

XDS110 Debug Probe

User's Guide



Literature Number: SPRUI94
January 2017

1	Overview	4
1.1	XDS110 Probe Feature Summary	4
1.2	XDS110 System Summary	4
1.3	XDS110 Performance	5
1.4	XDS110 Probe Overview.....	5
1.5	XDS110 Parts List.....	6
1.6	Acronyms, Abbreviations, and Definitions	6
2	Probe Interfaces	8
2.1	Supported Communication Protocols.....	8
2.2	USB	8
2.3	Debug Interface	8
2.4	Auxiliary Debug Interface	9
2.5	Probe Expansion Header.....	10
3	Functional Description and Operation	12
3.1	Basic Setup for the Debug Connection	12
3.2	Auxiliary Target Control	13
3.3	SWO Trace Capture.....	13
3.4	ETB Trace Support	13
3.5	LED Operation	13
3.6	Energy Trace	13
3.7	Host SW Interfaces.....	17
4	XDS110 Adaptors	25
4.1	Debug Connection Adaptors	25
4.2	Auxiliary Connection Breakout Board.....	25

List of Figures

1	XDS110 Probe High-Level Block Diagram	6
2	Debug Connection (CTI-20) Pin Mapping	9
3	AUX Connection Signal Mapping	10
4	Setup Power Source Control	12
5	Probe-Supplied Power and Voltage Level	13
6	Enabling ET on Connect	14
7	Selecting XDS110.....	15
8	EnergyTrace Overview	15
9	EnergyTrace and the Toolbar	16
10	EnergyTrace Dynamic View	16
11	xdsdfu Screenshot	19
12	dbgjtag Screenshot JTAG Integrity Test	22
13	dbgjtag SWD Integrity Test	23
14	dbgjtag Scan Path Test.....	23

List of Tables

1	Acronyms and Definitions.....	6
2	Expansion Header Signal Mapping	11
3	LEDs and Probe Operational States.....	13
4	CTI to Other Adaptor Pin Mapping.....	25
5	Auxiliary Breakout Board Signal Mapping	26

XDS110 Debug Probe

1 Overview

The XDS110 debug probe is a low-cost system for debugging and tracing embedded systems centered on Texas Instruments (TI) microcontroller, microprocessor, and DSP-based systems. The XDS110 has improved performance relative to the XDS100 probe family, and added several useful capabilities such as probe-supplied target power and enhanced I/O. The XDS110 also includes support for power and energy profiling through TI's EnergyTrace™ (ET) technology.

1.1 XDS110 Probe Feature Summary

- Basic debug communications to the target system
 - IEEE 1149.1 (JTAG)
 - IEEE 1149.7 (cJTAG)
 - ARM serial wire debug (SWD)
- Enhanced and auxiliary debug communications
 - Support for trace capture through ARM serial wire output (SWO) – UART mode only
 - Support for UART communications to and from the target system
 - Support for GPIO channels
- Target I/O voltage support from 1.8 V to 3.6 V
- Power profiling features
 - Support for TI EnergyTrace
- Host communications
 - USB 2.0 high-speed (HS) communication link to the debug host system
 - Probe power through USB 5-V supply
- Expansion
 - A 30-pin expansion interface that can support a wide array of auxiliary functions
- Target power can be supplied from the probe

1.2 XDS110 System Summary

- Host platforms supported
 - The system supports various versions of Windows®, Mac OS™, and Linux® operating systems. Consult the documentation for CCS and other development environments for more details.
- IDE versions supported
 - TI CCS v7.0 and later
 - IAR (see IAR documentation)
 - Keil (see Keil documentation)

EnergyTrace is a trademark of Texas Instruments.
ARM is a registered trademark of ARM, Limited.
Mac OS is a trademark of Apple, Inc..
Linux is a registered trademark of Linus Torvalds.
Windows is a registered trademark of Microsoft Corporation.

- TI platforms, devices, and ISAs supported
 - MSP432 MCUs
 - CC26xx/13xx wireless MCUs
 - CC32xx/31xx Wi-Fi MCUs
 - Hercules and Conqueror safety MCUs
 - Sitara
 - Stellaris MCUs
 - C2000 MCUs
 - C66xx
 - C64x+
 - C674x
 - C55xx
 - C54xx
 - DaVinci
 - OMAP

1.3 XDS110 Performance

The XDS110 debug probe has much higher debug performance than XDS100v2. Depending on the target device, host environment, and configuration, the XDS110 performs 3x to 5x better than the existing XDS100v2 debug probe.

Example: BeagleBone Cortex A8 (Code Composer 6.2 setup on Windows 7 PC):

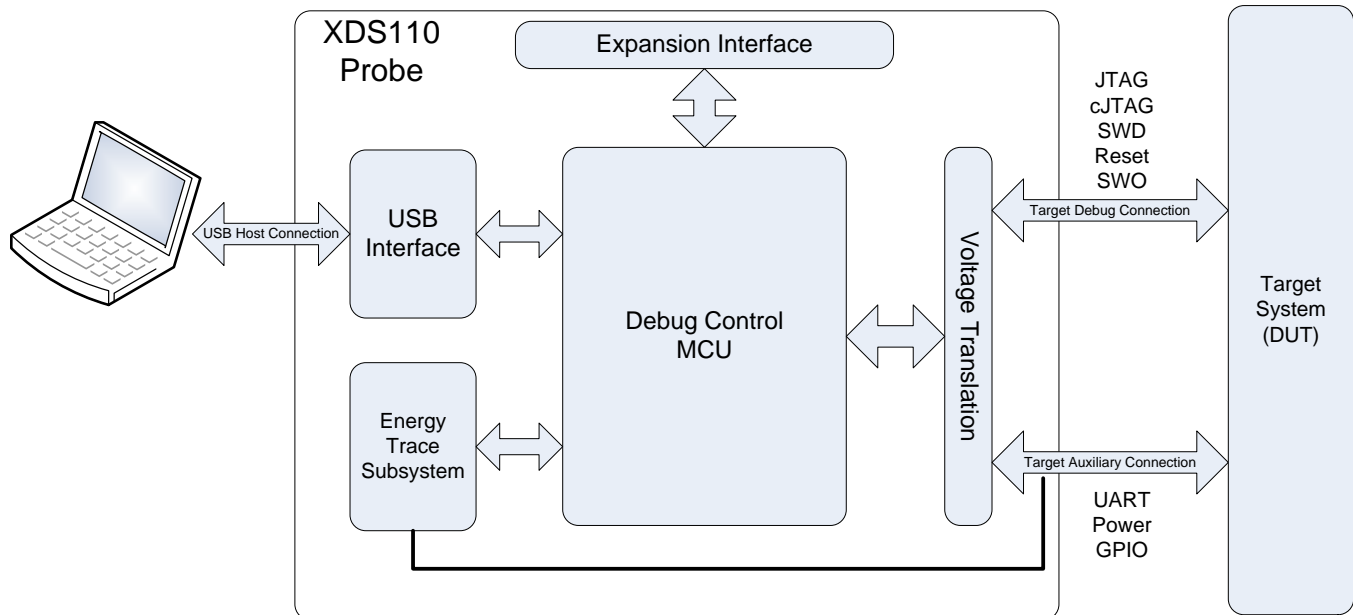
- XDS100v2 RAM download = 21 kbps
- XDS110 RAM download = 110 kbps

NOTE: Some TI scan-based debug platforms, such as the C6000, C5500, and C2000 DSP families, will not see the same magnitude of performance improvements.

