User's Guide
SLAU575F–March 2015–Revised March 2017

Code Composer Studio™ IDE 7.1+ for SimpleLink™ MSP432™ Microcontrollers

This manual describes the use of the TI Code Composer Studio™ IDE version 7.1 and higher with the SimpleLink™ MSP432™ low-power microcontrollers. This manual describes only Code Composer Studio IDE for Windows® operating systems. The versions of Code Composer Studio IDE for Linux® and OS X® operating systems are similar and, therefore, are not described separately.

Most descriptions in this guide are valid for versions of Code Composer Studio IDE below 7.1, but the SimpleLink software development kit (SDK) requires Code Composer Studio IDE 7.1 or higher.

Contents

1 Installing Code Composer Studio IDE 4
2 Updating Code Composer Studio IDE 4
3 Creating a Basic SimpleLink MSP432 Project 5
4 Importing SimpleLink Examples From TI Resource Explorer 7
5 Debugging Your Application 8
6 Using Serial Wire Output (SWO) Hardware Trace Analyzer 15
7 EnergyTrace™ Technology 17
8 Device Security 29
9 Low-Power Debug 34
10 Frequently Asked Questions 37
11 Additional Code Composer Studio IDE Information 38
12 References 39

List of Figures

1 Check for Updates 4
2 Creating a New Code Composer Studio IDE Project 5
3 New Project Wizard 6
4 New Project Files 6
5 New CCS Project, Open Resource Explorer 7
6 Selecting an Example in TI Resource Explorer 8
7 Project Properties 9
8 Choose Debugger Connection 10
9 Debug Settings for GNU Compiler and MSP-FET 11
10 Predefined Symbol for ROM Debugging 12
11 Launch Debug Session 13
12 Debug Session 14
13 Target Configuration 15
14 Connection Properties 15
15 Pulse Density and Current Flow 17
16 EnergyTrace™ Technology Preferences 18
17 EnergyTrace™ Technology Control Bar 19
18 Debug Session With EnergyTrace+ Graphs 20
19 Profile Window 20
20 States Window 21

Copyright © 2015–2017, Texas Instruments Incorporated

21 Power Window 21
22 Energy Window 22
23 Debug Session With EnergyTrace Graphs 23
24 EnergyTrace Profile Window 24
25 Zoom Into Power Window 24
26 Current Profile (Blue) With Recorded Profile (Yellow) 24
27 Energy Profile of the Same Program in Resume (Yellow Line) and Free Run (Green Line) 25
28 Comparing Profiles in EnergyTrace+ Mode 26
29 Comparing Profiles in EnergyTrace Mode 26
30 Show Target Configuration View 29
31 List of Target Configurations 30
32 Launch Selected Target Configuration 30
33 Debug View After Launching Target Configuration 30
34 Show All Cores 31
35 List of All Cores in the MSP432 31
36 Manually Connecting to the DAP 32
37 DAP is Connected 32
38 Executing the Factory Reset Script 32
39 Mass Erase Script Console Output 33
40 Code Composer Studio IDE Tools – GEL File 33
41 Factory Reset With Password GEL File 34
42 Properties Menu 35
43 Enabling Low Power Run 36
44 CPU Core Status Display Indicating Deep Sleep Mode 36
45 Program Counter Located at WFI Instruction 37
46 Change Debugger Settings to SWD 37

Trademarks

Code Composer Studio, SimpleLink, MSP432, E2E, LaunchPad are trademarks of Texas Instruments.
CoreSight is a trademark of ARM Ltd.
ARM is a registered trademark of ARM Ltd.
OS X is a registered trademark of Apple Inc.
IAR Embedded Workbench is a registered trademark of IAR Systems.
Linux is a registered trademark of Linus Torvalds.
Windows is a registered trademark of Microsoft Corporation.
All other trademarks are the property of their respective owners.

Copyright © 2015–2017, Texas Instruments Incorporated

Preface: Read This First

How to Use This Manual

This manual describes only those Code Composer Studio IDE features that are specific to the SimpleLink MSP432 low-power microcontrollers. It does not fully describe the MSP432 microcontrollers or the complete development software and hardware systems. For details on these items, see the appropriate TI documents listed in Important MSP432 Documents on the Web.

Important Documents on the Web

The primary sources of information about MSP432 microcontrollers are the device-specific data sheets and the technical reference manual. The MSP432 website contains the most recent version of these documents.

Documents that describe the Code Composer Studio tools (Code Composer Studio IDE, assembler, C compiler, linker, and librarian) can be found at www.ti.com/tool/ccstudio. A Wiki page (FAQ) that is specfic to the Code Composer Studio IDE is available at processors.wiki.ti.com/index.php/Category:CCS, and the TI E2E™ Community support forums provide additional help.

Documentation for third party tools, such as IAR Embedded Workbench® for ARM® or the Segger J-Link debug probe, can be found on the respective company's website.

If You Need Assistance

Support for the MSP432 devices and the hardware development tools is provided by the Texas Instruments Product Information Center (PIC). Contact information for the PIC can be found on the TI website. The TI E2E Community support forums for the MSP432 provide open interaction with peer engineers, TI engineers, and other experts. Additional device-specific information can be found on the MSP432 website.

Copyright © 2015–2017, Texas Instruments Incorporated

1 Installing Code Composer Studio IDE

The Code Composer Studio IDE can be obtained from the TI website.

- MSP432 low-power microcontrollers are supported by Code Composer Studio IDE 6.1 and higher versions. Previous versions do not support MSP432 MCUs.
- To benefit from the SimpleLink ecosystem, Code Composer Studio IDE 7.1 or higher is required.

2 Updating Code Composer Studio IDE

Use the Code Composer Studio IDE update feature to update the installation before starting to work with MSP432 low-power microcontrollers. To check for updates, click **Help** → **Check for Updates**.

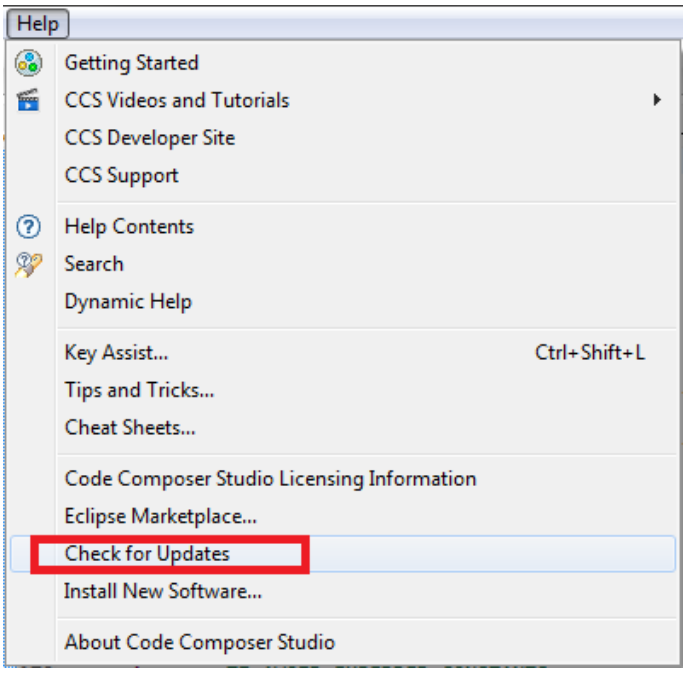

Figure 1. Check for Updates

Code Composer Studio IDE connects to the TI update server and retrieves information about relevant updates for the system. You can still select which update to install, but it is good practice to install all updates.

Copyright © 2015–2017, Texas Instruments Incorporated

3 Creating a Basic SimpleLink MSP432 Project

Code Composer Studio IDE organizes its projects in workspaces. A new workspace is generated automatically when you start Code Composer Studio IDE the first time. This workspace is blank.

You can either create a new project from scratch or start with one of the SimpleLink MSP432 SDK examples described in Section 4. TI recommends starting from one of the examples.

To create a new project in the current workspace:

1. Click **File** → **New** → **CCS Project**.

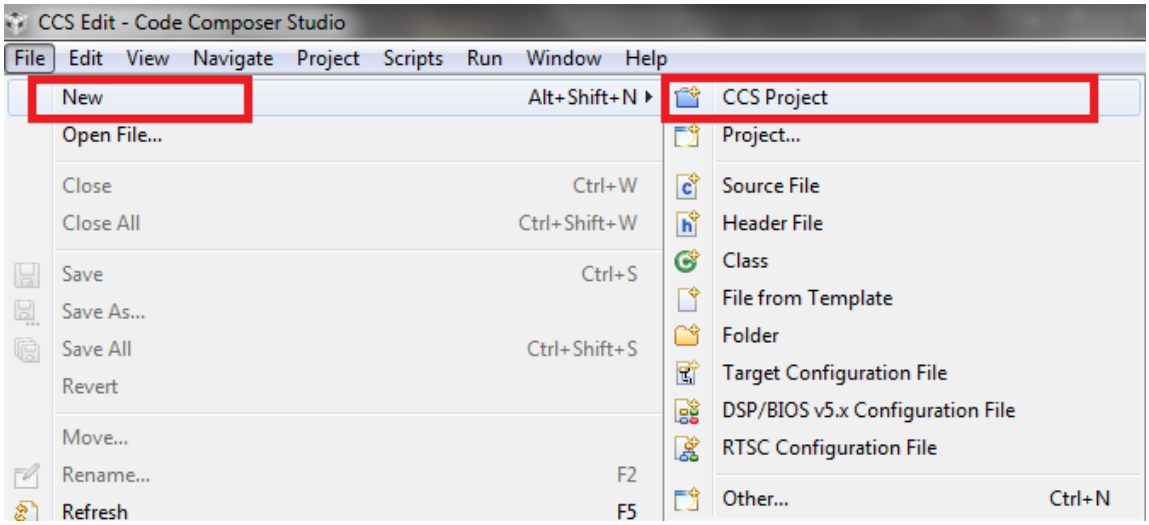

Figure 2. Creating a New Code Composer Studio IDE Project

2. Select **MSP432** as the target family (see Figure 3), which limits the list of devices in the device pulldown list.
3. Select the device being used; for example, MSP432P401R.
4. Select the debug connection to use. In the following example, this is an **XDS110** debug probe. These settings can be modified later in the Project Properties.
5. Type a unique project name.
6. Click **Finish.** A new project is created and, along with it, a number of files are copied into the workspace.

Copyright © 2015–2017, Texas Instruments Incorporated

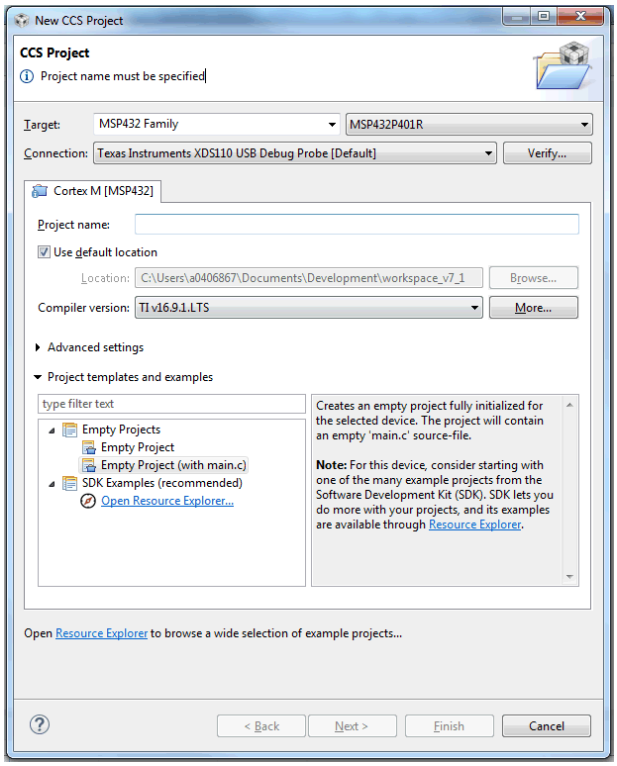

Figure 3. New Project Wizard

The workspace now contains a newly created project (), including:

- A basic main.c file (if you have chosen that in the New CCS Project dialog)
- The interrupt vector file **msp432_startup_ccs.c** where all interrupt handlers are predefined
- The linker command file **msp432p401r.cmd**
- The target configuration **MSP432P401R.ccxml** file including a link to the XDS-110 debug probe

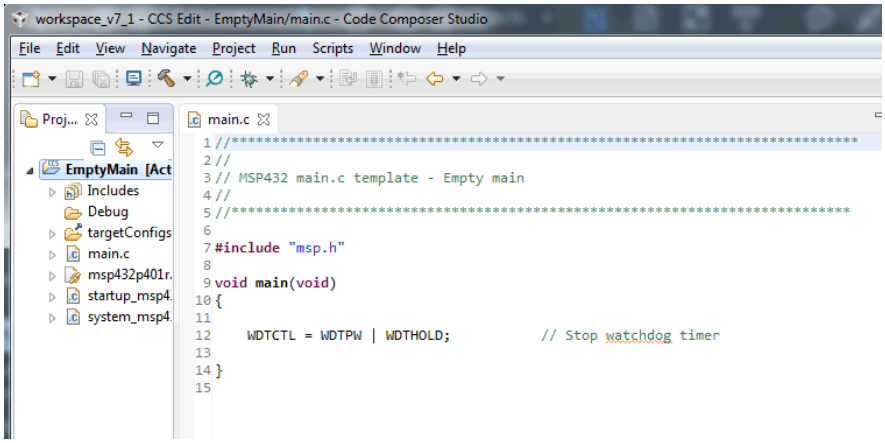

Figure 4. New Project Files

Now you can start developing code.

Copyright © 2015–2017, Texas Instruments Incorporated

4 Importing SimpleLink Examples From TI Resource Explorer

The recommended way to start a new project in Code Composer Studio IDE is to use an appropriate example from the TI Resource Explorer and adapt it. The TI Resource Explorer makes all examples and documentation from the SimpleLink MSP432 SDK available inside Code Composer Studio IDE. This includes drivers, RTOS, and examples from bare metal to high-level APIs.

The TI Resource Explorer is available online at http://dev.ti.com/tirex/ and is also included in Code Composer Studio IDE 7.0 and above. In Code Composer Studio IDE 7.1+, either select **View** → **Resource Explorer** or create a new project and select **SDK Examples** → **Open Resource Explorer** (see Figure 5).

