-chip flash tool settings? Is it possible to program RAM only? Is XRSn properly asserting?

Table 4-1. Common Error Codes Table (continued)

Error Message	Debug Steps
Error connecting to the target:(Error -1135 @ 0x0)The emulator reported an error. Confirm emulator configuration and connections, reset the emulator, and retry the operation.	This error message is usually caused by hardware related issues. Verify that VDDIO/VDD are properly supplied. Confirm that XRSn is asserting high, indicating that the device is properly coming out of reset. Long JTAG cables, improper termination, or excessive capacitance can also degrade the JTAG signal integrity as well.
Error connecting to the target: (Error -151 @ 0x0) One of the FTDI driver functions used during the connect returned bad status or an error. The cause may be one or more of: no XDS100 is plugged in, invalid XDS100 serial number, blank XDS100 EEPROM, missing FTDI drivers, faulty USB cable. Use the xds100serial command-line utility in the 'common/uscif' folder to verify the XDS100 can be located. (Emulation package 8.0.903.2)	1. Check the target configuration file to make sure that the correct debug probe is selected. 2. Check to see if debug probe is visible in PC device manager. 3. Try replacing the USB cable, or try a different debug probe to make sure the probe in use is not damaged.
Trouble Reading Register PC: (Error -1156 @ 0x0) Device may be operating in low-power mode. Do you want to bring it out of this mode? Choose 'Yes' to force the device to wake up and retry the operation. Choose 'No' to retry the operation without waking the device.	1. Verify that *Test Connection* in the target configuration passes. If the connection fails, follow the steps for that error code. 2. Set device to wait-boot mode. 3. Follow the *Manual Launch* instructions and connect to the device. 4. Verify that you are able to read PARTID in the memory browser. 5. Try again to program the device. 6. If applying these steps still fails to clear the error, check the following: Are there password locations on the device? On-chip flash tool settings? Is it possible to program RAM only? XRSn is properly asserting?
C28xx_CPU1: Error: (Error -1044 @ 0x0) The debug probe reported an error. Confirm debug probe configuration and connections, reset the debug probe, and retry the operation.	Try power cycling the debug probe without power cycling the target MCU. Try more reliable JTAG settings like lower clock frequency.
The value is '-230' (0xffffff1a). The title is 'SC_ERR_PATH_MEASURE'. The explanation is: The measured lengths of the JTAG IR and DR scan-paths are invalid. This indicates that an error exists in the link-delay or scan-path.	Verify that the intended JTAG/cJTAG mode is configured in the target configuration file (*.ccxml). Note that certain TI EVMs (LaunchPad/controlCARD) can only support 2-pin or 4-pin JTAG depending on the board design. Please refer to the EVM User's Guide to confirm. This error is also typically reported when the JTAG signals have poor signal quality. Verify JTAG connections between the target and probe conform with the data sheet. Try lower TCK frequency and check trace lengths. See the buffered case section of the hardware design guide.
C28xx_CPU1: Trouble Setting Breakpoint with the Action "Continue or Finish Stepping" at 0x83146: (Error -1066 @ 0x83146) Unable to set/clear requested breakpoint. Verify that the breakpoint address is in valid memory. (Emulation package 9.13.0.00201) C28xx_CPU1: Breakpoint Manager: Retrying with a AET breakpoint	For some devices, only one breakpoint is allowed when running code from flash since hardware breakpoints must be used. Check the device data sheet for the number of hardware breakpoints available.
Error Initializing Emulator: (Error -2083 @ 0x0) Unable to communicate with the debug probe. Confirm debug probe configuration and connections, reset the debug probe, and retry the operation.	1. Check the target configuration file to make sure that the correct debug probe is selected. 2. Check to see if debug probe is visible in the PC device manager. 3. Try replacing the USB cable, or try a different debug probe to make sure the probe in use is not damaged.

Table 4-1. Common Error Codes Table (continued)

Error Message	Debug Steps
`Error connecting to the target: (Error -2131 @ 0x0) Unable to access the device register. Reset the device, and retry the operation. If error persists, confirm configuration, pwer-cycle the board, and/or try more reliable JTAG settings (e.g. lower TCLK).`	1. Verify that the intended JTAG/cJTAG mode is configured in the target configuration file (*.ccxml). Note that certain TI EVMs (LaunchPad/controlCARD) can only support 2-pin or 4-pin JTAG depending on the board design. Please refer to the EVM User's Guide to confirm. 2. Verify that *Test Connection* in the target configuration passes. If the connection fails, follow the steps for that error code. 3. Set device to wait-boot mode. 4. Follow the *Manual Launch* instructions and connect to the device. 5. Verify that you are able to read PARTID in the memory browser. 6. Try again to program the device. 7. If applying these steps still fails to clear the error, check the following: Are there password locations on the device? On-chip flash tool settings? Is it possible to program RAM only? XRSn is properly asserting?

`This error is generated by TI's USCIF driver or utilities.`
`The value is '-242' (0xffffff0e).`
`The title is 'SC_ERR_ROUTER_ACCESS_SUBPATH'.`
`The explanation is: A router subpath could not be accessed.`
`The board configuration file is probably incorrect.`
	1. Check the target configuration file to make sure that the correct debug probe is selected.
	2. Check to see if the debug probe is visible in the PC device manager.
	3. Try replacing the USB cable, or try a different debug probe to make sure the probe in use is not damaged.

`Error connecting to the target: (Error -267 @ 0x0) The controller could not detect valid target supply. JTAG connection and/or connection setting specifying voltage level.`	This error message only happens if the VTREF pin on the debug probe is not connected to 3.3V. Make sure that the target board is powered on and XRSn is properly asserting.
`C28xx: Failed CPU Reset: (Error -1137 @ 0x0) Device is held in reset. Take the device out of reset, and retry the operation.`	1. Verify that *Test Connection* in the target configuration passes. If the connection fails, follow the steps for that error code. 2. Set device to wait-boot mode. 3. Follow the *Manual Launch* instructions and connect to the device. 4. Verify that you are able to read PARTID in the memory browser. 5. Try again to program the device. 6. If applying these steps still fails to clear the error, check the following: Are there password locations on the device? On-chip flash tool settings? Is it possible to program RAM only? XRSn is properly asserting?
`The JTAG IR Integrity scan-test has failed.` `The JTAG DR Integrity scan-test has failed.`	This error is reported when the JTAG signals have poor signal quality. Try lower TCK frequency and check trace lengths. See the buffered case section of the hardware design guide.