1.4 XDS110 Probe Overview

[Figure 1](#) shows a high-level diagram of the major functional areas and interfaces of the XDS110 probe. Details of these are contained in [Section 3](#).

Figure 1. XDS110 Probe High-Level Block Diagram



1.5 XDS110 Parts List

The XDS110 debug probe system consists of the following hardware:

- The XDS110 debug probe
- One USB cable with Type-A female host connector and micro-B female connector for the probe
- One debug connection cable with CompactTI 20-pin connector (CTI-20)
- One auxiliary connection cable with 14-pin header
- One CTI-20 to Cortex-M 20-pin adaptor
- One CTI-20 to Cortex-M 10-pin adaptor
- One CTI-20 to TI 14-pin adaptor
- One 14-signal AUX breakout adaptor

1.6 Acronyms, Abbreviations, and Definitions

Table 1 shows the common acronyms and abbreviations used in this document.

Table 1. Acronyms and Definitions

Acronym	Definition
cJTAG	Compact JTAG
CAN	Controller area network
CMSIS-DAP	Cortex microcontroller software interface standard – debug access port
ET	Energy trace
GPIO	General purpose input output
HS	High speed
IDC	Insulation-displacement connector
JTAG	Joint test action group
OTG	On the go
SSI	Synchronous serial interface
SWCLK	Serial wire clock
SWD	Serial wire debug

Table 1. Acronyms and Definitions (continued)

Acronym	Definition
SWDAT	Serial wire data
SWO	Serial wire output
TCK	Test clock
TDI	Test data input
TDO	Test data out
TMS	Test mode select
TRSTn	Test reset (not)
UART	Universal asynchronous receiver transmitter
USB	Universal serial bus



CAUTION

This debug probe contains components that can potentially be damaged by electrostatic discharge. Always transport and store the debug probe in the supplied ESD bag when not in use. Handle using an antistatic wristband. Operate on an antistatic work surface. For more information on proper handling, refer to [Electrostatic Discharge \(ESD\)](#) (SSYA010).

2 Probe Interfaces

The XDS110 probe supports a number of interfaces for host and target communication.

2.1 Supported Communication Protocols

The XDS110 probe supports the following industry standard interfaces for host to probe and probe to target communications:

- Host to probe communication
 - USB 2.0 device with HS USB PHY
 - USB Communication Device Class protocol used for UART support
 - Standard USB Bulk IN and OUT endpoints support TI custom protocols
- Probe to target communications
 - IEEE 1149.1 JTAG
 - IEEE 1149.7 cJTAG
 - ARM serial wire debug (SWD)
 - ARM serial wire output (SWO) – UART mode only
 - Transmit and receive UARTs with RS-232C signaling – no hardware handshakes

2.2 USB

Host to probe communication is accomplished through a USB link. The probe has a female micro-USB B type connector. The probe functions as a USB device only (no host mode or OTG). Power for the XDS110 probe is sourced from the USB V_{BUS} (+5 V).

2.3 Debug Interface

The XDS110 probe supports a debug connection interface through the standard CTI-20 connector (see [Section 2.3.1](#)). The supported debug features include:

- 5-pin 1149.1 JTAG connection (including TRSTn)
- 2-pin 1149.1 cJTAG connection
- 2-pin ARM SWD connection
- 1-pin SWO overlaid on JTAG TDO
- Target system reset
- Target voltage detect
- Target disconnect detect
- Four EMU signals for GPIO
 - 2 × Probe to target
 - 2 × Target to probe
 - These signals are replicated on the AUX connector because many of the debug adaptors for CTI-20 do not support connections with GPIO.

2.3.1 Physical Connection for Debug

The CTI-20 connection is a 20-pin IDC connection using .050 by .100 inch pitch. The pin mapping is shown in [Figure 2](#).

Figure 2. Debug Connection (CTI-20) Pin Mapping

SWDIO/TMS	1	2	NTRST
TDI	3	4	TDIS
VTRef	5	6	Key
SWO/TDO	7	8	GND
NC	9	10	GND
TCK	11	12	GND
GPIOOUT0	13	14	GPIOOUT1
nRESET	15	16	GND
GPIOIN0	17	18	GPIOIN1
NC	19	20	GND

2.4 Auxiliary Debug Interface

The XDS110 probe supports an auxiliary interface (AUX) for additional debug features through a second 14-signal cable and connector (see [Section 2.4.1](#)). Many of the AUX features and functions are not available on the standard CTI-20 connector. These additional features include:

- A probe/target UART
- Probe-supplied target power
 - May be monitored for ET
- Target power supply input
 - May be looped back to the Target after monitoring for ET
- Four GPIO signals
 - 2 × Probe to target
 - 2 × Target to probe
 - These signals are replicated the CTI-20 connector, but are also present for scenarios where adaptors on the CTI-20 are supporting debug connections without GPIO capability.

2.4.1 Physical Connection for AUX

The AUX connection is a 14-pin IDC connection using .05 inch pitch. It is Samtec FFSD-compatible, with the female connector on both ends. The pin mapping is shown in [Figure 3](#).

Table 2. Expansion Header Signal Mapping

Alternate Functions	Tiva Pin	XDS110 Signal Name	Pin	Pin	XDS110 Signal Name	Tiva Pin	Alternate Functions
GPIO (PB5), ADC (AIN11), I2C5 Data	120	ET_SSICLK	1	2	ET_PN0	107	GPIO (PN0)
GPIO (PB4), ADC (AIN10), I2C5 Clock	121	ET_SSIFSS	3	4	ET_PN1	108	GPIO (PN1)
GPIO (PE4), ADC (AIN9)	123	ET_SSIDAT0	5	6	ET_PN2	109	GPIO (PN2)
GPIO (PE5), ADC (AIN8)	124	ET_SSIDAT1	7	8	ET_PN3	110	GPIO (PN3)
Ground		GND	9	10	GND		Ground
ADC (AIN1), GPIO (PE2)	13	ET_AIN1	11	12	ET_SCL	112	GPIO (PN5)
ADC (AIN2), GPIO (PE1)	14	ET_AIN2	13	14	ET_SDA	111	GPIO (PN4)
GPIO (PB0), CAN1 RX, UART1 RX, I2C5 Clock	95	ET_PB0	15	16	ET_PM2	76	GPIO (PM3), Timer3 CCP0
GPIO (PC4), UART7 RX	25	ET_PC4	17	18	ET_PH3	32	GPIO (PH3)
GPIO (PB1), CAN1 TX, UART1 TX, I2C5 Data	96	ET_PB1	19	20	ET_PC5	27	GPIO (PC5), UART7 TX
		POD_NON_ET_VCC_SUPPLY	21	22	POD_NON_ET_VCC_SUPPLY		
		DEBUG_TARGET_VDD_IN	23	24	DEBUG_TARGET_VDD_IN		
Ground		GND	25	26	GND		Ground
Digital 3.3 V		E3V3	27	28	E5V0		Digital 5 V
Digital 3.3 V		E3V3	29	30	E5V0		Digital 5 V

3 Functional Description and Operation

3.1 Basic Setup for the Debug Connection

Setting up the XDS110 debug probe is similar to most of the other debug probes in the TI portfolio, and the general debug tool documentation can guide users on the basic setup.