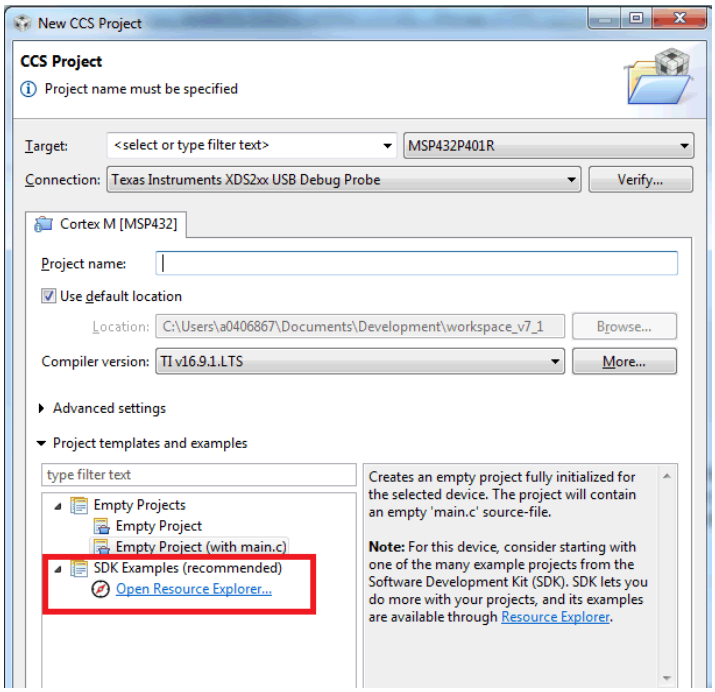

Figure 5. New CCS Project, Open Resource Explorer

In the TI Resource Explorer, select your MSP432 device in the search field to list only the examples that apply to that device, so that you can more easily find an example that suits you. Basic ADC examples, for example, are found in the hierarchy (see Figure 6).

Copyright © 2015–2017, Texas Instruments Incorporated

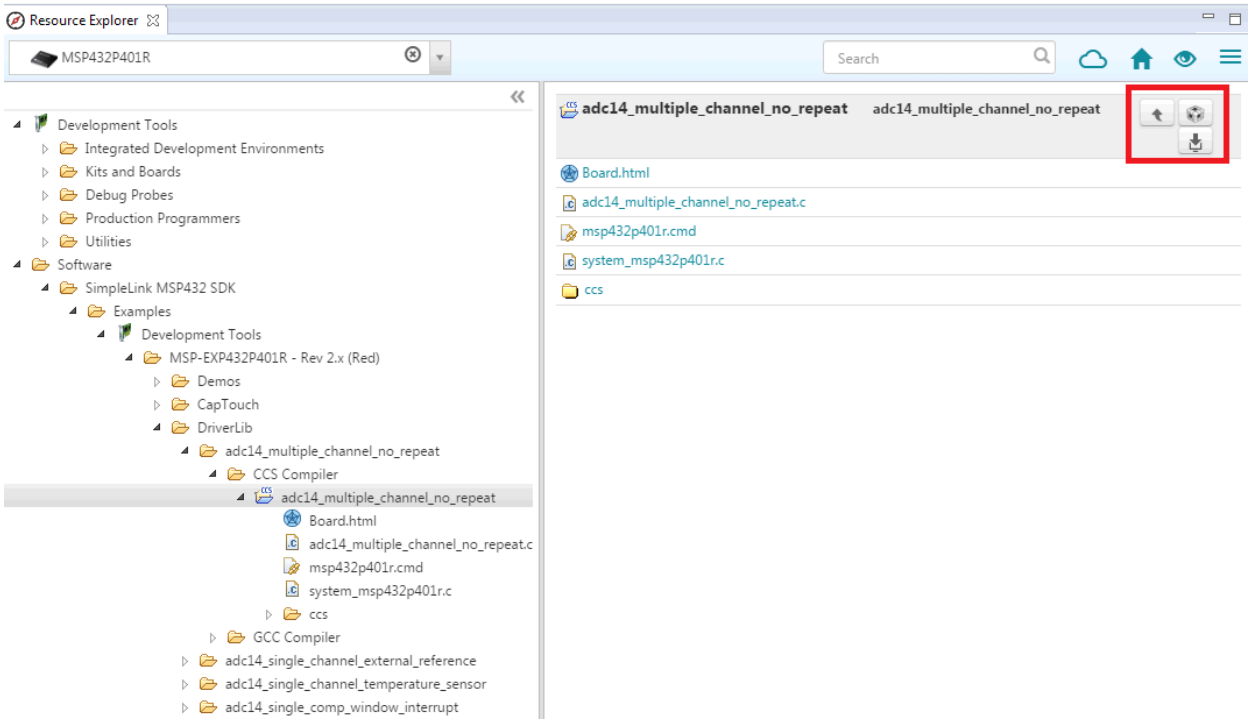

Figure 6. Selecting an Example in TI Resource Explorer

Click one of the buttons in the top right corner (see Figure 6) to download the SDK and import the example to Code Composer Studio IDE. When the SDK is already locally installed in the default location, all examples are taken directly from the installed SimpleLink MSP432 SDK.

Alternatively, you can import the example directly from the SDK installation path using the import dialog in your project explorer. After import, you can continue to compile or expand the example.

All documentation for the SDK can be found in the SDK installation path in the docs folder. Open Documention_Overview.html from that folder and then navigate to the Quick Start Guide for your IDE.

A one-time integration of the SimpleLink platform lets you add any combination of devices from the portfolio into your design. The ultimate goal of the SimpleLink platform is to achieve 100 percent code reuse when your design requirements change. For more information, visit www.ti.com/simplelink.

5 Debugging Your Application

The following debug probes have been tested successfully with Code Composer Studio IDE.

- Texas Instruments XDS100v2, XDS100v3, XDS200, XDS110 (including the XDS110 stand-alone probe)
- Texas Instruments MSP-FET (for more information visit the MSP-FET page)
- Segger J-Link (for more information, visit the TI J-Link Support Emulator wiki page)

  In Code Composer Studio IDE 6.1.3.x, the Segger installation instructions are directly linked in the Code Composer Studio IDE Apps Center.

5.1 Debugger Settings

After you create a project, you can change the debugger used to debug the application. To change the debugger, **right click** on the project in the workspace and choose **Properties** (see Figure 7)**.**

Copyright © 2015–2017, Texas Instruments Incorporated

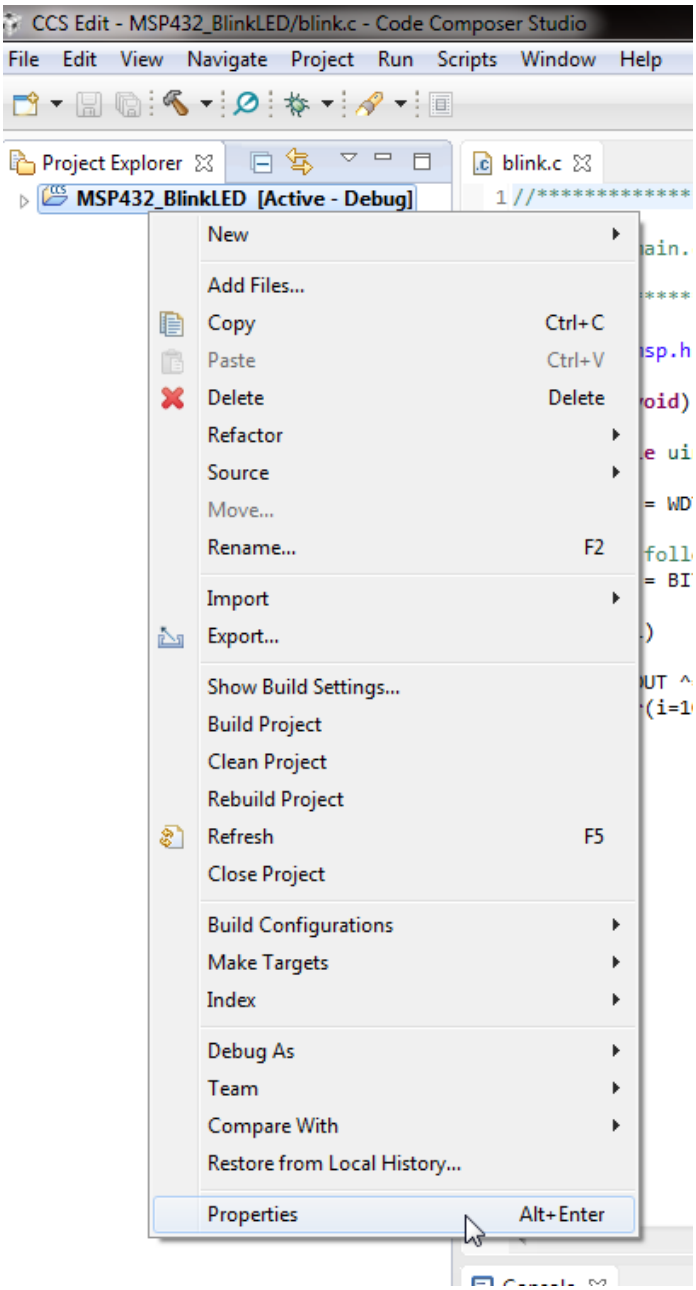

Figure 7. Project Properties

In the project properties window, make sure that you are in the **General** options pane. There you can see a drop-down list of target connections. Choose the debugger from this list and click **OK.**

Copyright © 2015–2017, Texas Instruments Incorporated

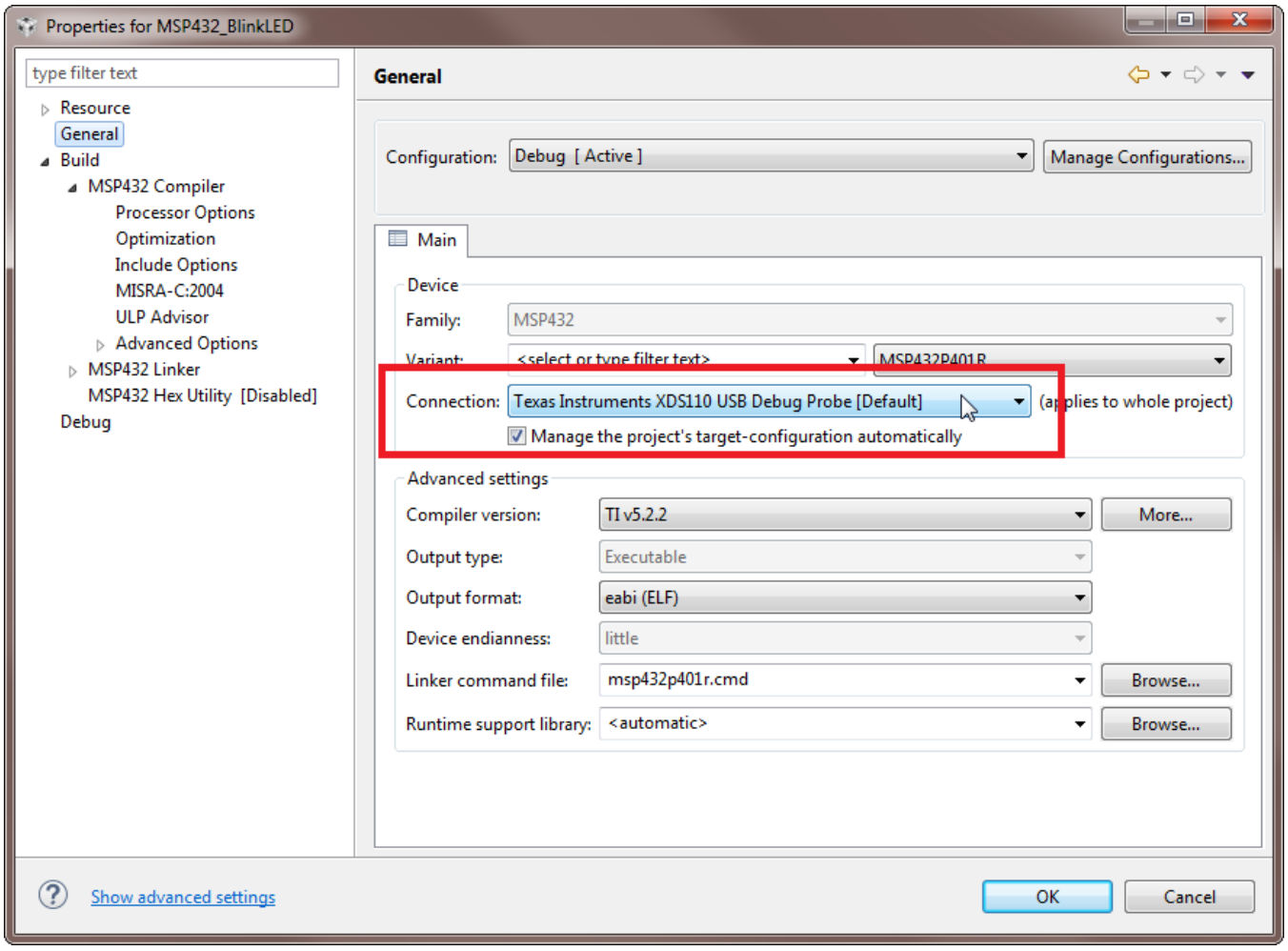

Figure 8. Choose Debugger Connection

More debugger settings can be made in the target configuration settings. For example, switching from JTAG to the less pin-consuming serial wire debug (SWD). See Section 6 for details on these settings.

NOTE: Figure 9 shows the additional settings that are needed for debugging when using MSP-FET and the GNU Linaro compiler (go to project properties → **Debug** → **Program/Memory Load Options**).