5 Multiple Devices in JTAG Chain

The JTAG standard allows for a single JTAG debug probe to make a daisy-chain connection with multiple devices. In practice, various constraints limit the number of target devices that can be connected in a chain. The XDS-class debuggers all have a limited number of instruction register (IR) bits that the debug probe can circulate before expecting to see the bits returned. Many older C2000 devices require 38 IR bits to be circulated per device, while all newer devices only require six IR bits to be circulated because of the ICEPick JTAG route controller. At the time of writing this application note, the following devices have ICEPicks: F2807x, F28M3x, F2837xD, F2837xS, F28004x, F2838x, F28002x, F28003x, F280013x, F280015x, F28P65x, and F28p55x. To determine if the device has an ICEPick, use Code Composer Studio to look at the *Advanced* tab of the target configuration, as shown in Figure 5-1.

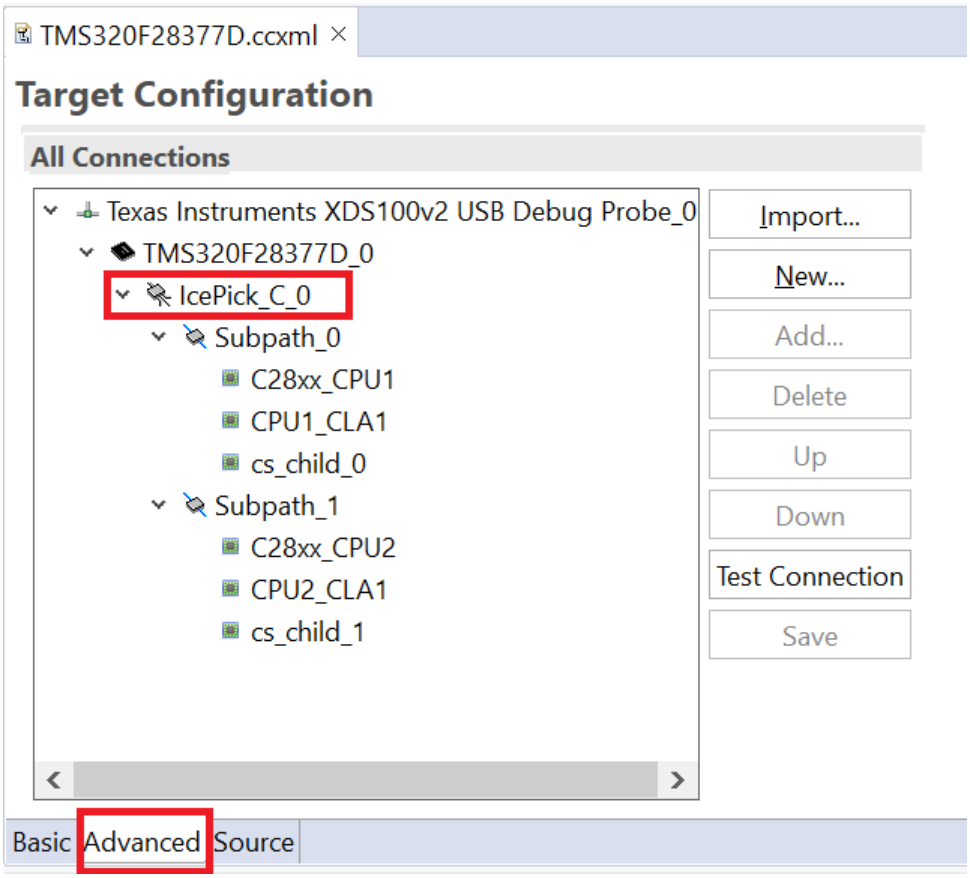

Figure 5-1. Target Config Advanced View

For devices without an ICEPick, this means that the XDS100, XDS110, and XDS200 debug probes can only reliably connect up to two devices in a chain. However, up to 12 devices can be connected with an ICEPick in a JTAG chain. The XDS560 can reliably connect to three devices without an ICEPick, and up to 18 devices with an ICEPick. See also the *TI ICEPick Module Type C Reference Guide*.

6 Non-Intrusive Debug

Users can non-intrusively connect to view the device's real-time state without interrupting execution. The following steps need to be followed to prevent a reset from being triggered on target connect:

Note

The following steps were used on a F28P55x device, but can be applied to any F28x device.

1. Open the device-specific GEL file located in the CCS installation folder at: *ccs\ccs_base\emulation\gel*. For this example, F28P55x devices use "f28p55x.gel".
   a. Depending on the device family, GEL file names can also correspond to the orderable part number (ex. "f2800157.gel").
2. In the OnTargetConnect() function, comment out any GEL_Reset() and RAM initialization lines. Save the file.

```
OnTargetConnect()
{

//    *(int *)(MEMCFG_BASE + MEMCFG_O_DXINIT)  = 0x00FF;              /* RAM INIT FOR M0/M1 Memory      */
//    while(!(*(int *) (MEMCFG_BASE + MEMCFG_O_DXINITDONE) == 0xFF));   /* Wait for InitDone Status       */

//    *(int *)(MEMCFG_BASE + MEMCFG_O_LSXINIT) = 0x00FF;              /* RAM INIT FOR LS1..LS7  Memory  */
//    while(!(*(int *) (MEMCFG_BASE + MEMCFG_O_LSXINITDONE) == 0xFF)); /* Wait for InitDone Status       */

//    *(int *)(MEMCFG_BASE + MEMCFG_O_GSXINIT) = 0x000F;              /* RAM INIT FOR GS0.GS3 Memory    */
//    while(!(*(int *) (MEMCFG_BASE + MEMCFG_O_GSXINITDONE) == 0xF));  /* Wait for InitDone Status       */

//    GEL_TextOut("\n\nRAM initialization done\n\n");

    if (GEL_IsInRealtimeMode()) /* If in real-time-mode */
    {
    }
    else /* Put device into C28x Mode */
    {
        C28x_Mode();
    }

    f28p55_Memory_Map(); /* Initialize the CCS memory map */

    /* Check to see if CCS has been started-up with the DSP already */
    /* running in real-time mode.  The user can add whatever        */
    /* custom initialization stuff they want to each case.          */

    if (GEL_IsInRealtimeMode()) /* Do real-time mode target initialization */
    {

    }
    else /* Do stop-mode target initialization */
    {
        // GEL_Reset(); /* Reset DSP */
    }
}
```

Figure 6-1. GEL OnTargetConnect() lines to be commented out

3. In CCS, navigate to the *Target Configuration* dropdown in the Debug window. Open the intended target configuration file. Verify that the correct debug probe, device, and initialization script (GEL) are configured.