Refer to the [CCS getting started guide](#) for details on setup and configuration steps. Similar documentation exists for other IDE vendors. Extra setup steps are required for some features, and these are outlined below.

3.1.1 Target-Supplied Power

The XDS110 debug probe can be used for debugging targets across 1.8-V to 3.6-V IO levels. The XDS110 probe can also be used to supply power to targets with 1.8-V to 3.6-V IO and with current draw limited to ~400 mA. Configuring the power supply capability requires some additional setup steps.

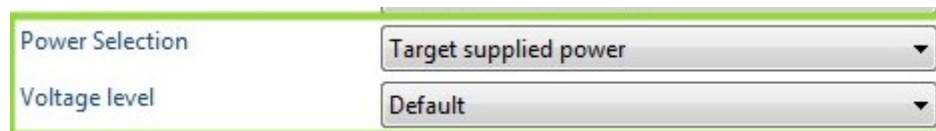
3.1.1.1 Hardware Setup

When the target is being powered externally, the debug probe does not supply the power. The only connection between the XDS110 and the target is the JTAG header. No other HW setup is required.

3.1.1.2 CCS Setup

To use the XDS110 probe in Code Composer Studio, CCSv7.0 or later must be installed. To set up the connection properties, open the CCXML target configuration for the target, click on the Advanced tab, and select the XDS110 in the hierarchy. The panel on the right shows all the XDS110 connection properties. Set the Power Selection field to Target supplied power, and the Voltage Level to Default, as shown in [Figure 4](#).

Figure 4. Setup Power Source Control



3.1.2 Probe-Supplied Power

The ability to supply target power from the probe is a new feature of the XDS110, and requires additional HW and SW setup.

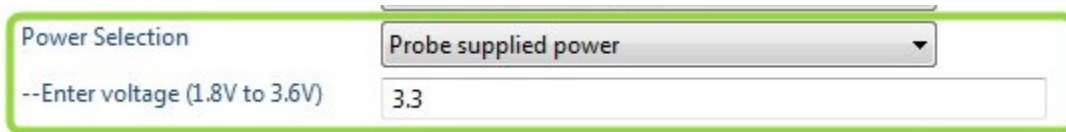
3.1.2.1 Hardware Setup

In this mode, the XDS110 JTAG header is connected to the target for debug, and the XDS110 AUX cable is used to supply the power. The TGTSUPPLYOUT and GND pins on the AUX connector (refer to the [Figure 3](#)) are connected to the supply pins of the target.

3.1.2.2 CCS Setup

To use the XDS110 probe in Code Composer Studio, CCSv7.0 or later must be installed. To set up the connection properties, open the CCXML target configuration for the target, click on the Advanced tab, and select the XDS110 in the hierarchy. The panel on the right shows all the XDS110 connection properties. Set the Power Selection field to Probe supplied power. An edit box appears for voltage level. Specify the target IO voltage level in the edit box. [Figure 5](#) shows an example configuration with supply set to 3.3 V.

Figure 5. Probe-Supplied Power and Voltage Level



The power supply to the target is turned on when a debug or EnergyTrace session is started, and turned off when the session terminates.

3.2 Auxiliary Target Control

There are additional interfaces between the probe and target that are mapped to GPIO signals on both the CTI-20 and AUX cable. A typical use case for these signals is to drive GPIO inputs to the target to control boot modes. Currently, the state of these signals can be set through the dbgjtag utility (see [Section 3.7.3.3](#)).

3.3 SWO Trace Capture

The XDS110 probe supports ARM® SWO (serial wire output) trace for TI's single-core MCU and WCS devices. SWO trace is a single-pin trace interface that can be used for profiling hardware events such as program counter, data reads/writes, and interrupt entry and exit, as well as application-initiated software messages. When the XDS110 probe is supporting debug communication through 2-pin protocols such as ARM SWD (2-pins) or 2-pin cJTAG, the target may reuse the TDO pin for SWO trace output. Currently, only UART format is supported for transport of SWO data from target to host.

For more details on how to use SWO trace in Code Composer Studio and the devices supported, refer the documentation at the following link: http://processors.wiki.ti.com/index.php/SWO_Trace.

3.4 ETB Trace Support

The XDS110 probe supports exporting trace data stored in on-chip buffers called ETB (embedded trace buffer). The trace data captured in the ETB is device or trace component-specific. For more details on using the ETB, refer the documentation at the following link: <http://www.ti.com/lit/ug/spruhm7b/spruhm7b.pdf>

3.5 LED Operation

The XDS110 probe supports two LEDs to provide feedback on the operating state to the user. [Table 3](#) maps LED functionality to probe operational states.

Table 3. LEDs and Probe Operational States

Green LED	Red LED	Probe Status
Off	Off	Probe is not powered, booting, or in Flashing mode
On	Off	Probe is operating normally but no active debug connection
On	On	Probe is operating normally and there is an active debug connection
On	Rapid Flash	Debug transactions are being processed

3.6 Energy Trace

3.6.1 Introduction

The XDS110 debug probe has on-board circuitry that can be used for measuring the target's energy consumption. The hardware circuitry provides high-accuracy energy consumption with low bandwidth current and power profile. The energy profiling range covers 1- μ A to 100-mA current draw, above which the tool will display an overcurrent message and shutdown. This tool is ideal for characterizing energy consumption, but not for capturing short current spikes, because sampling occurs over large time windows (~500 μ sec).

3.6.2 Specifications

3.6.2.1 Accuracy

- $\pm 2\%$ OR ± 500 nA, Condition: $I < 25$ mA, VBUS = 5-V constant
- $\pm 5\%$ OR ± 500 nA, Condition: $I > 25$ mA & $I < 100$ mA, VBUS = 5 V
- Overcurrent condition > 100 mA

3.6.2.2 Device Support

EnergyTrace is only available on single-core Cortex M devices at this time. This includes the MSP432, CC13/26xx, CC31/32xx, and the TM4C family of devices.