Copyright © 2015–2017, Texas Instruments Incorporated

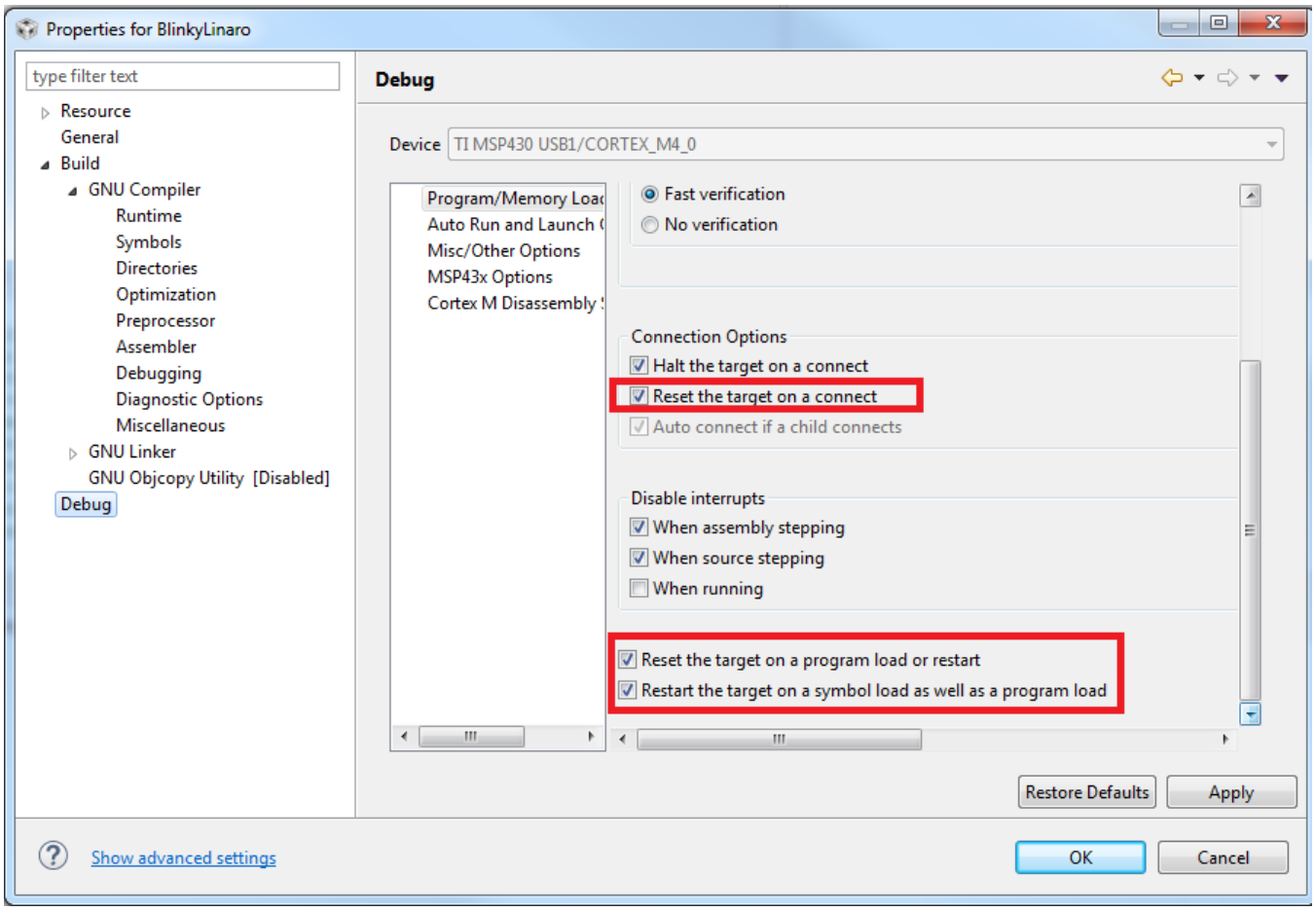

Figure 9. Debug Settings for GNU Compiler and MSP-FET

Copyright © 2015–2017, Texas Instruments Incorporated

5.2 Debugging ROM Driver Library

The MSP432P4xx family includes a complete peripheral driver library (DriverLib) fully integrated into the ROM memory. Developers can leverage the ROM DriverLib for multiple benefits including access to highly robust and tested APIs, single-cycle ROM execution speed at lower power consumption, and freeing up memory space for additional application code. Developers can gain access to ROM APIs by adding DriverLib header file to projects and linking to a prebuilt library.

The driver library source code is now part of the SimpleLink MSP432 SDK. The driver library is a low-level software layer below the TI drivers, which are also part of the SDK. In the SDK, the DriverLib source code can be found in <SDK_InstallationPath>\source\ti\devices\msp432p4xx\driverlib.

For more information on MSP432P4xx Driver Library and what is provided in ROM DriverLib, see the documentation in the SimpleLink MSP432 SDK.

To debug the ROM driver library, make sure that *TARGET_IS_MSP432P4XX* is listed in **Advanced Options → Predefined Symbols** (see Figure 10).

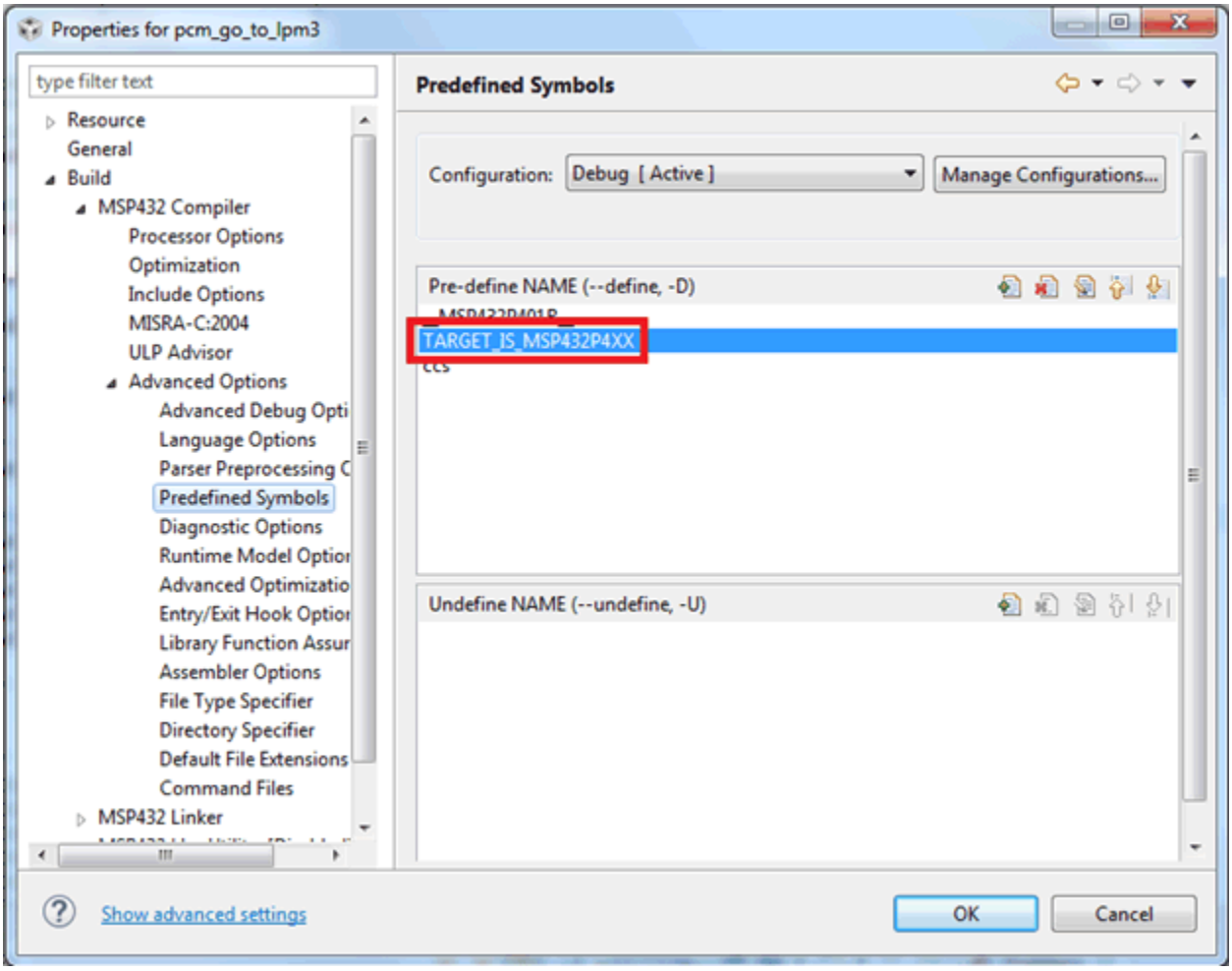

Figure 10. Predefined Symbol for ROM Debugging

Copyright © 2015–2017, Texas Instruments Incorporated

5.3 Start Debugging Session

After the project has been set to the correct debugger, you can begin to debug the application. To launch a debug session, click the icon in the top toolbar of the IDE (see Figure 11).

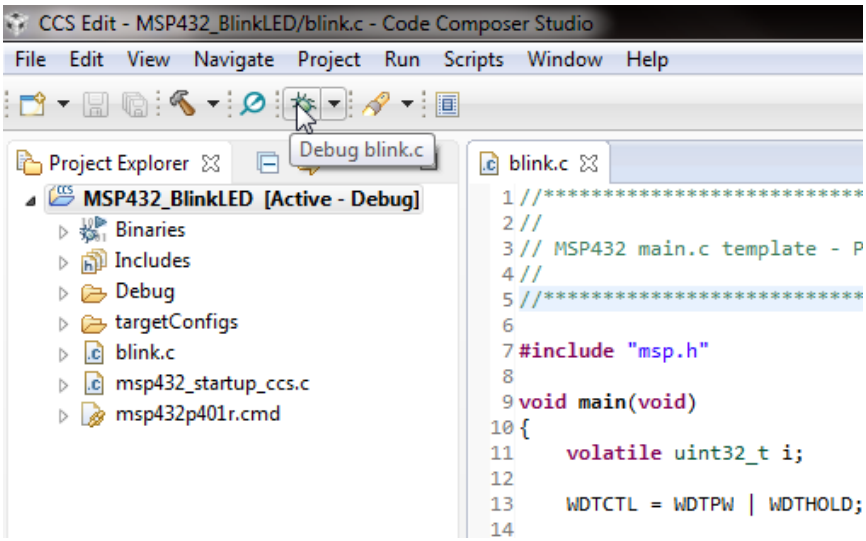

Figure 11. Launch Debug Session

After the IDE has finished compiling, linking, and downloading the code to the device, the debug session starts and you can begin to debug the application.

Copyright © 2015–2017, Texas Instruments Incorporated

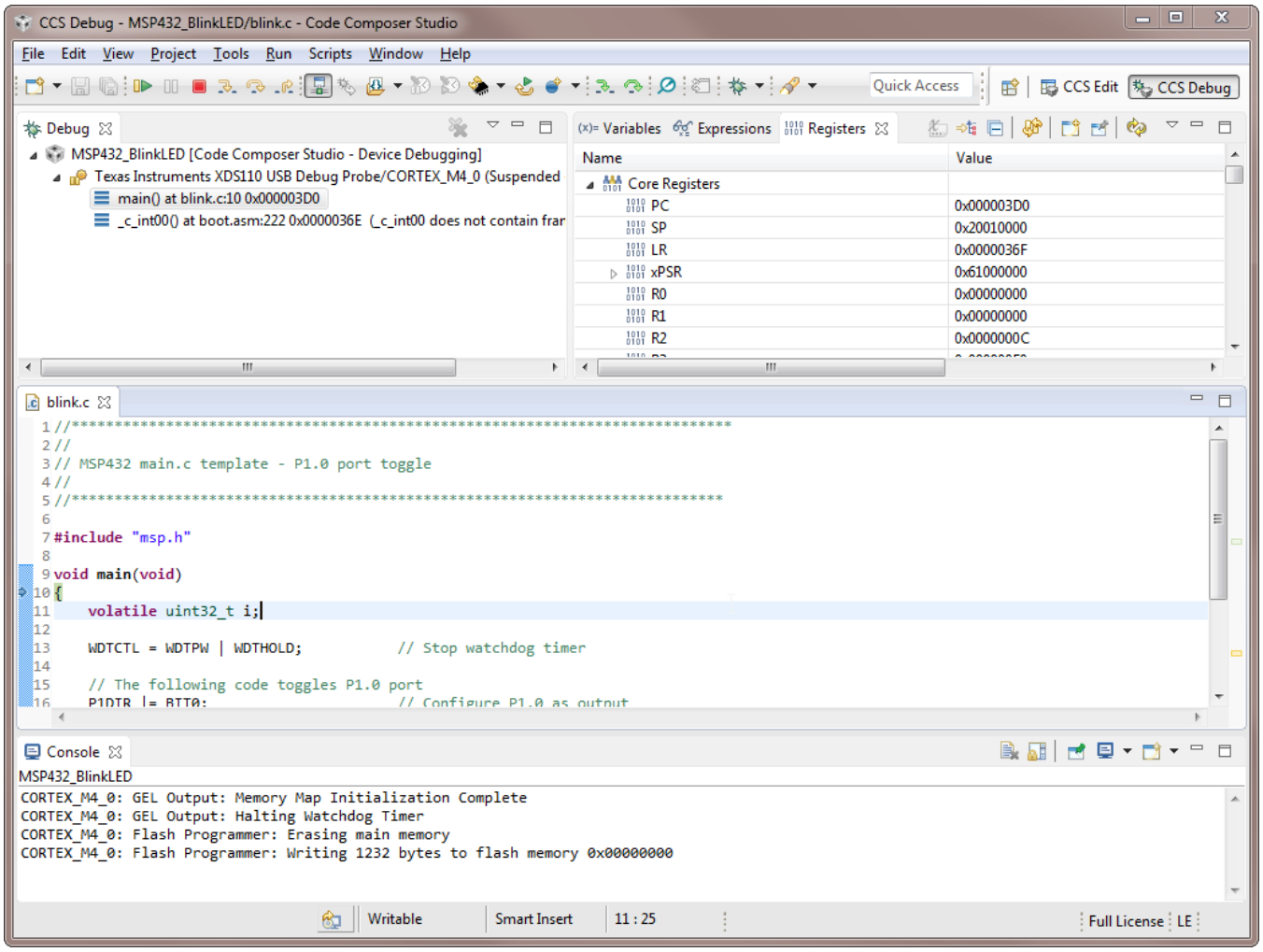

Figure 12. Debug Session

Copyright © 2015–2017, Texas Instruments Incorporated

6 Using Serial Wire Output (SWO) Hardware Trace Analyzer

In Code Composer Studio IDE, the SWO Trace tools for MSP432 MCUs are implemented using the features of the ARM CoreSight™ components, especially the Instrumentation Trace Macrocell (ITM) and Data Watchpoint and Trace Unit (DWT) (ETM is not present in MSP432 MCUs).

This user's guide describes only how to enable the SWO trace in Code Composer Studio IDE. For details of the trace hardware and example use cases, see Reference [3].

6.1 Configure the Project for SWO Trace

To enable SWO trace:

1. Expand the project in Project Explorer and open the projects *.ccxml file in the targetConfigs folder. Select the correct debug probe (see Figure 13).

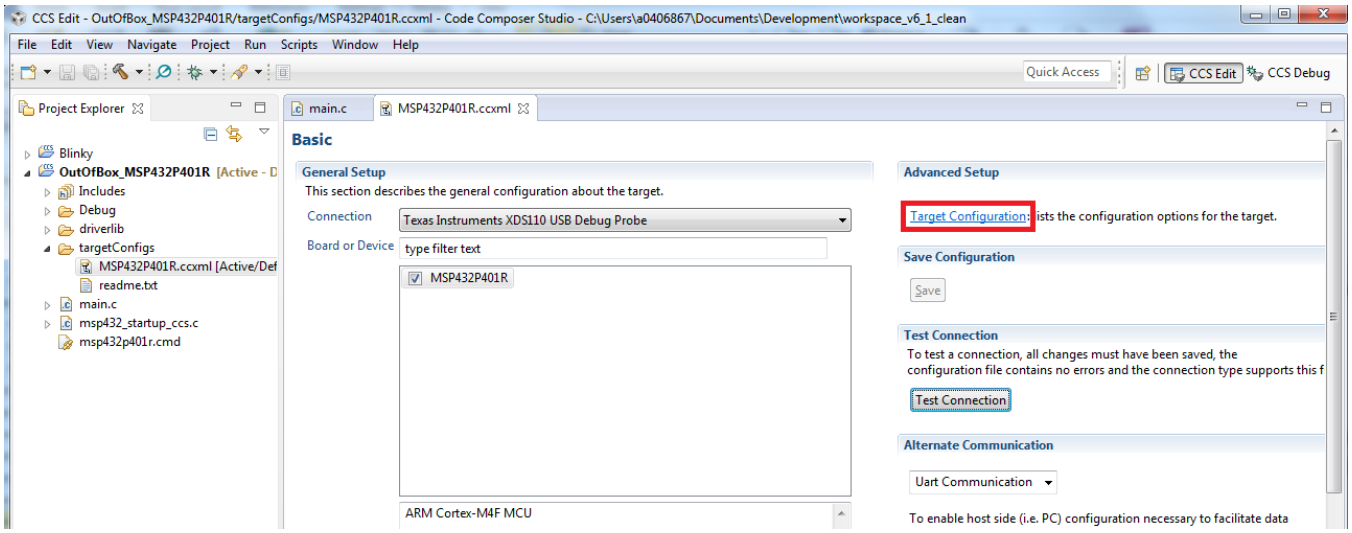

Figure 13. Target Configuration

2. Click Target Configuration in the Advanced Setup section then click on the selected debug probe.
3. Figure 14 shows the Connection Properties. In JTAG/SWD mode, select *SWD Mode – Aux COM port is target TDO pin*.