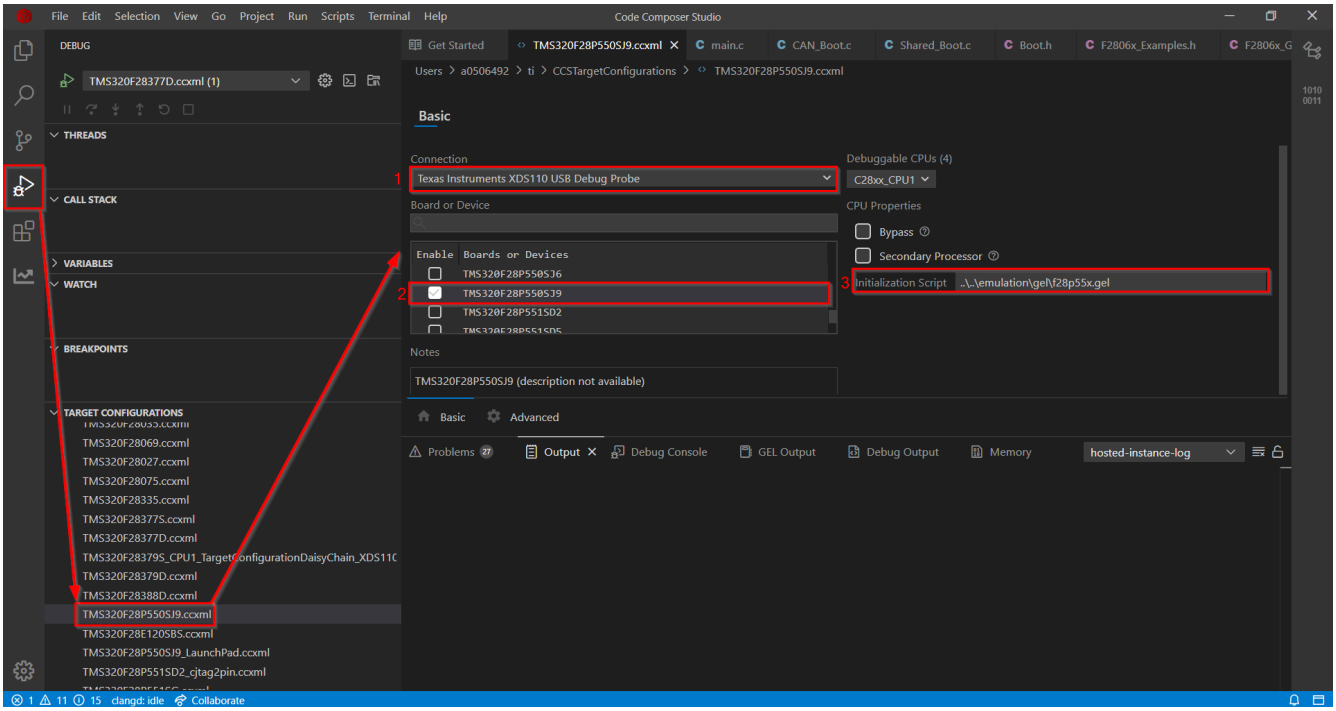

Figure 6-2. Setting up the Target Configuration File

4. Right-click the intended target configuration (ccxml) in the *Target Configuration* dropdown and select *Start Project-less Debug*.
5. Right-click "C28xx_CPU1" (or any CPU core) in the *Threads* dropdown and open the *Properties...* window.
6. Confirm "Halt the target on a connect" and "Reset the target on a connect" are both deselected under *Connection Options* in the Program/Memory Load Options menu (for the intended core). Save and Close the properties once configured.

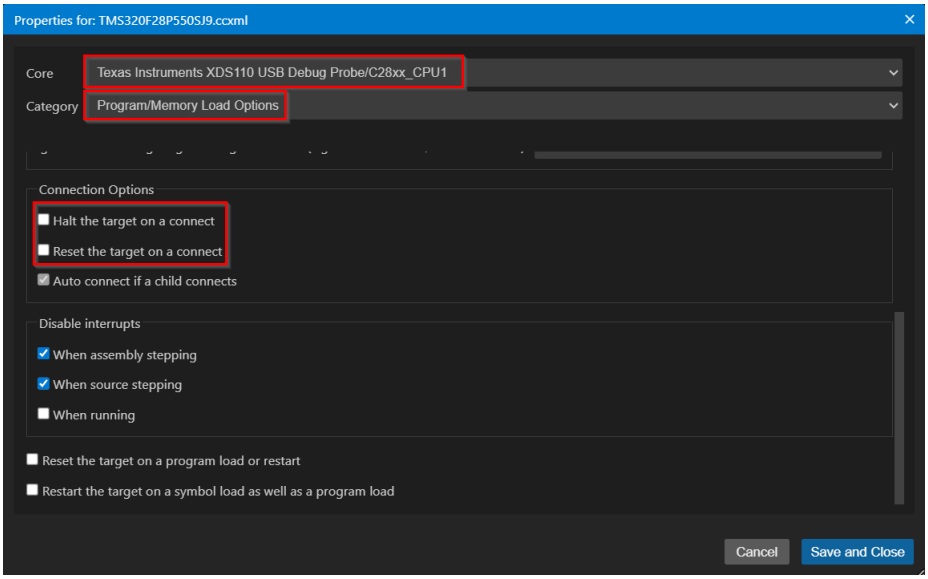

Figure 6-3. Target Configuration Properties

7. Re-launch the Project-less Debug session and connect to the intended CPU core. The connection can now be established non-intrusively.
   a. The program's symbols can also be loaded to observe the current state of the device.

7 Disabling and Resetting the JTAG TAP

Note

The following section is relevant to the following device families:

- F28004x
- F28002x
- F28003x
- F280013x
- F280015x
- F28P55x
- F28P65x
- F28E12x

The JTAG Test Access Port (TAP) is a standardized hardware interface (IEEE 1149.1) used for accessing, testing, and debugging the on-chip logic. The TAP acts as a FSM that shifts data in/out through TDI/TDO to control boundary scan registers for board-level testing, hardware debugging, and system programming.