3.6.2.3 Modes

Depending on the target capability, there are three modes of energy profiling:

1. EnergyTrace: Energy profiling only. This mode is supported for all the single-core Cortex M devices mentioned above.
2. EnergyTrace+: Energy profiling with program counter correlation. This mode is supported for the MSP432 device family only.
3. EnergyTrace++: Energy profiling with program counter and peripheral state correlation. This mode will be available in upcoming devices.

The XDS110 can support all of the above modes if the target device supports it.

3.6.3 Hardware Setup

The XDS110 debug probe must supply power to the target for measuring energy. Refer to [Section 2](#) for hardware setup required when the probe is supplying power.

3.6.4 Usage in Code Composer Studio

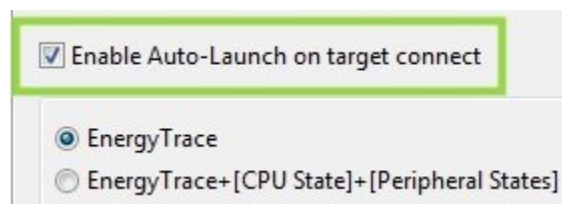
The EnergyTrace tool in Code Composer Studio can be used for profiling energy, power, and current consumption while a debug session is active, as well as outside of a debug session.

3.6.4.1 CCS Setup

Refer to [Section 2](#) for details on the target connection (CCXML) setup.

To start the EnergyTrace tool automatically on launching debug session, open up the Preferences section by going to the Window menu and selecting the Preference menu item. Enable the checkbox for “Enable Auto-launch on target connect”.

Figure 6. Enabling ET on Connect



Alternatively, if the EnergyTrace tool must be started after establishing a debug connection, then when the core is connected, go to the Tools menu and select the EnergyTrace menu item.

Set the target connection as XDS110. [Figure 7](#) indicates the specific fields.

Figure 7. Selecting XDS110

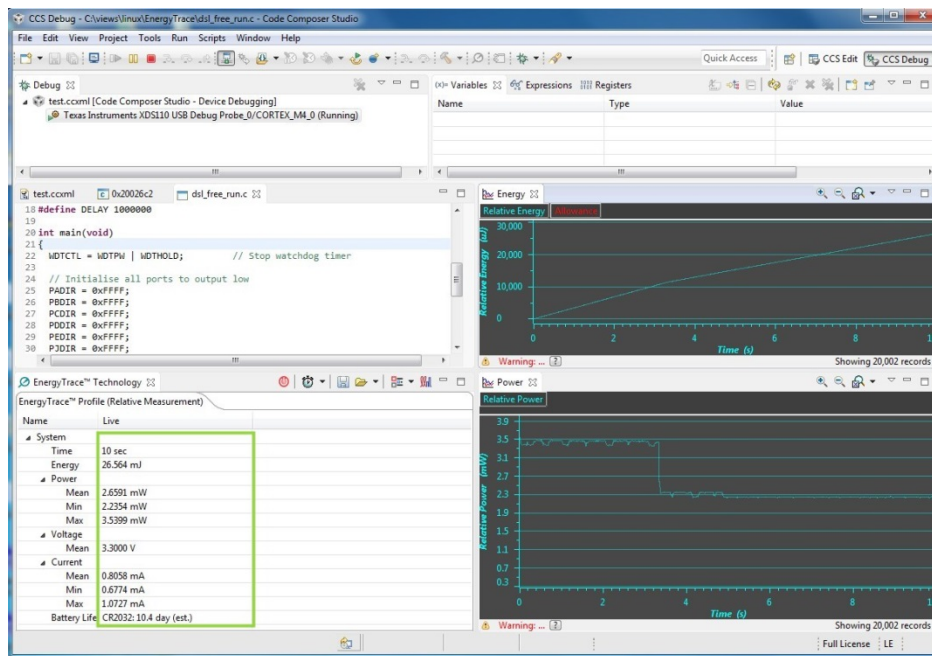


3.6.4.2 EnergyTrace Within a Debug Session

When a debug session is launched, the core is connected, and the EnergyTrace tool is started, three EnergyTrace windows appear. The windows include the EnergyTrace Technology (main view), which shows the statistical data, the Power graph, and the Energy graph. Trace collection starts when the core runs and stops, either when the core halts or when the stopwatch timer in the main view expires, depending on whichever occurs first. The trace collection time can be changed by clicking on the stopwatch icon in the EnergyTrace Technology window.

Figure 8 shows a sample collection of 10 seconds. The Main UI shows the average, minimum, and maximum current and power consumed, as well as the total energy consumption. It also indicates the estimated battery life. The graphs show the consumption over time.

Figure 8. EnergyTrace Overview

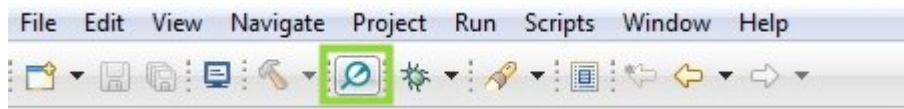


The EnergyTrace session is automatically terminated when the debug session is terminated. To terminate EnergyTrace collection while debugging, click on the red power icon in the EnergyTrace Technology window.

3.6.4.3 EnergyTrace Without Debug Intervention

In this capture mode, true energy measurements can be performed without debug overhead. To do so, first disconnect the JTAG header from the target. To start the tool, click on the blue icon with the EnergyTrace Logo from the CCS Toolbar, as shown in Figure 9.

Figure 9. EnergyTrace and the Toolbar

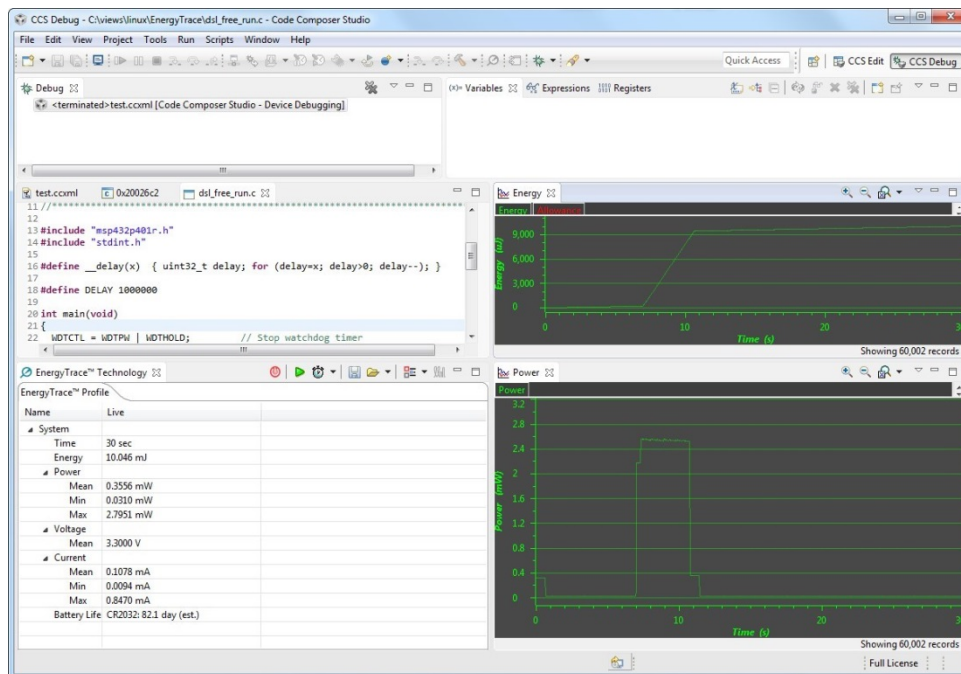


A dialog box opens up describing EnergyTrace measurement terminology. Click on the Proceed button. Three EnergyTrace windows open up. The windows include the EnergyTrace Technology (main view) display, which shows the statistical data, the Power graph, and the Energy graph.