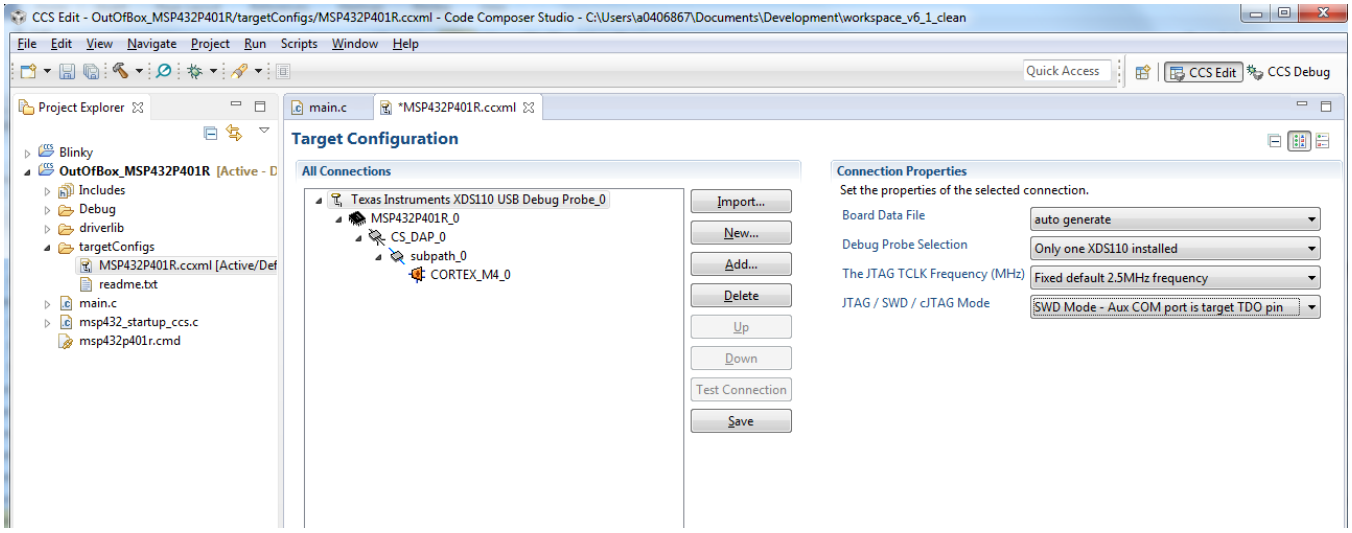

Figure 14. Connection Properties

Copyright © 2015–2017, Texas Instruments Incorporated

6.2 Run Serial Wire Trace

After finishing SWO configuration, start a debug session to start tracing:

1. Build the project either by selecting **Build Project** from the project's context menu or by selecting the hammer icon.
2. Enter a debug session: Right click the project name, select **Debug As → Code Composer Studio Debug Session**. Alternatively, click the bug icon.
3. In debug mode, go to **Tools → Hardware Trace Analyzer** and choose one of the following options:
   - Statistical Function Profiling for analyzing how often each function is called and what percentage of CPU cycles is consumed
   - Data Variable Tracing for obtaining a graph on the value of the variable using the ITM
   - Interrupt Profiling for getting data on when interrupts occurred and how these interrupt each other.
   - Custom Core Trace for tracing user defined strings at specified code locations

See *MSP432™ Debugging Tools: Using Serial Wire Output With CCS Trace Analyzer* for details.

Copyright © 2015–2017, Texas Instruments Incorporated

7 EnergyTrace™ Technology

EnergyTrace™ Technology is an energy-based code analysis tool that measures and displays the application's energy profile and helps to optimize it for ultra-low power consumption.

MSP432 devices with built-in **EnergyTrace+[CPU State]** (or in short **EnergyTrace+**) technology allow real-time monitoring of many internal device states while user program code executes. EnergyTrace+ technology is supported on selected MSP432 devices and debuggers.

EnergyTrace mode (without the "+") is the base of **EnergyTrace Technology** and enables analog energy measurement to determine the energy consumption of an application but does not correlate energy consumption to internal device information. The EnergyTrace mode is available for all MSP432 devices with selected debuggers, including Code Composer Studio IDE.

7.1 Energy Measurement

Debuggers with EnergyTrace technology support include a new and unique way of continuously measuring the energy supplied to a target microcontroller that differs considerably from the well-known method of amplifying and sampling the voltage drop over a shunt resistor at discrete times. A software-controlled dc-dc converter is used to generate the target power supply. The time density of the dc-dc converter charge pulses equals the energy consumption of the target microcontroller. A built-in on-the-fly calibration circuit defines the energy equivalent of a single dc-dc charge pulse.

Figure 15 shows the energy measurement principle. Periods with a small number of charge pulses per time unit indicate low energy consumption and thus low current flow. Periods with a high number of charge pulses per time unit indicate high energy consumption and also a high current consumption. Each charge pulse leads to a rise of the output voltage V_{OUT}, which results in an unavoidable voltage ripple common to all dc-dc converters.

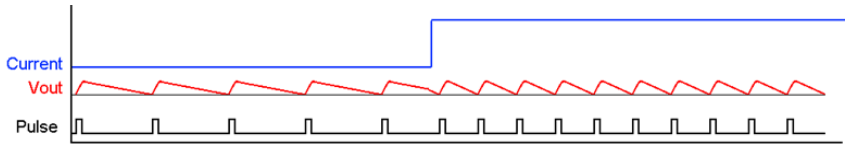

Figure 15. Pulse Density and Current Flow

The benefit of sampling continuously is evident: even the shortest device activity that consumes energy contributes to the overall recorded energy. No shunt-based measurement system can achieve this.

7.2 Integration With Code Composer Studio IDE

EnergyTrace technology is available as part of Texas Instrument's Code Composer Studio IDE for MSP432 microcontrollers. During debugging of an application, additional windows are available if the hardware supports EnergyTrace technology.

7.3 Enabling EnergyTrace Technology and Selecting the Default Mode

By default, the EnergyTrace technology feature is disabled in the Code Composer Studio Preferences. To enable it, go to **Window → Preferences → Code Composer Studio → Advanced Tools → EnergyTrace™ Technology** (see Figure 16).

Copyright © 2015–2017, Texas Instruments Incorporated

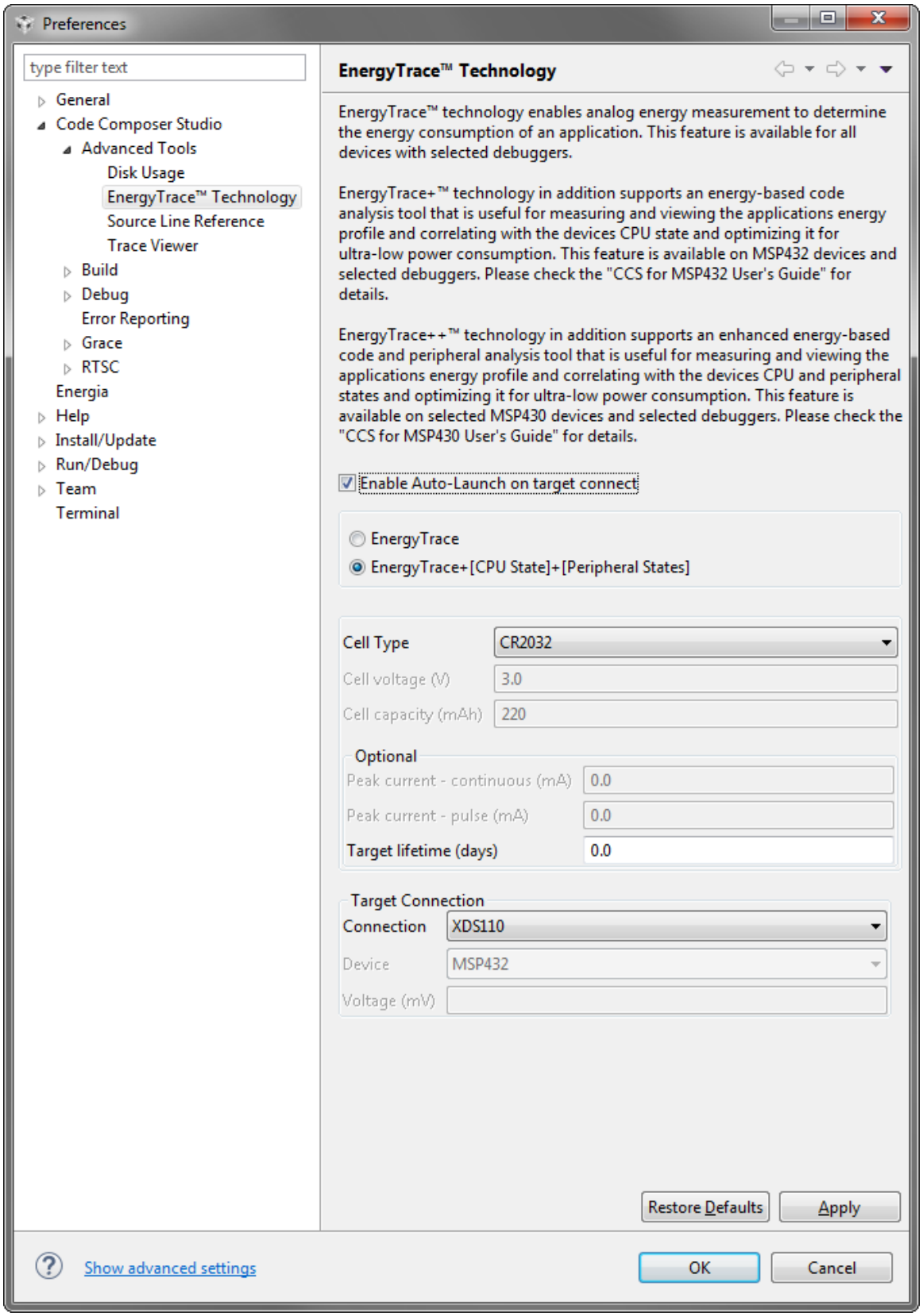

Figure 16. EnergyTrace™ Technology Preferences

Copyright © 2015–2017, Texas Instruments Incorporated

For the target connection, select XDS110 or, if using an MSP-FET that supports EnergyTrace technology, select one of the USB connections.

Two capture modes are supported.

- The full-featured **EnergyTrace+[CPU State]** mode that delivers real-time device state information together with energy measurement data
- The **EnergyTrace** mode that delivers only energy measurement data

Use the radio button to select the mode to enable when a debug session is launched. If an MSP432 device does not support device state capturing, the selection is ignored and Code Composer Studio starts in **EnergyTrace** mode.

While a debug session is active, click the icon in the **Profile** window to switch between the modes.

NOTE: If the EnergyTrace technology windows are not opened when a debug session starts, verify the following items:

- Does the hardware (debugger and device) support EnergyTrace technology? To determine if the selected device supports EnergyTrace technology, see the device-specific data sheet or the user guide that came with the evaluation board.
- Is EnergyTrace technology globally enabled in Window → Preferences → Code Composer Studio → Advanced Tools → EnergyTrace™ Technology?

7.4 Controlling EnergyTrace Technology

EnergyTrace technology can be controlled using the control bar icons in the **Profile** window (see Figure 17). Table 1 describes the function of each of these buttons.

Figure 17. EnergyTrace™ Technology Control Bar

Table 1. EnergyTrace™ Technology Control Bar Icons

	Enable or disable EnergyTrace technology. When disabled, icon turns gray.
	Set capture period: 5 sec, 10 sec, 30 sec, 1 min, or 5 min. Data collection stops after time has elapsed. However, the program continues to execute until the Pause button in the debug control window is clicked.
	Save profile to project directory. When saving an EnergyTrace+ profile, the default filename starts with "EnergyTrace_D" followed by a timestamp. When saving an EnergyTrace profile, the default filename starts with "EnergyTrace" followed by a timestamp.
	Load previously saved profile for comparison.
	Restore graphs or open Preferences window.
	Switch between **EnergyTrace+** mode and **EnergyTrace** mode

Copyright © 2015–2017, Texas Instruments Incorporated

7.5 EnergyTrace+ Mode

When debugging devices with built-in EnergyTrace+ support, the **EnergyTrace+ mode** gives information about both energy consumption and the internal state of the target microcontroller. The following windows are opened during the startup of a debug session:

- Profile
- States
- Power
- Energy

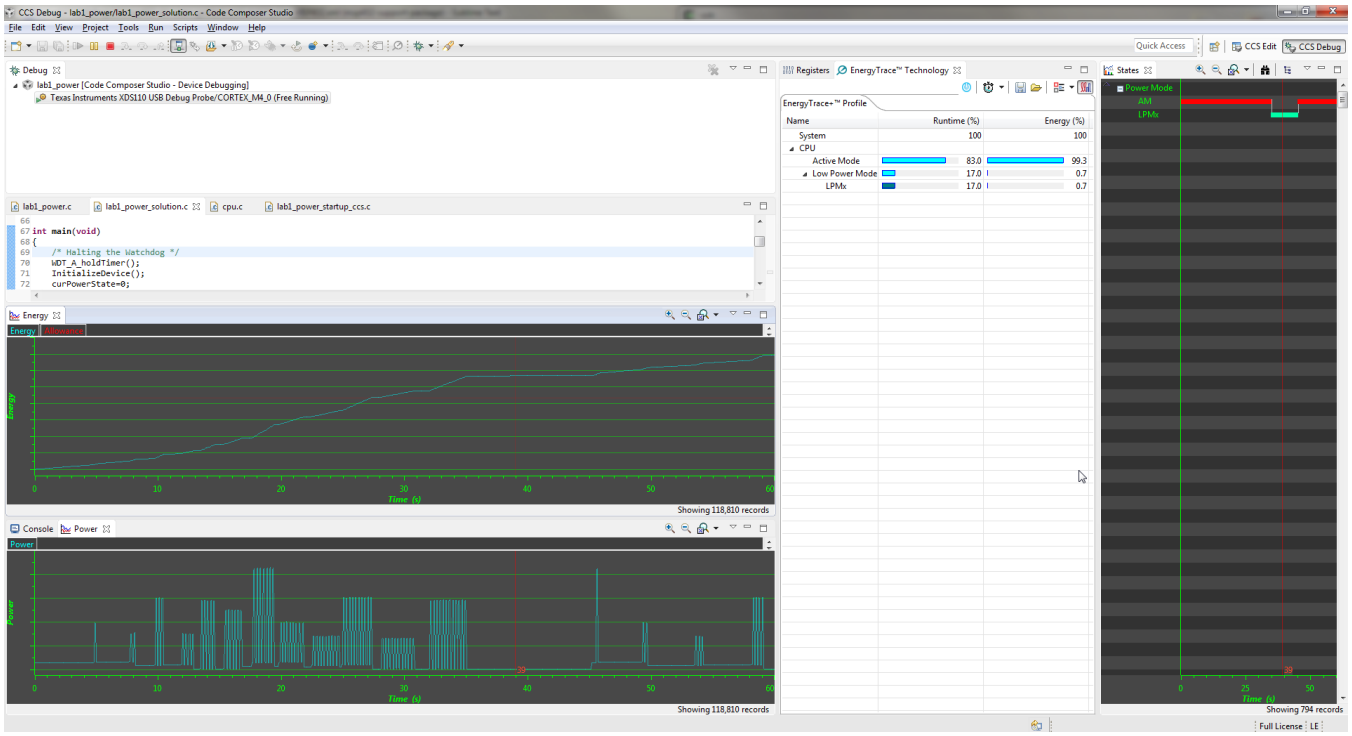

Figure 18. Debug Session With EnergyTrace+ Graphs

The **Profile** window (see Figure 19) is the control interface for EnergyTrace+. It can be used to set the capturing time or to save the captured data for later reference. The **Profile** window also displays a compressed view of the captured data and allows comparison with previous data.