The TAP_STATUS register reflects the status of the TAP at any given time. Normally when no JTAG emulator is connected to the device, the state is held in the TLR (Test Logic Reset) state. In some cases with excessive PCB noise, there can be unwanted TMS and TCK toggles that take JTAG out of the TLR state. When persistent, this can ultimately lead to unwanted activation of the JTAG Boundary Scan or some other JTAG mode that can directly interfere with the intended application. The TAP can potentially control the GPIOs by bypassing normal application logic through boundary scan (BSCAN) mode, which can ultimately cause the system to hang.

Figure 7-1. TAP_STATUS Register

31	30	29	28	27	26	25	24
DCON	RESERVED						
R-0h	R-0-0h						
23	22	21	20	19	18	17	16
RESERVED							
R-0-0h							
15	14	13	12	11	10	9	8
TAP_STATE							
R-0h							
7	6	5	4	3	2	1	0
TAP_STATE							
R-0h							

Table 7-1. TAP_STATUS Register Field Descriptions

Bit	Field	Type	Reset	Description
31	DCON	R	0h	DebugConnect indication from IcePick. Reset type: PORESETn
30-16	RESERVED	R-0	0h	Reserved

Table 7-1. TAP_STATUS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
15-0	TAP_STATE	R	0h	TAP State Vector. With bits representing, Connect coresponding POTAP* output to the 0x0001:TLR, 0x0002:IDLE, 0x0004:SELECTDR, 0x0008:CAPDR, 0x0010:SHIFTDR, 0x0020:EXIT1DR, 0x0040:PAUSEDR, 0x0080:EXIT2DR, 0x0100:UPDTDR, 0x0200:SLECTIR, 0x0400:CAPIR, 0x0800:SHIFTIR, 0x1000:EXIT1IR, 0x2000:PAUSEIR, 0x4000:EXIT2IR, 0x8000:UPDTIR Reset type: PORESETn

Table 7-2 lists the various BSCAN tests implemented on C2000 devices. These modes can be referenced in the BSDL models provided for each device family on the TI product page. Not all of the BSCAN modes gates GPIO function. The scan sequence 'xxxxxx' represents a binary input on TDI that ultimately places the device in the corresponding BSCAN mode once the final bit is clocked by TCK.

Table 7-2. Boundary Scan Modes

Instruction	TDI Serial Scan	GPIO Impact	System Impact
EXTEST	'b011000	Gates GPIOs (Configurable)	Possible
SAMPLE	'b011011	None	None
BYPASS	'b111111	None	None
HIGHZ	'b011110	Gates GPIOs	All pins disconnected
IDCODE	'b000100	None	None
PRELOAD	'b011011	None	None

To avoid any unintentional BSCAN modes from being entered, place strong enough external pull resistors (particularly on TMS and TCK) on the board to prevent noise from activating JTAG. For maximum reliability, if TDI is unused in the system, change the GPIO direction to be an output and drive low. Since 'b000000 is an unused instruction scan sequence, this won't change the device behavior when scanned in.

Note

On F28P55x and F28P65x devices, the TAP_CONTROL register is provided to disable BSCAN mode entirely by setting the BSCAN_DIS bit. Debugger access through the JTAG is still allowed even if BSCAN mode is disabled.

From a software perspective, the TAP_STATUS register can be polled by the application code to detect device disturbance. The SOFTPRES40[JTAG_nTRST] register can also be used to reset the JTAG TAP through software. Users can detect the TAP transitioning from the TLR to the IDLE state (which doesn't impact the system yet), and reset the TAP back to the TLR state before the device issues an unknown instruction. Please refer to this E2E FAQ for details on how to use SOFTPRES40 to reset the JTAG TAP state, even when the register is undocumented in CCS or the device-specific Technical Reference Manual.

Note

Use the SOFTPRES40 register with caution, as this prevents connecting a debugger unless the code qualifies writes to this register with some other GPIO state or other means to distinguish between noise and debugger accesses. For example, the DCON bit in the TAP_STATUS register can be used to indicate if there is a debugger connected to the device.

8 JTAG Connectivity Debug Flows

The following flowcharts provide step-by-step guidance for isolating and performing common troubleshooting advice to resolve JTAG connectivity issues. If at the end of the flow there are still issues, submit questions to the *TI Engineer to Engineer C2000 Support Forum* for support.

8.1 Overall Debug Flow

How to use these flow charts:

1. Review the steps in Figure 8-1 and go through the High-Voltage Isolation Flow first. This is important even if isolation is not a main concern, since isolation can have an impact on the emulation aspect of the PCB.
2. Follow the Main JTAG Debug flow chart. Once an intermediary step is completed, return to the Main flow and continue if there are remaining issues.
3. If the problem cannot be solved after using all flow elements, submit a question to the TI E2E support forums. A list at the end of Section 9 provides information on what to include in your question for the most efficient response from TI.

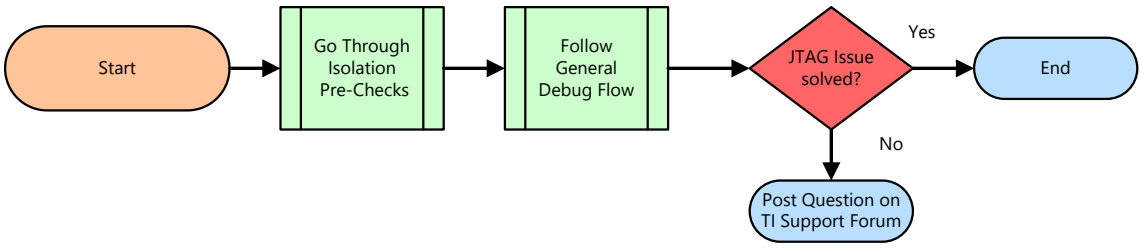

Figure 8-1. Overall Debug Flow

8.2 High-Voltage Isolation Check Flow

Many C2000 applications are high power in nature. Because of this, isolate the power plane of the target board from the host computer while debugging. Many TI manufactured boards have isolated emulation, or can be enabled using onboard options. Figure 8-2 shows the flow chart to help identify whether isolated JTAG is present, and if so, troubleshoot common issues in these systems. There are also standalone debug probes that provide isolation as well; this flow does not pertain to those probes.