To start EnergyTrace capture, click on the green Start icon in the EnergyTrace Technology window. The duration of the capture can be changed by using the stopwatch icon in the main view. To stop trace collection while trace capture is in progress, click on the Pause icon that replaces the Start icon. Trace capture can be restarted by clicking on the green Start icon again.

Figure 10 was taken with an MSP432 LaunchPad running a low-power application. The target was reset in the middle of the capture, causing a shoot up of the power and current consumed, indicating wakeup followed by a dip, which indicates entry into a low-power state. Note that the device is completely disconnected.

Figure 10. EnergyTrace Dynamic View



The EnergyTrace Technology window shows the true values for average, minimum, and maximum current and power consumed, as well as the total energy consumption. It also indicates the estimated battery life. The graphs show the true consumption numbers over time.

To terminate EnergyTrace collection, click on the red power icon in the EnergyTrace Technology window.

3.6.4.4 MSP432 EnergyTrace+ (Energy Consumption With Program Correlation)

For details on how to invoke ET+ mode, available only with the MSP432 device family, refer to the documentation at the following link: http://processors.wiki.ti.com/index.php/Energy_Trace_for_MSP432.

3.6.5 Usage With Command Line Utility – stune

The Windows-based CCS installation has an interactive command line utility called stune which can be used for capturing EnergyTrace data to a CSV file. This does not require CCS to be running. However, the setup does require the target’s connection file (.ccxml), which is created in CCS.

To use stune, open a command window and change directories to the stune directory in the CCS installation at: <Install base>\ccs_base\emulation\analysis\bin\stune. Type “stune” and press enter to drop into the stune command shell.

The first step is to establish connection. Use the connect command with the target connection file in the following format within stune:

```
connect -c targetConfig.ccxml xds
```

EnergyTrace data can be collected either for a fixed duration, or for an indefinite duration with user intervention to stop trace collection. In fixed duration mode, when trace collection stops, the average, maximum, and minimum current and total energy consumed is reported. In the indefinite mode, when trace collection starts, the window is updated with live updates for the current and energy values, until the user terminates the collection by pressing Ctrl+C.

For example:

To capture 5 seconds of EnergyTrace, enter the following command:

```
energytrace -D 5000 -o output.csv et
```

The `-D` option specifies the time in milliseconds.

The `-o` option specifies the output filename. The file format is fixed as CSV.

To capture EnergyTrace with user intervention to stop collection, omit the `-D` option.

```
energytrace -o output.csv et
```

The `-o` option specifies the output filename. The file format is fixed as CSV.

To stop collection, press the Ctrl+C key combination.

3.7 Host SW Interfaces

3.7.1 Serial Communications

A bidirectional UART channel is provided for additional host to target communications (with the probe as a UART-to-UART bridge and the UARTs mapped on the AUX header). The UART channel is realized on the host through a USB CDC driver, and enumerated as Virtual Comm Port.

3.7.2 CMSIS-DAP

CMSIS-DAP is a standard interface for creating debug probes capable of debugging ARM Cortex microcontrollers through the CoreSight debug access port (DAP). CMSIS-DAP support consists of software that is ported into the firmware of an interface chip, such as the TM4C129 CPU of the XDS110. This firmware code provides a standardized USB interface that allows the host to make DAP access requests, and handles converting those requests into the necessary JTAG or SWD protocols.

CMSIS-DAP has been included as part of the XDS110 debug probe. Any debugger able to communicate with a CMSIS-DAP-enabled debug probe can use the XDS110 directly with no other software required.

Debuggers that include support for CMSIS-DAP include the following: Keil uVision, IAR Workshop, and OpenOCD.

The USB VID/PID of the XDS110 are 0x0451/0xbef3. Some CMSIS-DAP debuggers may require the user to provide these IDs to find the XDS110.

3.7.3 TI XDS Utilities

The XDS110 probe supports three utilities that can be useful for managing debug functionality external to a debug IDE.

3.7.3.1 Rest Control Utility – `xds110reset`

`xds110reset` is a command line utility to control the board reset feature of the XDS110. `xds110reset` was created to provide reset control, without the need to install the entire XDS software stack.

The reset line controller is the board or system reset pin on the debug header. Asserting this pin should cause a hard reset on the target device, similar to pressing a manual reset button. On the TI 20-pin header, this is pin 15 (nSRST). On the Cortex-M 10-pin header, this is pin 10 (nRESET). And on the ARM 20-pin header, this is pin 15 (nSRST). This reset is not available on the TI 14-pin header.

xds110reset provides the following features:

- Toggle the board reset with configurable delay.
- Assert or deassert the board reset line.
- Choose which XDS110 to use by serial number

Installation Path:

```
.../<CCS Install>/ccs_base/common/uscif/xds110
```

Usage:

```
xds110reset<command> <...>
```

Supported commands:

- -a <NAME>, -action <NAME>
Choose an action to perform. NAME may be assert, deassert, or toggle. If not specified, toggle is executed by default.
- -d <VALUE>, -delay <VALUE>
Set the asserted time for the reset toggle in milliseconds. Has no effect if action is assert or deassert. If not specified, delay is set to 50 ms by default.
- -s <TEXT>, -serial <TEXT>
Select the XDS110 probe by serial number. TEXT is the serial number to use, up to eight characters. If not specified, the first XDS110 found is used.
- -h, -help
Show help for these commands, and exit.

Examples:

How to reset the target using an XDS110:

```
xds110reset
```

xds110reset connects to the first XDS110 it finds. It then toggles the board reset line, holding it asserted for 50 ms.

How to toggle the board reset, holding asserted for 3 seconds:

```
xds110reset --action toggle --delay 3000
```

The given delay is the time between asserting the reset and releasing it. The --delay option is only used when the --action option is toggle.