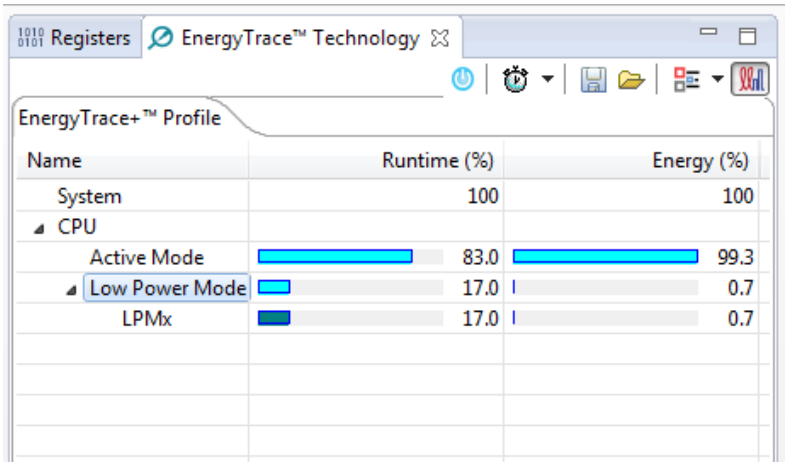

Figure 19. Profile Window

Copyright © 2015–2017, Texas Instruments Incorporated

The **Profile** window enables a quick overview of the resource use of the profiled application.

- CPU: Shows information about program execution
  - Low Power Mode: Shows a summary of low-power mode use.
  - Active Mode: Shows which functions have been executed during active mode. Functions in the run-time library are listed separately under the _RTS_ subcategory. If the device supports IP Encapsulation, a line labeled as <Protected> is displayed to indicate the time executing out of IP encapsulated memory.

The **States** window (see Figure 20) shows the real-time trace of the target microcontroller's internal states during the captured session. State information includes the power modes, on and off state of peripheral modules, and the state of the system clocks.

Figure 20 shows a device going into a low-power mode then back to active mode. The **States** window allows a direct verification of whether or not the application exhibits the expected behavior; for example, that a peripheral is disabled after a certain activity.

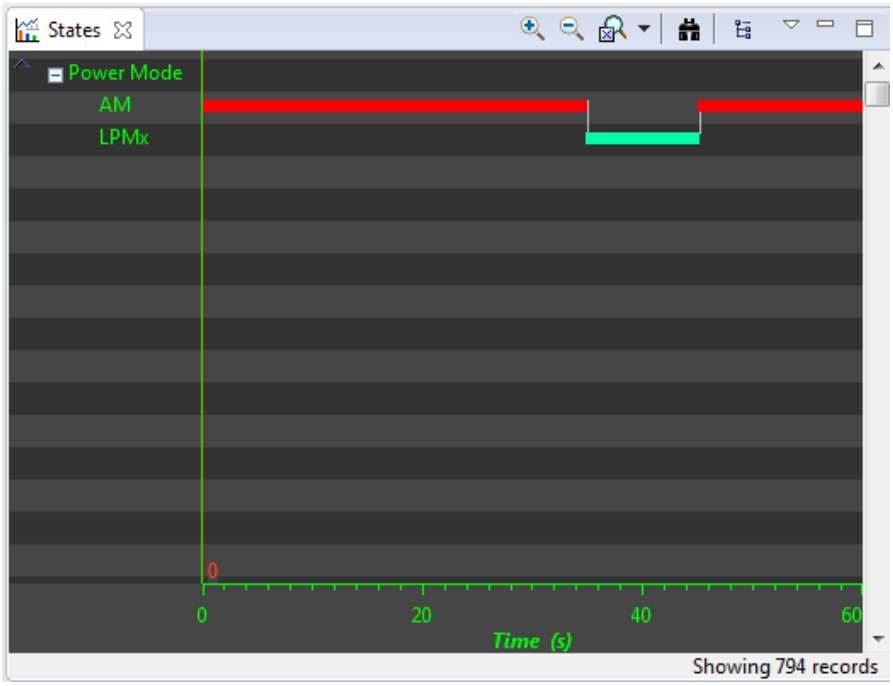

Figure 20. States Window

The **Power** window (see Figure 21) shows the dynamic power consumption of the target over time. The current profile is plotted in light blue color, while a previously recorded profile that has been reloaded for comparison is plotted in yellow color.

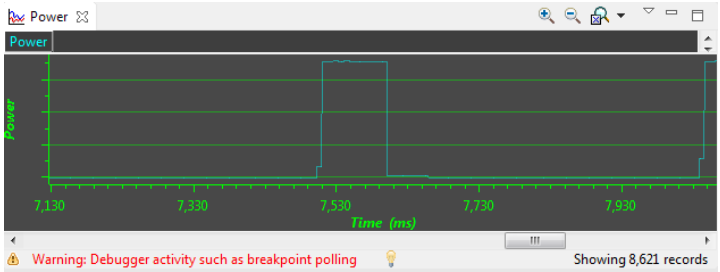

Figure 21. Power Window

Copyright © 2015–2017, Texas Instruments Incorporated

The **Energy** window (see Figure 22) shows the accumulated energy consumption of the target over time. The current profile is plotted in light blue color, while a previously recorded profile that has been reloaded for comparison is plotted in yellow color.

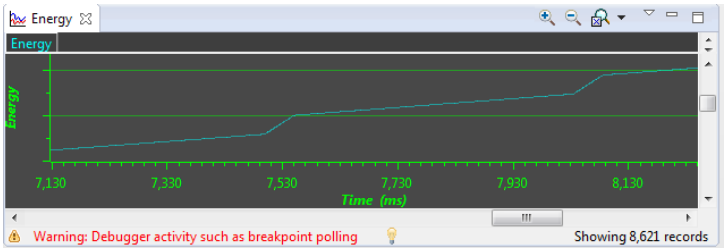

Figure 22. Energy Window

NOTE: During the capture of the internal states, the target microcontroller is constantly accessed by the JTAG or Spy-Bi-Wire debug logic. These debug accesses consume energy; therefore, no absolute power numbers are shown on the Power and Energy graph vertical axis. To see absolute power numbers of the application, it is recommended to use the **EnergyTrace mode** in combination with the **Free Run** option. In this mode, the debug logic of the target microcontroller is not accessed while measuring energy consumption.

Free Run mode is nonintrusive to the device but still uses the debug port, which affects energy measurement. For highest accuracy, run EnergyTrace without debugging or Free Run. Also note that the graphs for energy and power are printed in blue instead of green when no accurate measurement is possible.

Copyright © 2015–2017, Texas Instruments Incorporated

7.6 EnergyTrace Mode

This mode allows a standalone use of the energy measurement feature with MSP432 microcontrollers that do not have built-in EnergyTrace+ support. It can also be used to verify the energy consumption of the application without debugger activity. If the **EnergyTrace** mode is selected in the Preferences window, the following windows open when a debug session starts (see Figure 23):

- Profile
- Power
- Energy

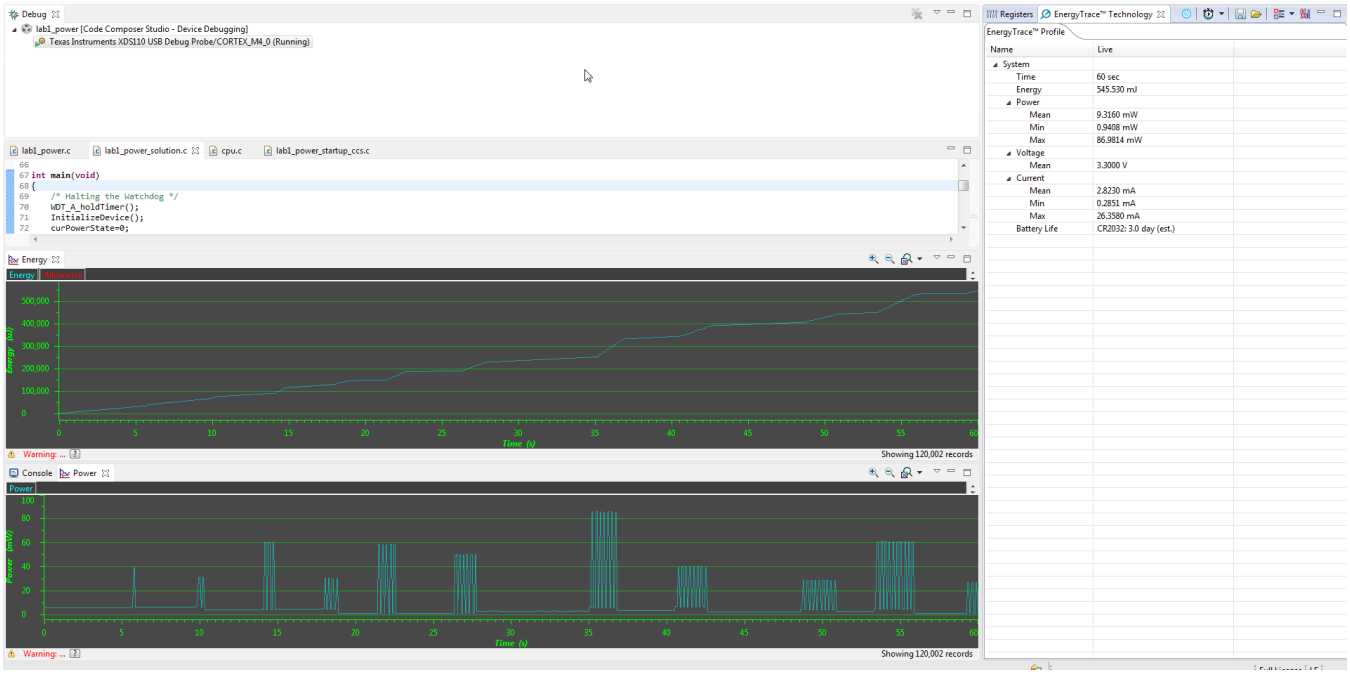

Figure 23. Debug Session With EnergyTrace Graphs

In the **EnergyTrace** mode, the **Profile** window shows statistical data about the application that has been profiled (see Figure 24). The following parameters are shown:

- Captured time
- Total energy consumed by the application (in mJ)
- Minimum, mean, and maximum power (in mW)
- Mean voltage (in V)
- Minimum, mean, and maximum current (in mA)
- Estimated life time of a CR2032 battery (in days) for the captured energy profile

NOTE: The formula to calculate the battery life time assumes an ideal 3-V battery and does not account for temperature, aging, peak current, and other factors that could negatively affect battery capacity. Changing the target voltage (for example, from 3.6 V to 3 V) might cause the analog circuitry to behave differently and operate in a more or less efficient state, hence reducing or increasing energy consumption. The value shown in the Profile window cannot substitute measurements on real hardware.

Copyright © 2015–2017, Texas Instruments Incorporated

EnergyTrace™ Profile

Name	Live
System	
Time	10 sec
Energy	14.61 mJ
Power	
Mean	1.75 mW
Min	1.068 mW
Max	7.943 mW
Voltage	
Mean	3.59 V
Current	
Mean	0.49 mA
Min	0.298 mA
Max	2.213 mA
Battery Life	CR2032: 18.8 day (est.)

Figure 24. EnergyTrace Profile Window

The **Power** window (see Figure 25) shows the dynamic power consumption of the target over time. The current profile is plotted in light blue color, while a previously recorded profile that has been reloaded for comparison is plotted in yellow color.

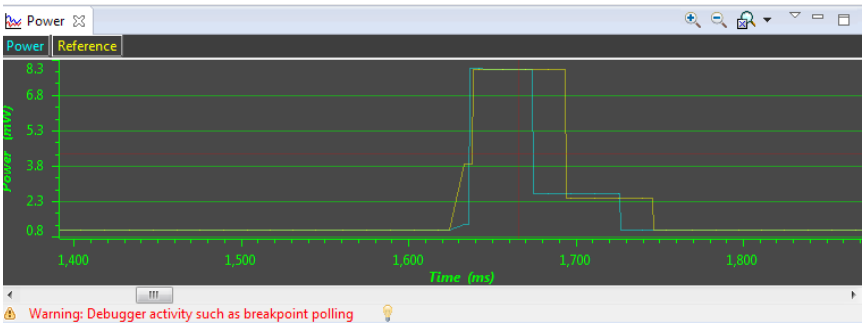

Figure 25. Zoom Into Power Window

The **Energy** window (see Figure 26) shows the accumulated energy consumption of the target over time. The current profile is plotted in light blue color, while a previously recorded profile that has been reloaded for comparison is plotted in yellow color.

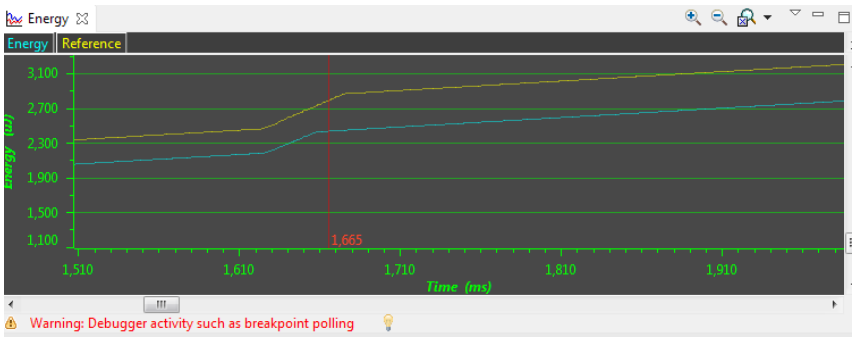

Figure 26. Current Profile (Blue) With Recorded Profile (Yellow)

Copyright © 2015–2017, Texas Instruments Incorporated

NOTE: During program execution through the debugger's view **Resume** button, the target microcontroller is constantly accessed by the JTAG or Spy-Bi-Wire protocol to detect when a breakpoint has been hit. Inevitably, these debug accesses consume energy in the target domain and change the result shown in both Energy and Power graphs. To see the absolute power consumption of an application, TI recommends using the **Free Run** mode. In Free Run mode, the debug logic of the target microcontroller is not accessed. See Figure 27 for an example of the effect of energy consumption coming from debug accesses. The yellow profile was recorded in **Resume** mode, and the green profile was recorded in **Free Run** mode.