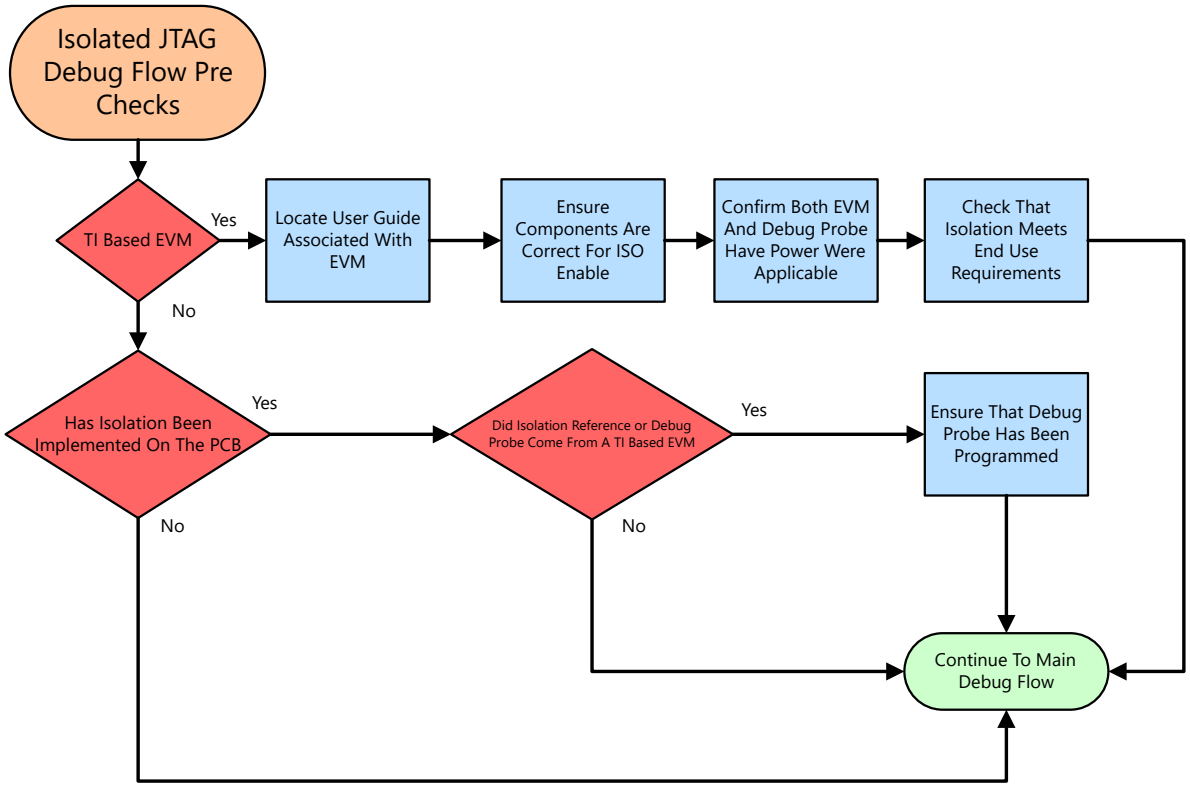

Figure 8-2. JTAG Isolation Pre-Checks

8.3 Main JTAG Debug Flow

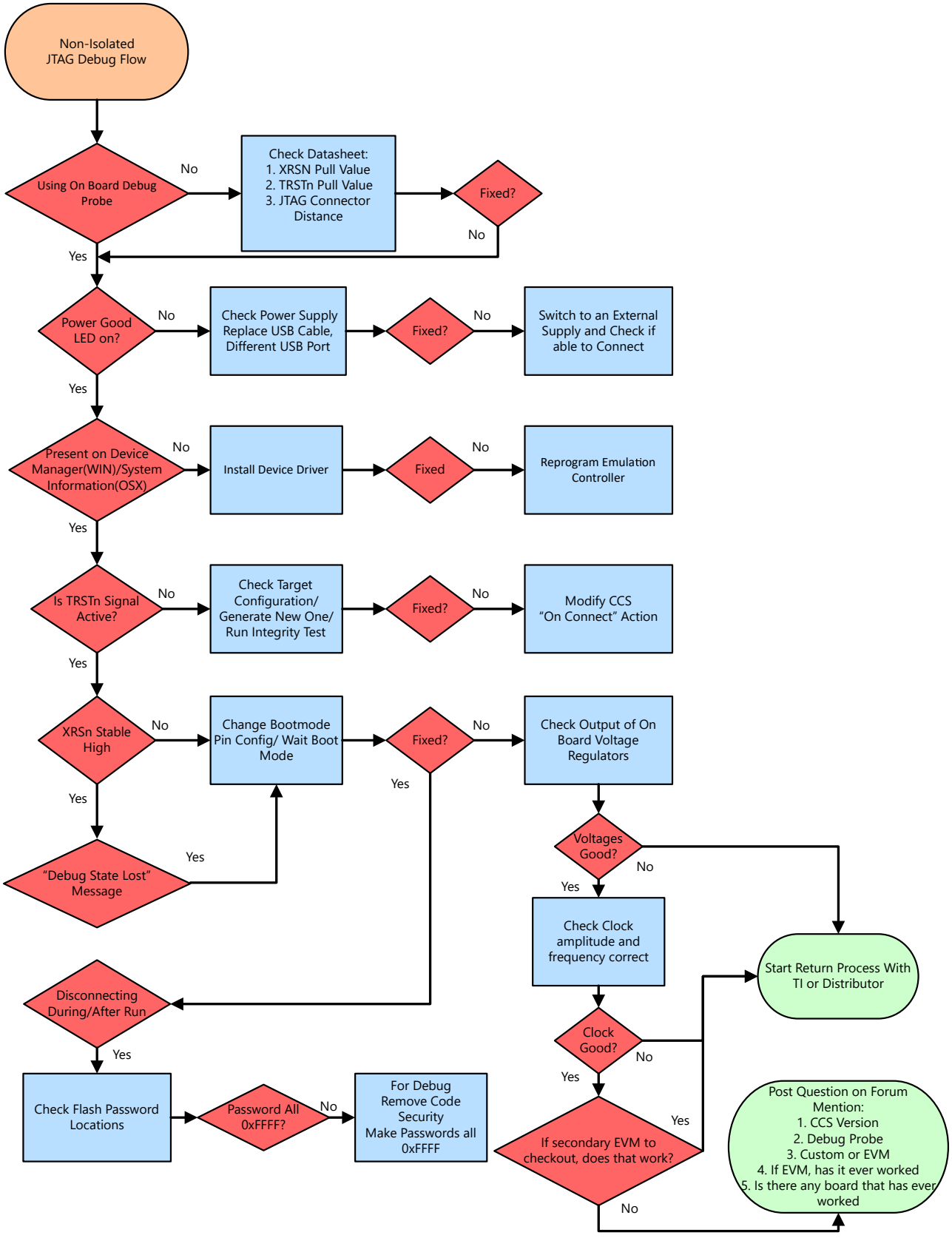

Figure 8-3. JTAG Debug Flow

9 Detailed Flow Step Information

Beginning with the Overall Debug Flow (Figure 8-1), decide if the High Voltage Isolation Check Flow (Figure 8-2) is needed or not. Once complete, go through the Main JTAG Debug Flow (Figure 8-3) step by step.