3.7.3.2 Firmware Maintenance Utility - xdsdfu

xdsdfu is a command line utility that provides several features for examining and maintaining the firmware of the XDS110 debug probe. While the XDS software stack includes an auto-update feature for ensuring the latest firmware is always flashed, the user may need to manually examine or update the firmware. xdsdfu also allows the user to view and set the XDS110 probe serial number.

xdsdfu provides the following features:

- Report the XDS110 firmware version and serial number
- Place the XDS110 into flash programming mode (DFU mode)
- Download the bootloader into the XDS110
- Download the firmware into the XDS110
- Set a new serial number into the XDS110
- Reset the XDS110 to restart the firmware

Installation path:

```
.../<CCS Install>/ccs_base/common/uscif/xds110
```

Usage:

```
xdsdfu <command> <...>
```

Supported commands:

- -e
Enumerate connected devices, show info, then exit.
- -m
Switch XDS110 into programming mode (DFU mode).
- -b <FILE>
Download the given bootloader file into the device.
- -f <FILE>
Download the given firmware file into the device.
- -s <TEXT>
Set the XDS110 serial number to given text, any eight character string (no spaces). This option replaces the entire serial number.
- -r
Reset the XDS110 on completion of another command.
- -? or -h
Show help for these and additional commands.

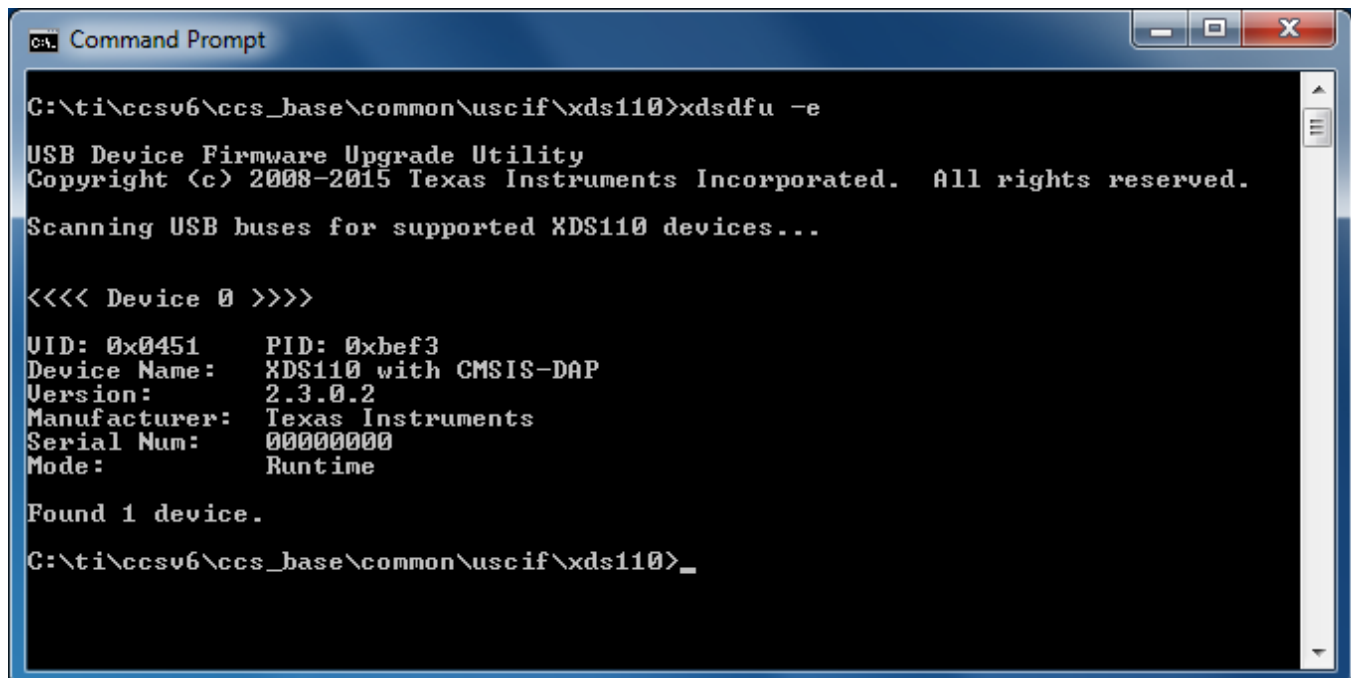
Examples:

How to examine the firmware in all connected XDS110 probes:

```
xdsdfu -e
```

xdsdfu examines all connected XDS110 probes and devices in DFU mode, and reports the details of each device. Sample output is shown in [Figure 11](#).

Figure 11. xdsdfu Screenshot



```

C:\>cd C:\ti\ccsv6\ccs_base\common\uscif\xds110
C:\ti\ccsv6\ccs_base\common\uscif\xds110>xdsdfu -e
USB Device Firmware Upgrade Utility
Copyright (c) 2008-2015 Texas Instruments Incorporated. All rights reserved.
Scanning USB buses for supported XDS110 devices...

<<<< Device 0 >>>>
UID: 0x0451      PID: 0xbef3
Device Name:    XDS110 with CMSIS-DAP
Version:        2.3.0.2
Manufacturer:   Texas Instruments
Serial Num:     00000000
Mode:           Runtime

Found 1 device.
C:\ti\ccsv6\ccs_base\common\uscif\xds110>_

```

How to program new firmware into the XDS110 probe:

```
xdsdfu -m
xdsdfu -f firmware.bin -r
```

The -m command must be executed separately. When -m is executed, the XDS110 reconfigures its USB interface to enable the DFU mode. It then reconnects as a different USB device, and the OS needs a moment to recognize it. The -r command tells the XDS110 to reboot after programming the firmware.

How to program a new bootloader and firmware into the XDS110 probe:

```
xdsdfu -m
xdsdfu -b bootloader.bin -r
xdsdfu -m
xdsdfu -f firmware.bin -r
```

The -m commands must be executed separately, but the second -m may not be necessary if the XDS110 probe flash was blank.

How to change the serial number of the XDS110 probe:

```
xdsdfu -m
xdsdfu -s 00000000 -r
```

3.7.3.3 Connection Diagnostic Utility – dbgjitag

dbgjitag is a command line utility that provides multiple commands for testing and operating the features of Texas Instruments debug probes. dbgjitag is a tool for diagnosing problems with the debug connection, and dbgjitag allows the user to exercise some control over features of the debug probe.

dbgjitag provides the following features for XDS110 users:

- Report the installed XDS emupack software version
- Test the reliability of the debug connection (JTAG, cJTAG, and SWD)
- Measure the scan path (JTAG and cJTAG)
- Reset the probe and target into test-logic-reset state
- Reset the target board through the system reset pin (nSRST)
- Configure, read, and write the XDS110 GPIOs

Installation path:

```
...<CCS Install>/ccs_base/common/uscif
```

Usage:

```
dbgjitag <command>, <variable=value>
```

Supported commands for XDS110:

- -f @<debug probe>
Select which debug probe and scan mode to use. For XDS110 use: @xds110, @xds110cjtag, or @xds110swd for JTAG, cJTAG, or SWD modes.
- -f <board file>
Select which CCS board configuration file to use.
- -r
Reset the debug probe and its target through the nTRST pin. If used with the commands below, -r is executed first.
- -v
Enable verbose output.
- -S integrity
Test the DR/IR scan paths with fixed data (JTAG and cJTAG), or test the SWD connection by reading the target IDCODE (SWD).