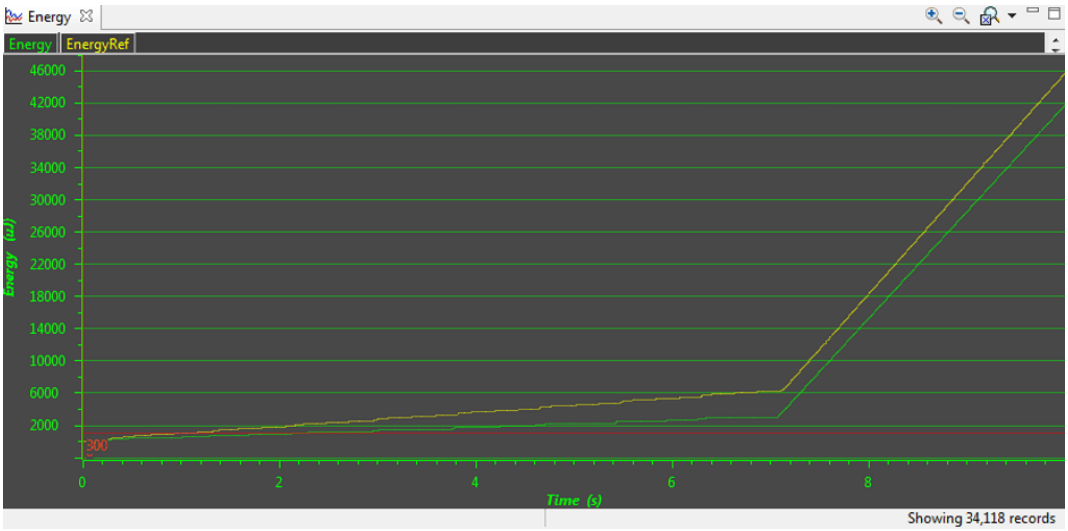

Figure 27. Energy Profile of the Same Program in Resume (Yellow Line) and Free Run (Green Line)

7.7 Comparing Captured Data With Reference Data

The EnergyTrace technology can be used in various ways. One is to check the device's internal states over time against the expected behavior and correct any misbehavior; for example, due to a peripheral not being disabled after periodic usage. Another way is to compare the captured data against previously captured data. The previously captured data is called the *reference data* in the following discussion.

After the reference data has been loaded, a yellow reference graph is plotted in the Power and Energy windows. The Power window shows the power profiles of both data sets over time and is useful to determine any changes in static power consumption; for example, due to use of a deeper low-power mode or disabling of unused peripherals. It also shows how the dynamic power consumption has changed from one measurement to the other; for example, due to ULP Advisor hints being implemented. The Energy window shows the accumulated energy consumption over time and gives an indication which profile is more energy efficient.

In the **EnergyTrace+** mode, the condensed view of both captured and reference data is displayed in the Profile window (see Figure 28). You can quickly see how the overall energy consumption and use of power modes, peripherals, and clocks changed between both capture sessions. In general, parameters that have become better are shown with a green bar, and parameters that have become worse are shown with a red bar. For example, time spent in Active Mode is generally seen as negative. Hence, if a code change makes the application spend less time in active mode, the negative delta is shown as a green bar, and the additional time spent in a low-power mode is shown as a green bar.

Copyright © 2015–2017, Texas Instruments Incorporated

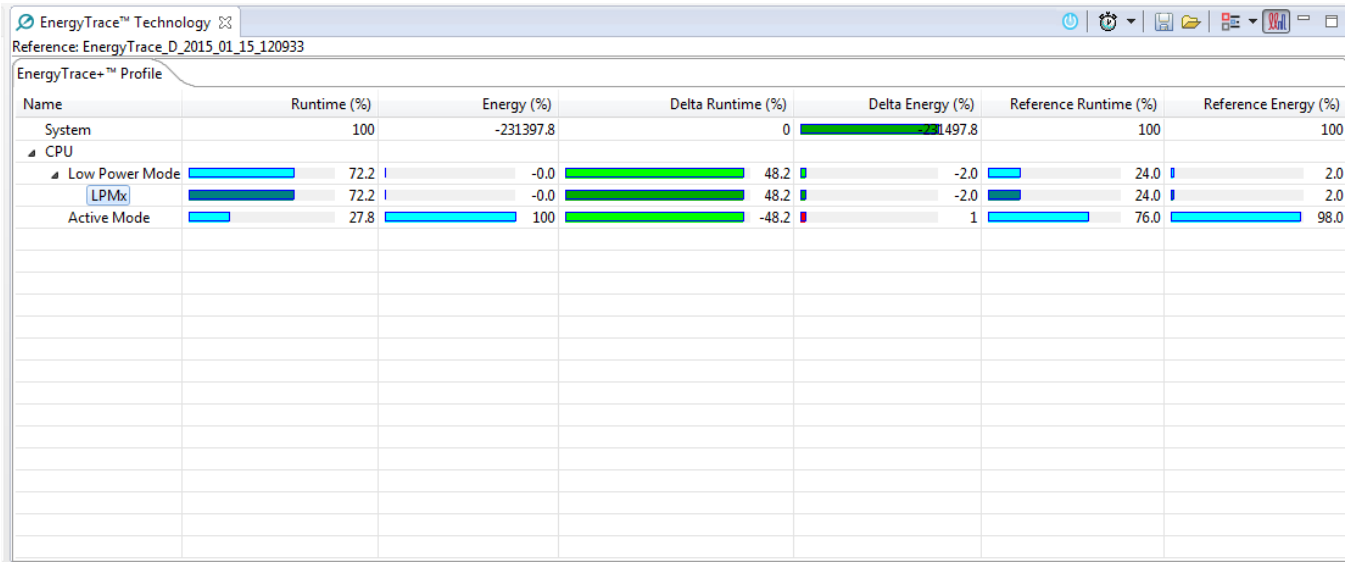

Figure 28. Comparing Profiles in EnergyTrace+ Mode

In the **EnergyTrace** mode, no States information is available to generate an exhaustive report. However, the overall energy consumed during the measurement is compared and, with it, the Min, Mean, and Max values of power and current. Parameters that have become better are shown with a green bar, and parameters that have become worse are shown with a red bar (see Figure 29).

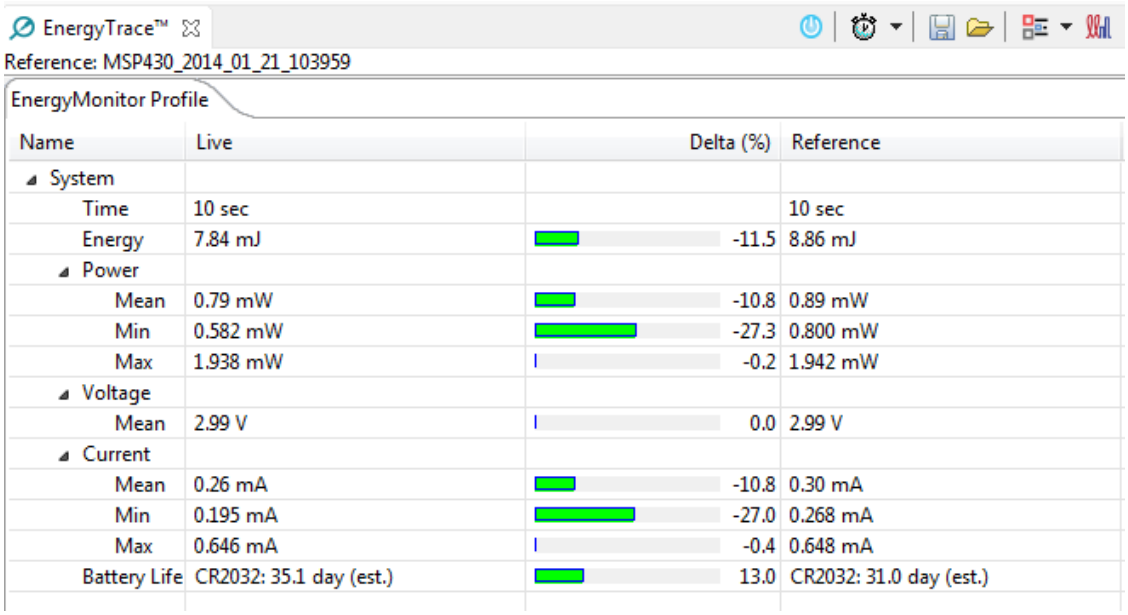

Figure 29. Comparing Profiles in EnergyTrace Mode

The delta bars are drawn linearly from 0% to 50%. Deltas larger than 50% do not result in a larger delta bar.

Copyright © 2015–2017, Texas Instruments Incorporated

7.8 EnergyTrace Technology FAQs

Q: What is the sampling frequency of EnergyTrace+ technology?

A: The sampling frequency depends on the debugger and the selected debug protocol and its speed setting. It typically ranges from 1 kHz (for example, when using the Spy-Bi-Wire interface set to SLOW) up to 3.2 kHz (for example, when using the JTAG interface set to FAST). The debugger polls the state information of EnergyTrace+ from the device status information. Depending on the sampling frequency, a short or fast duty cycle active peripheral state may not be captured on the State graph. In addition, the higher sampling frequency affects the device energy consumption under EnergyTrace.

Q: What is the sampling frequency of EnergyTrace technology?

A: The sampling frequency to measure the energy consumption is the same independent of which debug protocol or speed and is approximately 4.2 kHz in Free Run mode.

Q: My Power graph seems to include noise. Is my board defective?

A: The power values shown in the Power graph are derived (that is, calculated) from the accumulated energy counted by the measurement system. When the target is consuming little energy, a small number of energy packets over time are supplied to the target, and the software needs to accumulate the dc-dc charge pulses over time before a new current value can be calculated. For currents under 1 µA, this can take up to one second, while for currents in the milliamp range, a current can be calculated every millisecond. Additional filtering is not applied so that detail information is not lost. Another factor that affects the energy (and with it, the current) that is consumed by the target is periodic background debug access during normal code execution, either through capturing of States information or through breakpoint polling. Try recording in Free Run mode to see a much smoother Power graph.

Q: I have a code that repeatedly calls functions that have the same size. I would expect the function profile to show an equal distribution of the run time. In reality, I see some functions having slightly more run time than expected, and some functions slightly less.

A: During program counter trace, various factors affect the number of times a function is detected by the profiler over time. The microcontroller code could benefit from the internal cache, thus executing some functions faster than others. Another influencing factor is memory wait states and CPU pipeline stalls, which add time variance to the code execution. An outside factor is the sampling frequency of the debugger itself, which normally runs asynchronous to the microcontroller's code execution speed, but in some cases shows overlapping behavior, which also results in an unequal function run time distribution.

Q: My profile sometimes includes an <Undetermined> low-power mode, and there are gaps in the States graph Power Mode section. Where does the <Undetermined> low-power mode originate from?

A: During transitions from active mode to low-power mode, internal device clocks are switched off, and occasionally the state information is not updated completely. This state is displayed as <Undetermined> in the Profile window, and the States graph shows a gap during the time that the <Undetermined> low-power mode persists. The <Undetermined> state is an indication that the application has entered a low-power mode, but which mode cannot be accurately determined. If the application is frequently entering low-power modes, the <Undetermined> state will probably be shown more often than if the application only rarely uses low-power modes.

Q: When capturing in EnergyTrace mode, the min and max values for power and current show deviation, even though my program is the same. I would expect absolutely the same values.

A: The energy measurement method used on the hardware counts dc-dc charge pulses over time. Energy and power are calculated from the energy over time. Due to statistical sampling effects and charge and discharge effects of the output voltage buffer capacitors, it is possible that minimum and maximum values of currents vary by some percent, even though the program is identical. The captured energy, however, should be almost equal (in the given accuracy range).

Copyright © 2015–2017, Texas Instruments Incorporated

Q: What are the influencing factors for the accuracy of the energy measurement?

A: The energy measurement circuit is directly supplied from the USB bus voltage, and thus it is sensitive to USB bus voltage variations. During calibration, the energy equivalent of a single dc-dc charge pulse is defined, and this energy equivalent depends on the USB voltage level. To ensure a good repeatability and accuracy, power the debugger directly from an active USB port, and avoid using bus-powered hubs and long USB cables that can lead to voltage drops, especially when other consumers are connected to the USB hub. Furthermore the LDO and resistors used for reference voltage generation and those in the calibration circuit come with a certain tolerance and ppm rate over temperature, which also influences accuracy of the energy measurement.

Q: I am trying to capture in EnergyTrace+ mode or EnergyTrace mode with a MSP432 device that is externally powered, but there is no data shown in the Profile, Energy, Power and States window.

A: Both EnergyTrace+ mode and EnergyTrace mode require the target to be supplied from the debugger. No data can be captured when the target microcontroller is externally powered.

Q: I cannot measure LPM currents when I am capturing in EnergyTrace+ mode. I am expecting a few microamps but measure more than 150 µA.

A: Reading digital data from the target microcontroller consumes energy in the JTAG domain of the microcontroller. Hence, an average current of approximately 150 µA is measured when connecting an ampere meter to the device power supply pins. If you want to eliminate energy consumption through debug communication, switch to EnergyTrace mode, and let the target microcontroller execute in Free Run mode.

Q: My LPM currents seem to be wrong. I am expecting a few microamps, but measure more, even in Free Run mode or when letting the device execute without debug control from an independent power supply.

A: The most likely cause of this extra current is improper GPIO termination, as floating pins can lead to extra current flow. Also check the JTAG pins again, especially when the debugger is still connected (but idle), as the debugger output signal levels in idle state might not match how the JTAG pins have been configured by the application code. This could also lead to extra current flow.

Q: When I start the EnergyTrace+ windows through View → Other → EnergyTrace before launching the debug session, data capture sometimes does not start.

A: Enable EnergyTrace through Window → Preferences → Code Composer Studio → Advanced Tools → EnergyTrace™ Technology. When launching a debug session, the EnergyTrace+ windows automatically open, and data capture starts when the device executes. If you accidentally close all EnergyTrace+ windows during a debug session, you can reopen them through View → Other → EnergyTrace.