This is a supplement list to the flow charts. This list provides more background on the directive for each step to help better understand what is to be accomplished.

9.1 Isolation Pre-Check Flow

1. **TI Based EVM:** If the EVM under debug is TI-based, then the flow follows this branch in the flow chart.
2. **Locate User Guide Associated With EVM:** All TI-based EVMs have a User's Guide or Quick Start Guide that detail the features of the EVM as well as components that are critical to the correct operation of the EVM. Reviewing these up front assists in the debug process (explained in this document).
3. **Board Has Isolation Implemented:** Using the reference guide from the previous step, or other documentation, if not a TI-produced EVM, determine if the board under debug has isolation circuitry implemented or not.
4. **Make Sure Components Are Correct For ISO Enable:** Referring back to the documentation for the EVM, make sure that any switches, jumpers, or shunts are correctly populated to achieve the desired isolation state for the EVM.
5. **Confirm Both EVM And Emulation Have Power Where Applicable:** To achieve proper isolation of the power planes between the local and high power domains, isolators are used to connect the two planes and allow the emulation signals to get to the MCU. Since there are separate power planes there must be two paths for powering each plane. Make sure both planes have power to connect to the device through emulation.
6. **Check That Isolation Meets End Use Requirements:** While not an essential check for initial system debug, because it is important to become familiar with the isolation devices that are used such that it is understood if the devices meet the requirements of the end system. While TI EVMs comprehend this to the end application, if there are TI EVMs mixed with custom EVMs, this can still be a necessary check.
7. **Did Isolation\Emulation Reference Come From A TI Based EVM:** Many times the isolation and emulation circuits from a TI EVM are re-used for a custom design. While electrically sound, what is often overlooked is that the emulation chipset still needs to be programmed. For a TI-based EVM, this occurs before the EVM is sold, on a custom board; however, this process still needs to be comprehended in the production flow.

9.2 JTAG Debug Flow

1. **Using Onboard Debug Probe:**
   a. **Yes:** Many C2000 MCU boards have a JTAG debug probe implemented on the PCB. Unless there is an application requirement, TI suggests using the onboard debug probe for development purposes. The XDS100 and the XDS110 are two target debug probes that are found on TI C2000 Evaluation Modules (EVMs).
   b. **No:** If a stand-alone debug probe is used, where the board design is custom, the implementation of the JTAG header and passives need to be verified before continuing the debug process. The device-specific data sheet contains the reference schematic for the proper pullup or pulldown values to provide proper behavior. If the PCB is manufactured by TI, this step can be skipped.
2. **Power Good LED On:** This step is meant to verify that the target is powered correctly from a power source without the use of any external equipment like a voltmeter. All of TI C2000 development boards have LEDs to indicate power is being supplied to the MCU. Other LEDs can be used to indicate some out-of-box code is successfully running. For the location and function of these LEDs, consult the device-specific user's guide for the EVM under debug.

3. **Replace Cables:** If the Power Good LEDs are not observed, there is likely a problem with the power source supplied to the EVM. Many TI EVMs use the USB connection not only to provide a debug path from the host to the target, but also use the 5V from the USB to power the EVM. A simple check can be to change the USB cable to make sure this is not the issue. If there is insufficient power from the host, a powered USB hub can help as well.
4. **Switch to an External Power Supply:** If the onboard power supply of the TI-built board is not providing power at the appropriate level and the USB cables are known to be good, try switching to an external power supply for the EVM. To understand if this is supported, see the device-specific user's guide for your EVM. In this case, some probing of the voltages on the board is necessary to determine if the power supply is the issue or something on the PCB is inhibiting the voltage to the MCU.
5. **Present in Device Manger:** For the JTAG debug probe to communicate to the PC, the driver files need to be installed. This typically occurs co-incident to the installation of the Code Composer Studio (CCS). To verify that the drivers are successfully installed, connect the PC to the JTAG debug probe and power up. Then go to Control Panel → Device Manager (Figure 9-1) and locate the associated debug probe.

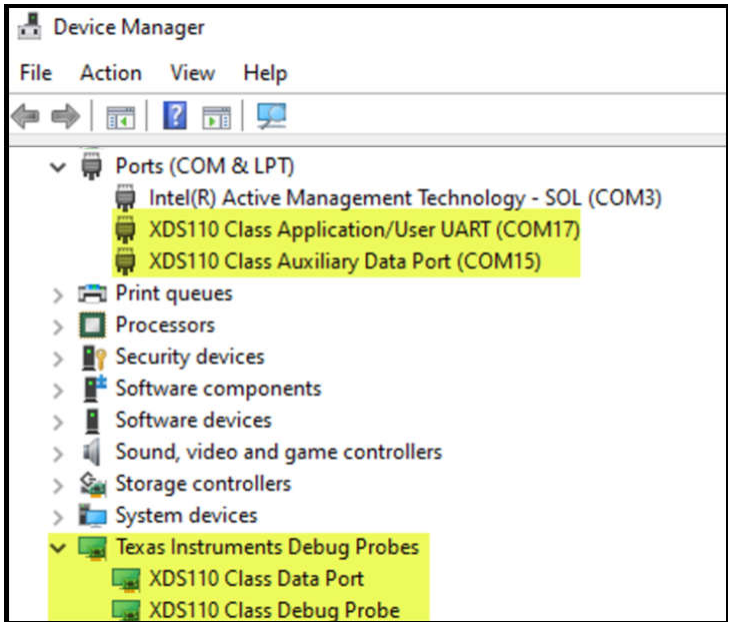

Figure 9-1. Microsoft® Windows® 10 Device Manager Showing Successful Detection of XDS110 Debug Probe

6. **Reprogram Emulation Controller:** This step makes sure that the device that functions as the emulation controller has the correct firmware.
   a. XDS100v1: The host device is the FTDI FT2232 following guide on re-programming
   b. XDS100v2: The host device is the FTDI FT2232following guide on re-programming
   c. XDS110: The host device is a TI MCU TM4C1294NCPDRI3R following guide for re-programming.
7. **Install Device Driver:** Another possible reason for the debug probe not showing up in the host PC/MAC system is the driver is not installed. Typically this occurs with the installation of CCS, but consult the debug probe product page for possible drivers.
8. **Is TRSTn Signal High At the MCU:** This step is checking for a certain behavior when CCS is trying to connect to the target. One of the first actions is that Test Reset (TRSTn) goes inactive high, activating the core debug connection to the external debug probe. If TRSTn does not change state during a CCS Connect Target operation, then the debug probe needs to be checked for proper configuration correctly both at the device level and inside the operating system of the host.