- -S pathlength
Measure the length of the DR/IR scan paths (JTAG and cJTAG).
- -Y reset, system = boolean
Reset the debug probe and its target through the nSRST or nTRST pins.
- -Y gpiopins, config=number, write=number, read=boolean, mask=number
Configure, write, and read the user GPIO pins.
- -h
Show a list of major commands.
- -<command> help
Show help for a given major command. For example, -S help.

Examples:

How to get help:

```
dbgjtag -h
```

When executing dbgjtag, the -h command displays a list of the major commands and display information about the specific build of dbgjtag, including the emupack version. To find help on a specific major command, execute the command with “help” for the sub-command. For example, to get help for the scan command, execute the following:

```
dbgjtag -S help
```

This lists all of the sub-commands for -S with explanations of the parameters used for each.

How to test the scan connection for JTAG and cJTAG modes:

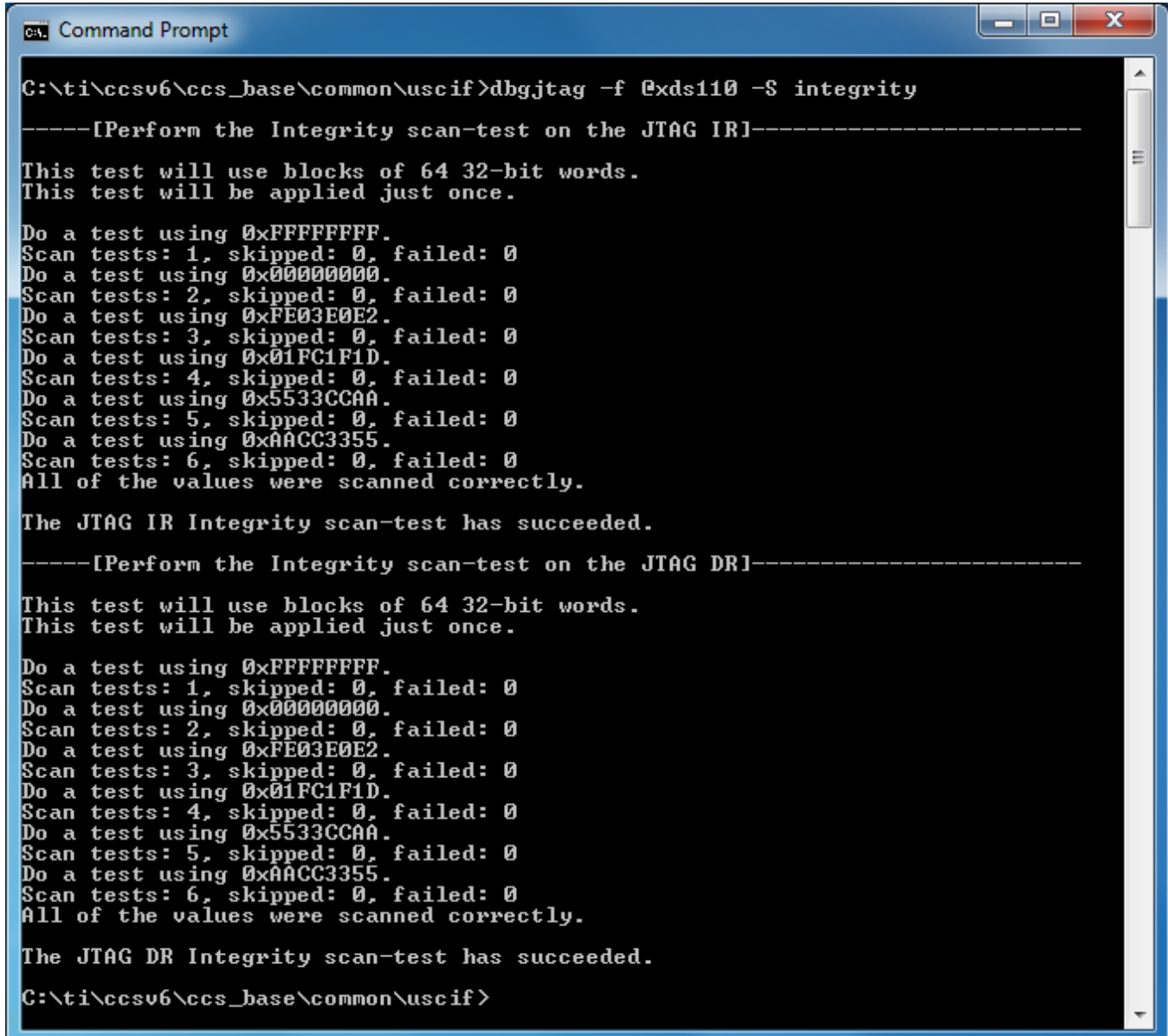
For JTAG:

```
dbgjtag -f @xds110 -S integrity
```

For cJTAG:

```
dbgjtag -f @xds110cjtag -S integrity
```

The output for a successful test appears as in [Figure 12](#).

Figure 12. dbgjtag Screenshot JTAG Integrity Test


```

C:\ti\ccsv6\ccs_base\common\uscif>dbgjtag -f @xds110 -S integrity

-----[Perform the Integrity scan-test on the JTAG IR]-----

This test will use blocks of 64 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 64 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.

C:\ti\ccsv6\ccs_base\common\uscif>
    
```

The IR and DR scan paths are tested separately. Look for both to have succeeded to know that the debug connection is working correctly. This is the test performed by the Test Configuration button in CCS.

How to test the debug connection for SWD mode:

```
dbgjtag -f @xds110swd -S integrity
```

The output for a successful test appears as in [Figure 13](#).

Figure 13. dbgjtag SWD Integrity Test

```

C:\ti\ccsv6\ccs_base\common\uscif>dbgjtag -f @xds110swd -S integrity
-----[Perform the SWD Mode Integrity test]-----
This test will read the IDCODE register 100 times.
The IDCODE register value is 0x2ba01477.
The SWD Mode Integrity test has succeeded.
C:\ti\ccsv6\ccs_base\common\uscif>_

```

The SWD connection is tested by reading the target IDCODE register multiple times. Look for the test to report that it has succeeded to know that the debug connection is working correctly. This is the test performed by the Test Configuration button in CCS.

How to measure the scan path length in JTAG and cJTAG modes:

For JTAG:

```
dbgjtag -f @xds110 -S pathlength
```

For cJTAG:

```
dbgjtag -f @xds110cjtag -S pathlength
```

The result of the test appears as in [Figure 14](#).