Copyright © 2015–2017, Texas Instruments Incorporated

8 Device Security

On newer MSP432 device variants it is possible to protect the factory reset command with a password. If you have done this skip the following chapter and continue with “Factory reset with password”

8.1 Factory Reset Without Password

If you have disabled JTAG access on the device or are working on an application where you need to unlock a secure IP zone, the lock can be only be removed by erasing all Flash memory, including USER and INFO memory through a debugger invoked reboot cycle. To unlock a device, the following steps are required

- Select a Target Configuration that matches the current debugger type
- Execute a script that triggers a reboot erase

These steps are explained in detail next.

Go to **View** → **Target Configurations** to see the available debugger configurations.

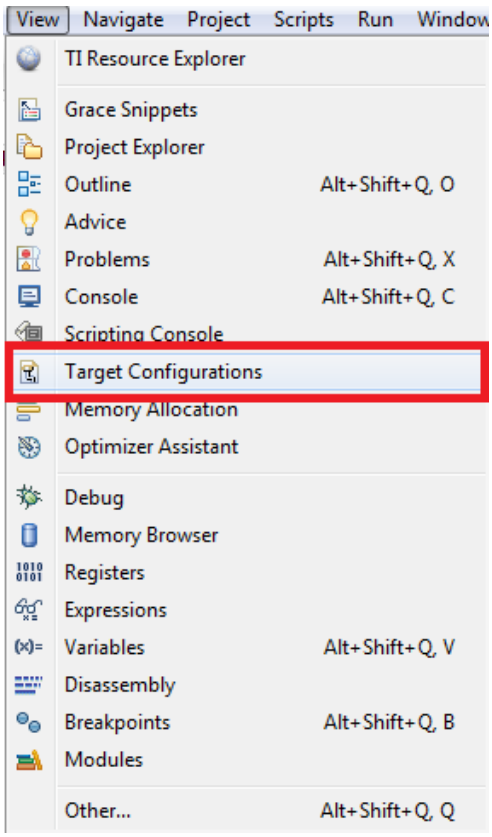

Figure 30. Show Target Configuration View

Code Composer Studio IDE opens a view that shows the target configurations it can identify in the current workspace. Pick the one that works for the device and debugger. This example uses the configuration file for an XDS100v2 debugger.

Copyright © 2015–2017, Texas Instruments Incorporated

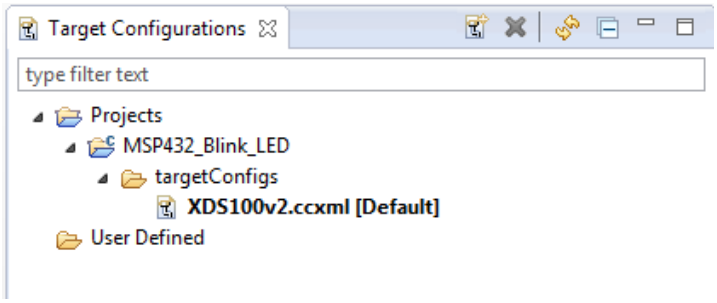

Figure 31. List of Target Configurations

Now right click on the target configuration and select **Launch Selected Configuration**.

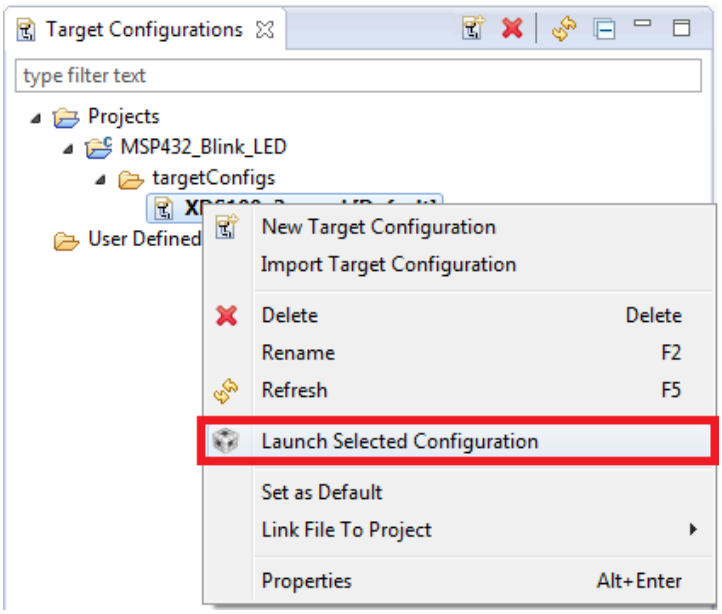

Figure 32. Launch Selected Target Configuration

The debugger now connects to the device (which is still possible), but does not try to halt the CPU, write to registers, or even download code (which would not be possible). The Debug view shows the CPU core, but marks it as disconnected.

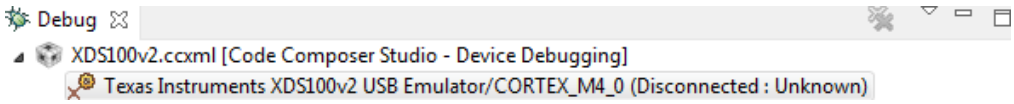

Figure 33. Debug View After Launching Target Configuration

To access the Debugger Access Port, or DAP, right-click on the highlighted line that shows the CPU core, and select **Show all cores** from the drop down menu.

Copyright © 2015–2017, Texas Instruments Incorporated

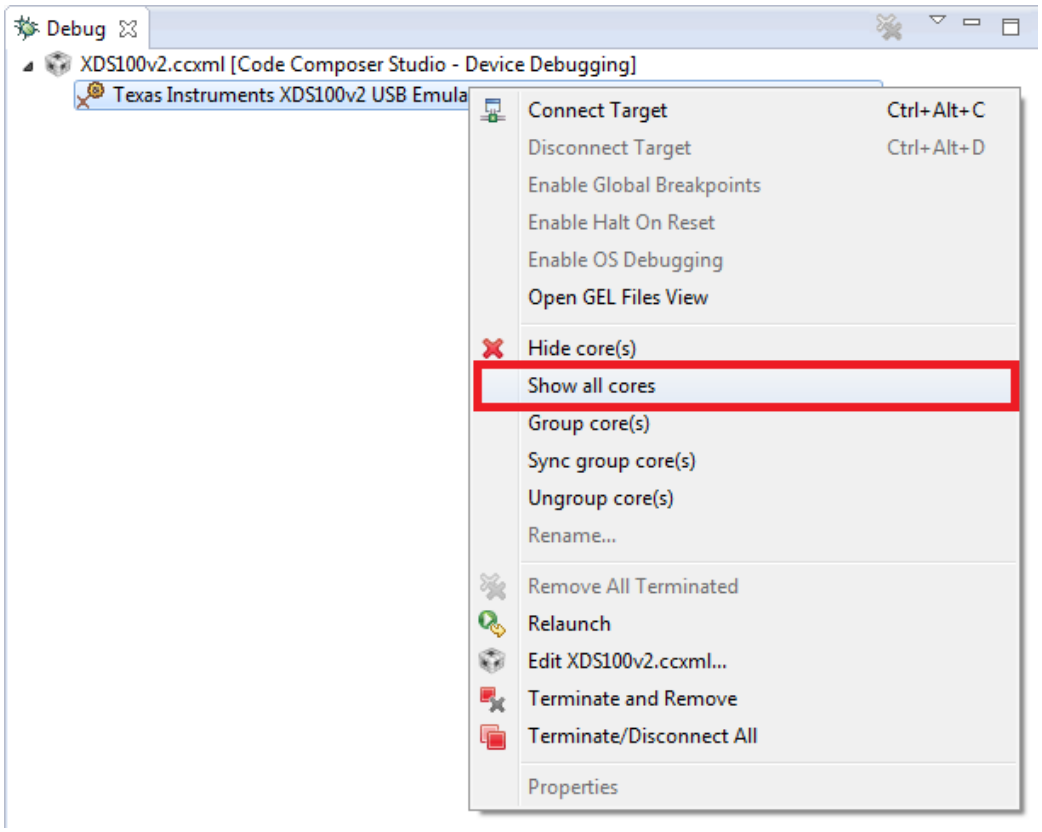

Figure 34. Show All Cores

The MSP432 Debug Access Port, or DAP, is now listed under **Non Debuggable Devices**.

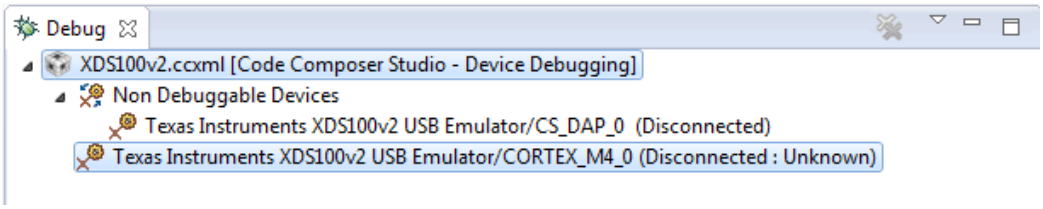

Figure 35. List of All Cores in the MSP432

Now right-click on the DAP and select **Connect Target**.

Copyright © 2015–2017, Texas Instruments Incorporated

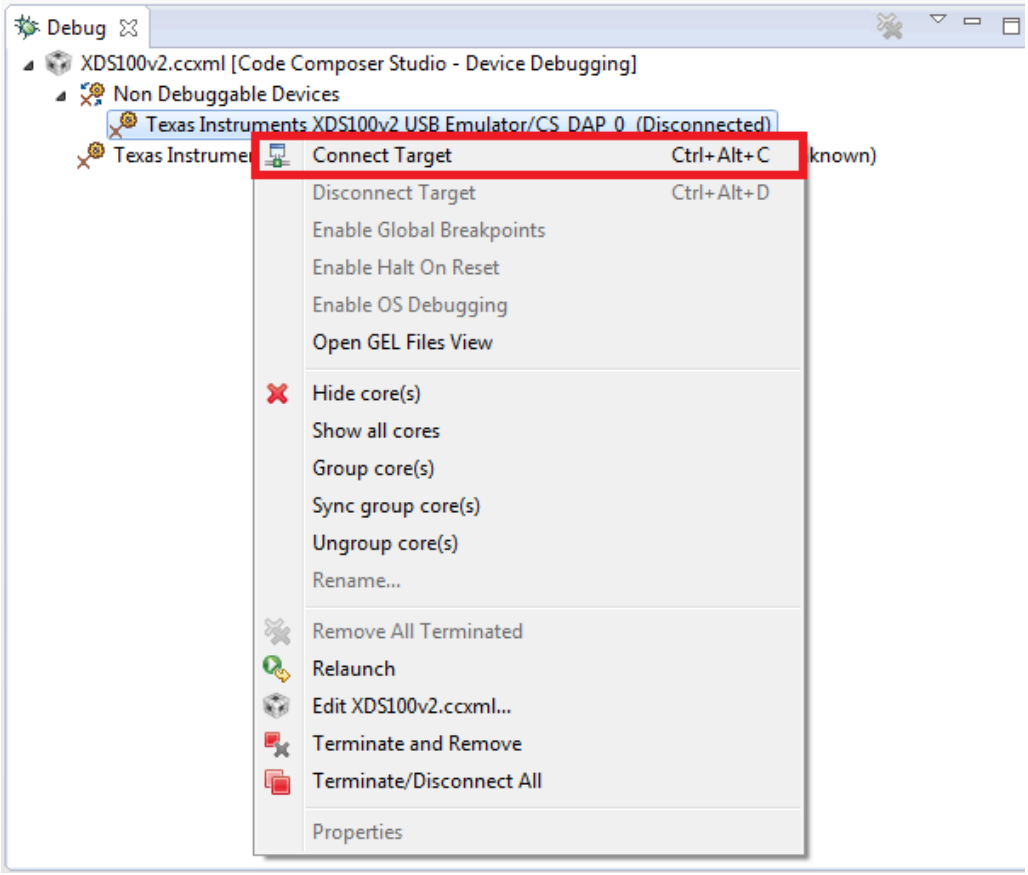

Figure 36. Manually Connecting to the DAP

When the debugger has connected to the DAP, the Debug view window changes, indicating a successfully connected DAP.

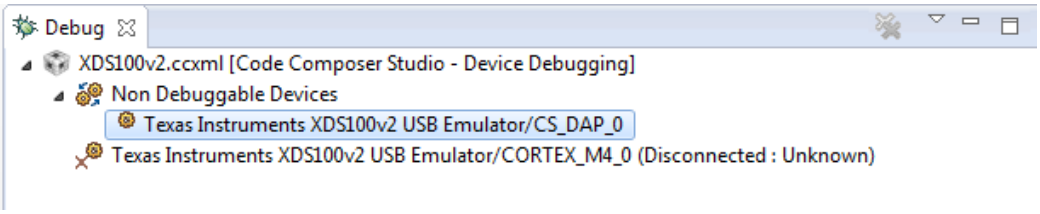

Figure 37. DAP is Connected

Now you need to execute a Code Composer Studio IDE script that performs triggers a reboot reset through the DAP. Click **Scripts** → **default** → **MSP432_Factory_Reset**.

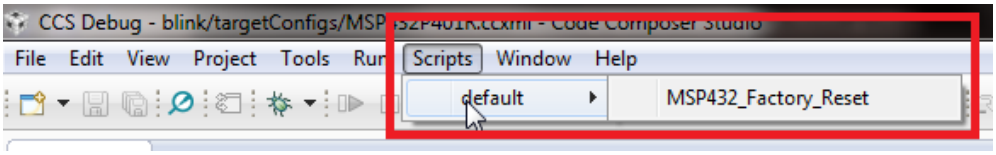

Figure 38. Executing the Factory Reset Script

As the script is executed, the Console window shows that the mass erase has been executed. Now you can terminate the debug connection. After you have power cycled the device, it is accessible in a normal way again.

Copyright © 2015–2017, Texas Instruments Incorporated

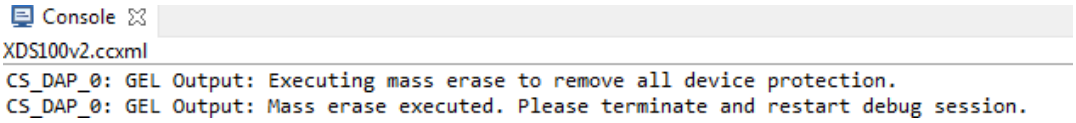

Figure 39. Mass Erase Script Console Output

8.2 Factory Reset With Password

If you have enabled password protection for the factory reset and disabled JTAG access on the device or are working on an application where you need to unlock a secure IP zone, the lock can be removed only by erasing all flash memory, including USER and INFO memory, through a debugger-invoked reboot cycle.

Precondition: Factory reset with password must have been configured using the flash mailbox, either with the help of the security and update tool (see Reference [6]) or by manipulating the flash mailbox with an application.

To unlock a password protected device, the following steps are required:

1. Open ccs_base/emulation/gel/msp432_factory_reset_password.gel and enter the password in the user section.
2. Launch Code Composer Studio IDE debug session or target configuration and connect to the device.
3. Open Tools → GEL Files.