9. **Check Target Configuration:** The Target Configuration File (.ccxml) contains the information necessary to connect to your target device and the JTAG debug probe being used. To view the current target configurations, select Target Configurations (Figure 9-2) under the "View" tab in CCS. Double click on the .ccxml corresponding to the target that is being debugged. If the drivers for the debug probe are installed correctly and the correct options are selected, the *Test Connections* button (Figure 9-3) is available and ready to execute. The datalog from this test can assist in isolating the cause for the connection issues, do not skip this step. Many example projects are installed as part of C2000Ware or controlSuite have a *target configs* folder. This has a .ccxml file pre-created based on a an assumption of a default EVM and debugger. This file is used when the *Debug* icon is used to launch the debug session. If the *Debug* button is the desired method to launch the debug session, the .ccxml in *target configs* need to be modified.

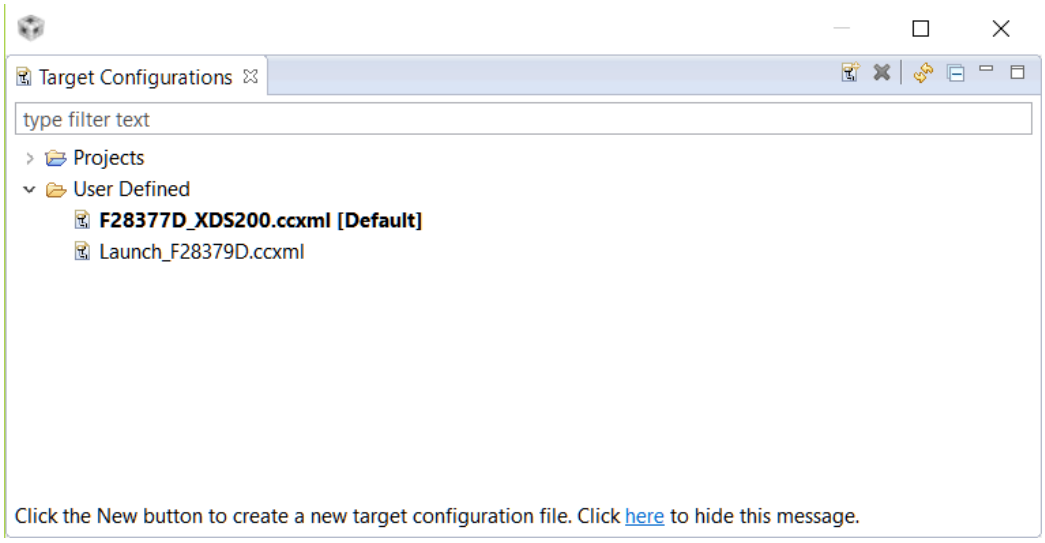

Figure 9-2. Target Configurations View

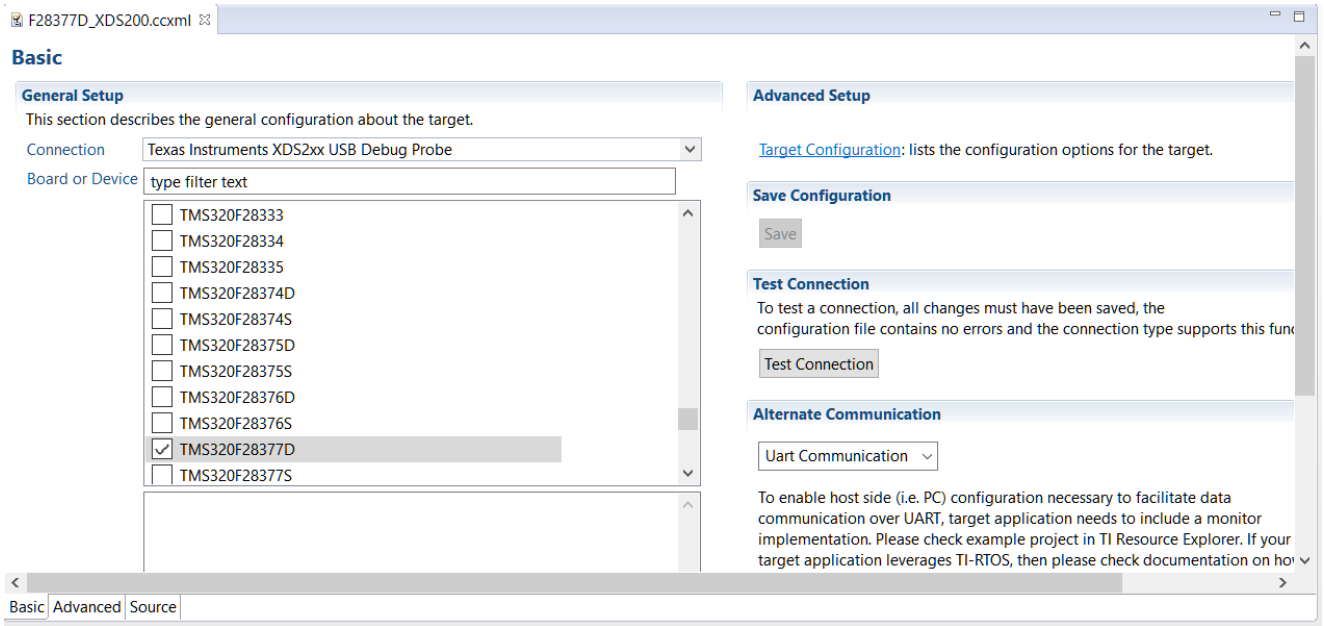

Figure 9-3. Test Connections

10. **Modify the CCS *On Connect* Action:** There are two ways to launch the debug session from CCS. One way is to right click the desired target configuration from the previous step and select *Launch Selected Configuration*. Once this is done the target CPU can be connected by right clicking on the CPU cores and select "Connect Target". The other way is to use the Debug Button (Figure 9-4), which not only launches the configuration, but also connects, loads the target program file into memory, and executes to *main*. These settings can be modified but this is the default operation. The default actions can be modified by either right clicking on the .ccxml file being used or selecting *Debug Options* from the arrow drop-down menu next to the *Debug* button and changing the auto run and launch options in the Target sub-menu. During the troubleshooting phase of this document, TI recommends using the former method of *Connect Target*. This helps to isolate any issues that are not purely JTAG related, but caused by code execution or other system interactions. Once the system is verified to be stable for launching and connecting the target, use the Debug button to handle these steps.