Figure 14. dbgjtag Scan Path Test

```

C:\ti\ccsv6\ccs_base\common\uscif>dbgjtag -f @xds110 -S pathlength
-----[Perform the standard path-length test on the JTAG IR and DR]-----
This path-length test uses blocks of 64 32-bit words.
The test for the JTAG IR instruction path-length succeeded.
The JTAG IR instruction path-length is 6 bits.
The test for the JTAG DR bypass path-length succeeded.
The JTAG DR bypass path-length is 1 bits.
C:\ti\ccsv6\ccs_base\common\uscif>

```

The IR and DR paths are measured separately. The IR measurement is the sum of all of the IR lengths in the scan path. The DR measurement is the sum of all of the DR lengths, with all of the devices in the scan path put into BYPASS. Because the DR length while in BYPASS is 1 bit, the DR measurement can be used as a count of the number of devices in the scan path. In the pictured example, the scan path consists of a single device with an IR length of 6 bits.

How to do a board reset with the XDS110:

```
dbgjtag -f @xds110 -Y reset, system=yes
```

dbgjtag toggles the board reset pin on the debug header (nSRST). This the same type of reset done by the xds110reset utility, but the xds110reset utility has additional control over the duration of the reset.

How to do a debug reset (Test-Logic-Reset) with the XDS110 (JTAG and cJTAG):

For JTAG:

```
dbgjtag -f @xds110 -r
```

For cJTAG:

```
dbgjtag -f @xds110cjtag -r
```

dbgjtag puts the XDS110 probe and connected targets into the JTAG test-logic-reset (TLR) state. To do this, it both toggles the nTRST pin on the debug header and executes a JTAG state transition to TLR through the TMS and TCK pins. Thus, if a target device does not have an nTRST pin, this command still puts it into TLR through the JTAG state transition.

The -r command can be used in conjunction with other commands. For example:

```
dbgjtag -f @xds110 -S integrity -r
```

The -r command is always done first. In this case, dbgjtag puts the probe and target into TLR before executing the scan integrity test.

How to use the XDS110 GPIO pins:

The XDS110 includes 4 GPIO pins on the AUX connector that can be controlled by the user. dbgjtag includes a command to configure, write, and read these GPIO pins:

```
dbgjtag -f @xds110 -Y gpiopins, config=number, write=number, read=boolean, mask=number
```

The GPIO pins are accessed by reading and writing the lower four bits of the number value. GPIO 3 and GPIO 2 are inputs into the XDS110. GPIO 1 and GPIO 0 are outputs from the XDS110. GPIO 1 and GPIO 0 are initially configured as inputs (hiZ). The user must configure these as outputs to enable these pins.

config=number sets the direction of each GPIO. Writing a b'1' for a GPIO configures it as an output. Writing a b'0' for a GPIO configures it as an input. Only GPIO 1 and GPIO 0 can be configured as outputs.

write=number sets the output level of the GPIO pins configured as outputs. Writing a b'0' to a GPIO sets the output level low. Writing a b'1' to a GPIO sets the output level high.

read=boolean selects if the command also reads the current value of the GPIO pins. "boolean" may be either "yes" or "no". If "yes," the value of the pins is read and displayed.

mask=number provides a bit mask to limit which GPIOs are to be affected. If a GPIO bit is set to b'0' in the mask, that GPIO is not affected by the config, write, or read commands. If not supplied, all pins are affected by the other commands.

Using the -v (verbose) command with the -Y gpiopins command modifies the output to display the current config value and always read the pins, and display the result.

Examples:

```
dbgjtag -f @xds110 -Y gpiopins, config=0x3, write=0x0
```

Configure GPIO 3 and GPIO 2 as inputs and GPIO 1 and GPIO 0 as outputs. Set the values of GPIO 1 and GPIO 0 both to b'0'.

```
dbgjtag -f @xds110 -Y gpiopins, config=0x3, write=0x3, mask=0x2
```

The mask limits the operations to GPIO 1. Configure GPIO 1 as an output and set its value to b'1'. The configuration and values for the other three GPIOs are not affected.

```
dbgjtag -f @xds110 -Y gpiopins, read=yes
```

Read the current value of the GPIO pins.

```
dbgjtag -f @xds110 -Y gpiopins -v
```

Display both the configuration and current value of the GPIO pins (following other output displayed by the verbose mode).

4 XDS110 Adaptors

4.1 Debug Connection Adaptors

4.1.1 Supported Debug Adaptors

The XDS110 supports a native CTI-20 debug connector. The product also supplies adaptor boards that support adapting to the following on-board debug connectors:

- ARM Cortex-M 20-pin (CM20)
- ARM Cortex-M 10-pin (CM10)
- TI Legacy 14-pin (TI14)

4.1.2 Debug Adaptor Pin Mapping

Table 4 defines the signal mapping used for each adaptor type.

Table 4. CTI to Other Adaptor Pin Mapping

Signal	XDS110 Pin	CTI20 Pin	CM20 Pin ⁽¹⁾	CM10 Pin ⁽¹⁾	TI14 Pin
SWDIO/TMS	1	1	2	2	1
TRSTn	2	2			2
TDI	3	3	8	8	3
TDIS	4	4			4
VTREF	5	5	1	1	5
KEY	6	6	7	7	6
TDO/SWO	7	7	6	6	7
GND	8	8	3,5,9,11,15,17,19	3,5,9	8
RTCK		9			9
GND	10	10			10
SWCLK/TCK	11	11	4	4	11
GND	12	12			12
EMU0/TRIGOUT0	13	13			13
EMU1/TRIGOUT1	14	14			14
nRESET	15	15	10	10	
GND	16	16			
EMU2/TRIGIN0	17	17			
EMU3/TRIGIN1	18	18			
EMU4		19			
NC	20	20			

⁽¹⁾ Any signal not listed in this table is not connected through the adaptor.

4.2 Auxiliary Connection Breakout Board

The XDS110 probe product also ships with a breakout board for the auxiliary (AUX) signals, so that boards without a native AUX connection can be wired into the functions supported through this interface.

4.2.1 Auxiliary Breakout Board Pin Mapping

Table 5 defines the signal mapping used for the 12-pin inline stake connector of the breakout board.

Table 5. Auxiliary Breakout Board Signal Mapping

XDS110 AUX Signal	Break Board Interface Pins
TGTSUPPLYIN	1
TGTSUPPLYOUT	2
TGTVDD(Sense)	3
GND	4,5,10
GPIOOUT0	6
GPIOOUT1	7
GPIOIN0	8
GPIOIN1	9
UARTTX	11
UARTRX	12

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ("TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products (<http://www.ti.com/sc/docs/stdterms.htm>), [evaluation modules](#), and [samples](http://www.ti.com/sc/docs/sampterm.htm) (<http://www.ti.com/sc/docs/sampterm.htm>).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2017, Texas Instruments Incorporated