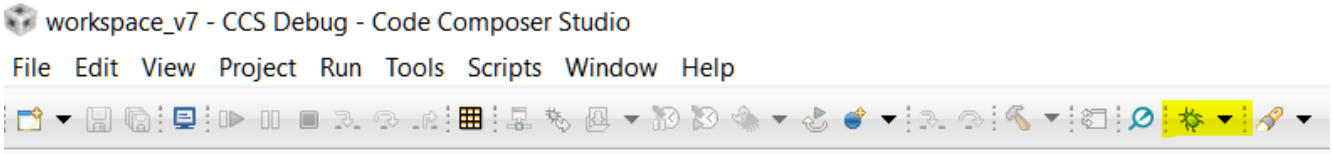

Figure 9-4. Code Composer *Debug* Button

11. **XRSn State:** Looking at XRSn on an oscilloscope, XRSn needs to be inactive high when the device is operational. XRSn low, or pulsing from low to high to low, can indicate one of several issues. If the pulses are periodic, it is likely the watchdog (WD) on the MCU is causing the reset because it is not being serviced or is not disabled. This toggle behavior is not a bad thing, indicating that the MCU is powering up, and executing code, but the behavior can cause instability in the debug flow. If there is non-determinate pulsing or XRSn is always low, this can indicate that the internal Brown Out Reset (BOR) is being triggered due to a supply voltage issue or some issue on the PCB. This is different than the previously-mentioned static supply checks. Both of these potential issues can also happen during code execution. The issues can either disconnect the debug session or prevent it from reliably connecting.
12. **Change BootMode:** Check the hardware files to make sure BootMode pins are in the correct state for the mode expected. If the XRSn pin shows the behavior mentioned above, or if the state of the Flash memory is unknown, going into Wait Boot Mode puts the device into a safe state allowing for reads of memory and registers. For more details on the boot pins and the selection required for Wait Boot Mode, check the *Boot* section of the device-specific data sheet.
13. ***Debug State Lost* CCS Message:** Even if XRSn is in the desired inactive high state, there still can be issues that prevent or end the debug connection. Often this behavior is related to the code that is executing on the device. For this reason, put the device in Wait Boot Mode.
14. **Check VREG Setting:** Any voltage supplied to the device outside the recommended operating conditions can cause a brown out reset (BOR) event to occur. In these situations, measure the voltage rails of the device. Using the schematic files located in either C2000Ware or controlSuite, the probe points for the rails of the device can be verified. If this issue is happening during code execution, there can be an issue with the amount of current the source can provide to the device. If the EVM under debug is a TI manufactured device, any rails generated from the external supply ought to be OK, by design and at this point the checks are being done to verify the board integrity is good.
15. **Check Clocks (JTAG Clock/System Clock):** Measure and confirm the JTAG clock and crystal or external clock source are as per data sheet defined levels. Check the manufacturer's data sheet for the debug probe. This is the final step to make sure that the device is supplied with the inputs required to function correctly. Many Piccolo™ class devices have a built in, zero pin oscillator. Using this as the functional clock can be helpful in case of external clock uncertainty. For the available clock sources and the tolerances, see the device-specific data sheet. While the JTAG clock is typically maintained at the default speed from the initial setup file, slowing down the clock rate can be helpful to see if this improves the initial connection or connection stability. This can be especially helpful on custom designed PCBs.
16. **Second Device Check:** If after all the previous steps do not fix the issue, a second PCB or EVM can possibly be used to determine whether the issue is local to one EVM. If a second device fails in the same manner there is likely either a setup issue or issue external to the EVM at play.

17. **Disconnecting During\After Run:** If a device is password locked, the Emulation Code Security Logic (ECSL) in the Code Security Module (CSM) disables JTAG emulation to the device, resulting in JTAG connection issues. This can occur before connection per above, but can also occur during debug if a secure region of memory is accessed while the debugger is connected. While Wait Boot Mode allows connections, this mode does not correct the issue of access to secure memory while debugging. To correct this, the CSM must be unlocked through use of a known password. For locking and unlocking a device, see the device specific data sheet on the CSM module and the associated steps. If the password is unknown, the device cannot be unlocked. Debug is limited to unsecured regions.
18. **Post question on E2E.ti.com:** If, at the end of this flow, there are still issues connecting or maintaining connection with the device through JTAG, questions or issues can be posted to the TI C2000 Engineer To Engineer Forum. When posting, please provide the following information in addition to documenting the issue:
    a. Subject or Title of the Post: "JTAG Connectivity Issue - (Insert Part Number here)
    b. CCS Version
    c. Debug Probe Used
    d. Type of target: TI made EVM or custom
    e. Confirmation of which steps in this guide were taken
    f. Custom board schematics of the JTAG connection, if possible, if not a TI EVM

10 References

These are specific support pages that can be helpful in either the debug of a JTAG issue or the selection of the appropriate JTAG emulation device for the end system

- TI Guide to Debugging Common JTAG issues
- Texas Instruments, *Getting Started With C2000™ Real-Time Control Microcontrollers (MCUs)*
- C2000 real-time microcontrollers

11 Revision History

Changes from Revision C (August 2024) to Revision D (April 2026)	Page
• Added isolated debug feature for XDS110	2
• Slightly revised format of Common Error Codes section	5
• Added Section 6 section	9
• Added Section 7 section	11

Changes from Revision B (January 2023) to Revision C (August 2024)	Page
• Added Debug Steps for LaunchPad™ Development Kits and controlCARDs section	3
• Added Common Error Codes section	5

Changes from Revision A (January 2022) to Revision B (January 2023)	Page
• Updated the numbering format for tables, figures and cross-references throughout the document	2
• Documenting ability of certain JTAG Debug Probes to support JTAG daisy chains	8

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you fully indemnify TI and its representatives against any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale, TI’s General Quality Guidelines, or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.

Last updated 10/2025