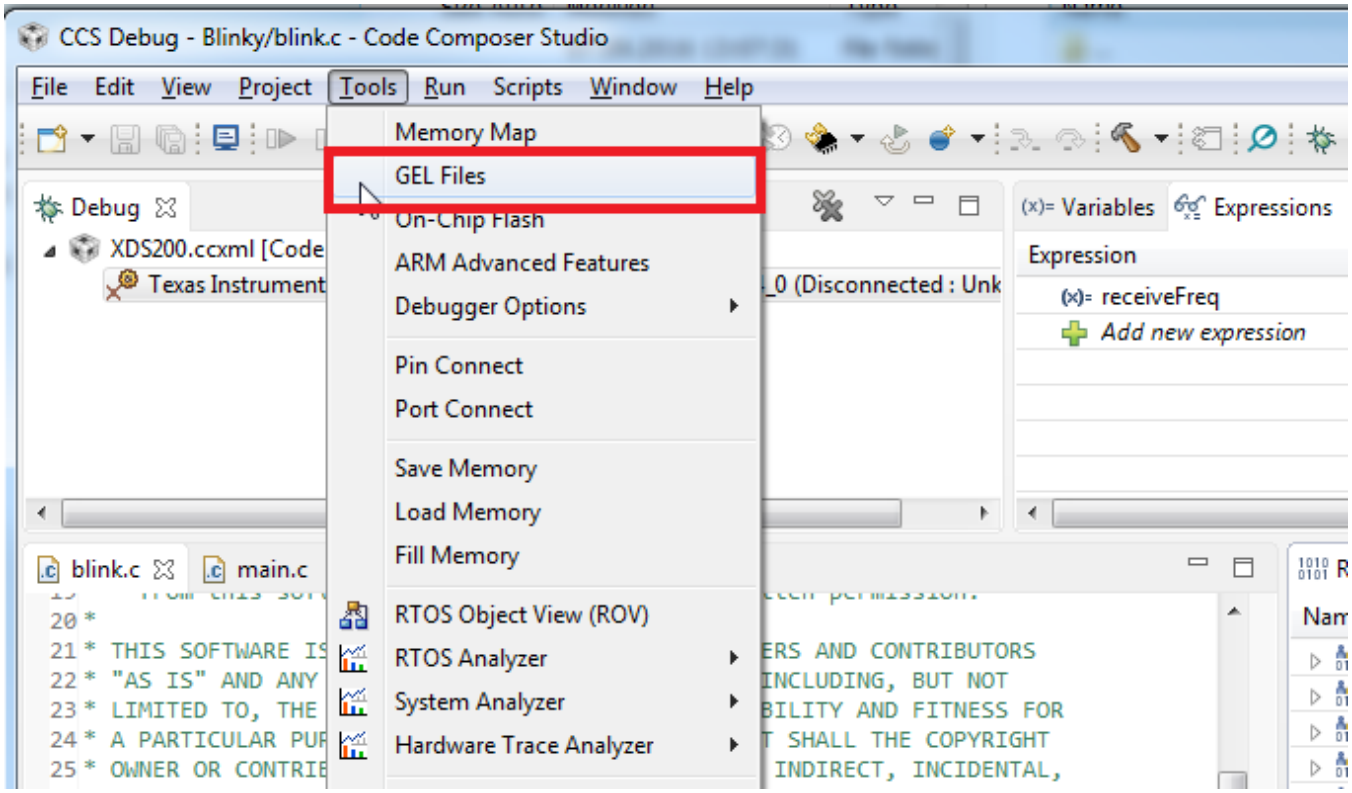

Figure 40. Code Composer Studio IDE Tools – GEL File

4. Right click in the new window and select "Load GEL...".
5. Browse for msp432_factory_reset_password.gel and click Open.
6. To run the factory reset with password, select Scripts → default → MSP432_FACTORY_RESET_PASSWORD

Copyright © 2015–2017, Texas Instruments Incorporated

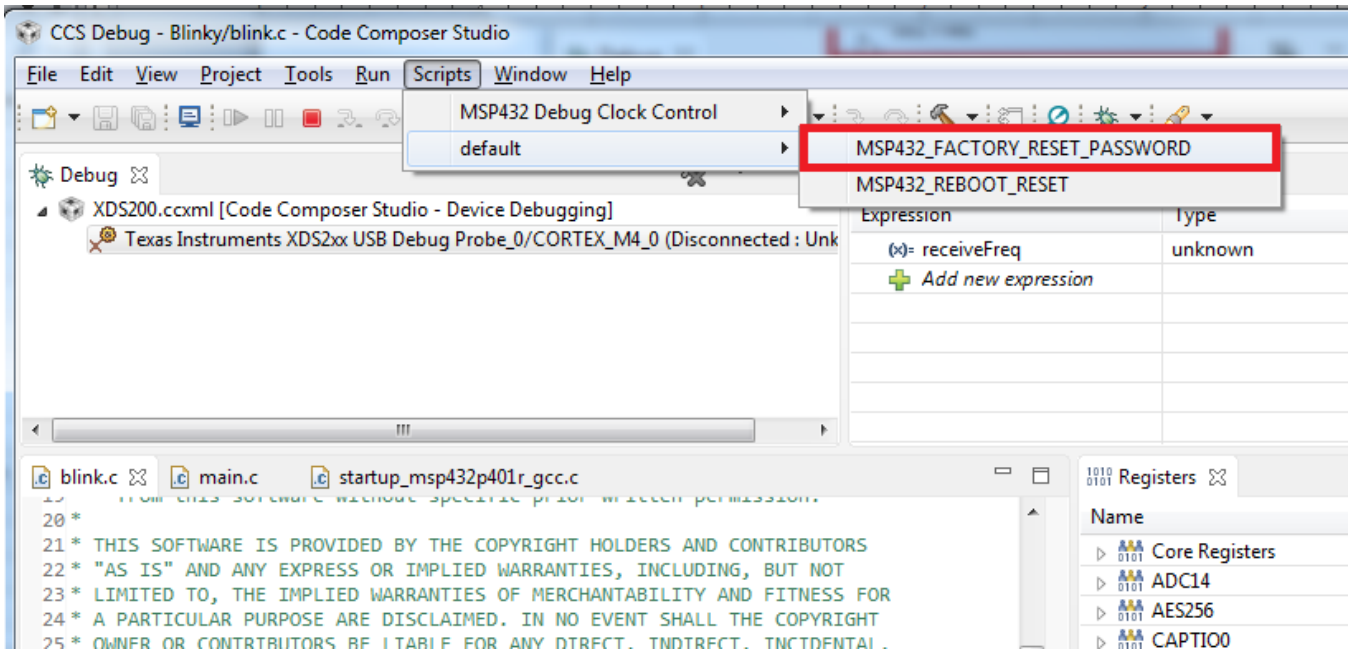

Figure 41. Factory Reset With Password GEL File

As the script is executed, the console window shows that the mass erase is being executed. When the erase is complete, terminate the debug connection. Cycle the power to the device to reenable access to it.

NOTE: The factory reset with password command is a one-time password. When the password is sent correctly, a factory reset is conducted, which resets EVERY security setting including the factory reset with password settings.

The GEL file is available with MSP432 device support files 6.3.1.x and higher.

9 Low-Power Debug

Under normal debug control, MSP432 microcontrollers do not transition into low-power modes deeper than LPM0 mode; that is, into either LPM0_VCORE0 or LPM0_VCORE1 mode. This behavior is a consequence of the standardized Cortex-M debug architecture. Therefore, current consumption and IRQ wake-up timing are different from a free-running application.

To verify the current consumption and timing of an application while still under basic debug control, MSP432 offers the **Low Power Run** feature. When enabled, the MCU transitions into exactly the low-power mode that the application specifies, with internal clocks disabled and the power management module shutting down internal power domains. This has some implications:

- Previously set breakpoints are reached, but the IDE does not automatically indicate that the device has halted. The user must click the halt icon to enable the IDE to reconnect and show where the program has halted.
- Auto Run after loading a program does not work: The breakpoint that is set automatically by the IDE [for example, at main()] is reached, but the IDE does not switch to halt automatically. When the user halts manually after program load, the program counter is at start of main().
- SWO trace does not work when transitioning into power modes lower than AM0_SL or AM1_SL mode

To enable the feature in Code Composer Studio IDE, right click on the active project in the Project Explorer and click on **Properties**.

Copyright © 2015–2017, Texas Instruments Incorporated

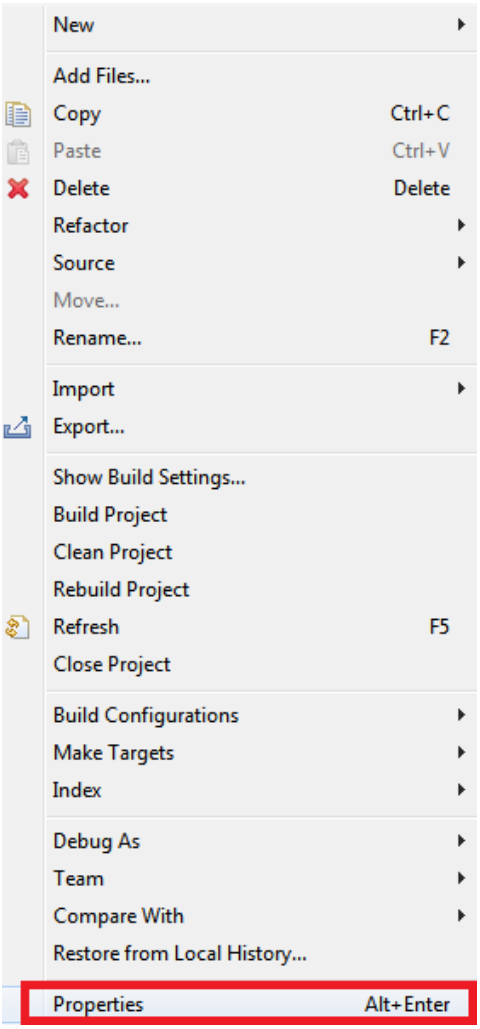

Figure 42. Properties Menu

In the Properties window, select **Debug**, then go to **Misc/Other Options**, and enable the **Allow power transitions while running if supported (low power running)** option.

Copyright © 2015–2017, Texas Instruments Incorporated

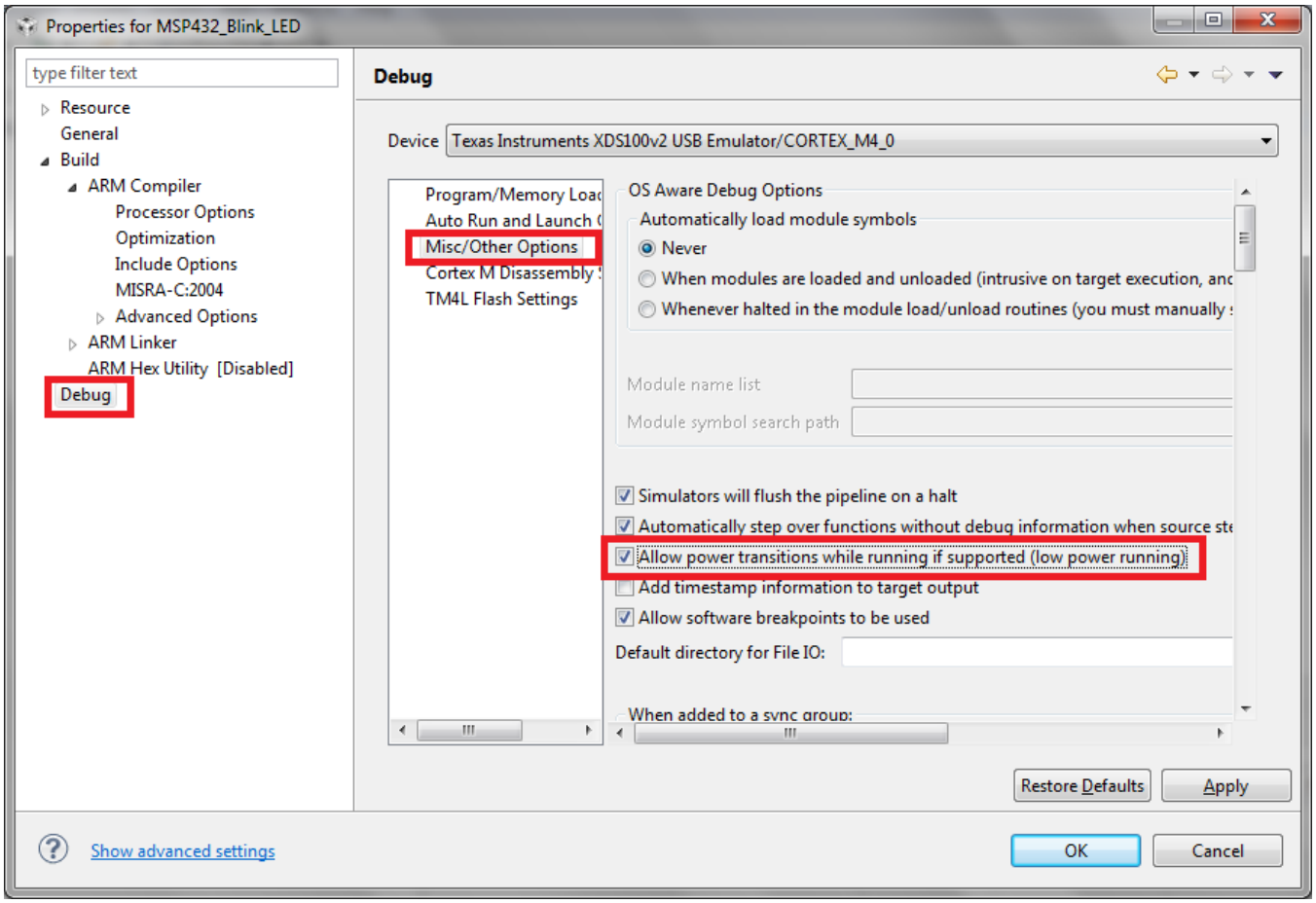

Figure 43. Enabling Low Power Run

When Low Power Run is enabled, the MCU will go to any low-power mode specified. You can verify the effect by watching the CPU core status display during a debug session. When the MCU goes to a low-power mode equal or deeper than DSL, the debugger will report a loss of connection, due to the MSP432 clocks all being disabled, including the clock that operates the Debug Access Port (DAP).

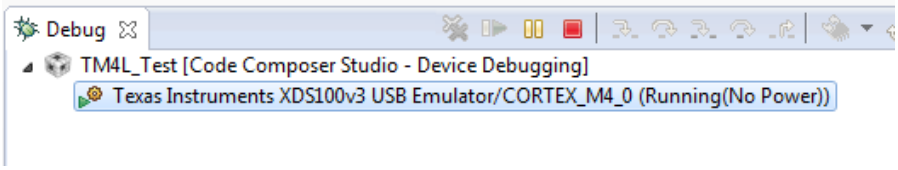

Figure 44. CPU Core Status Display Indicating Deep Sleep Mode

When halting the device, the debugger will reconnect, and the IDE will show the current program counter location, which in this case will be the WFI assembly instruction that sent the MSP432 to sleep.

Copyright © 2015–2017, Texas Instruments Incorporated

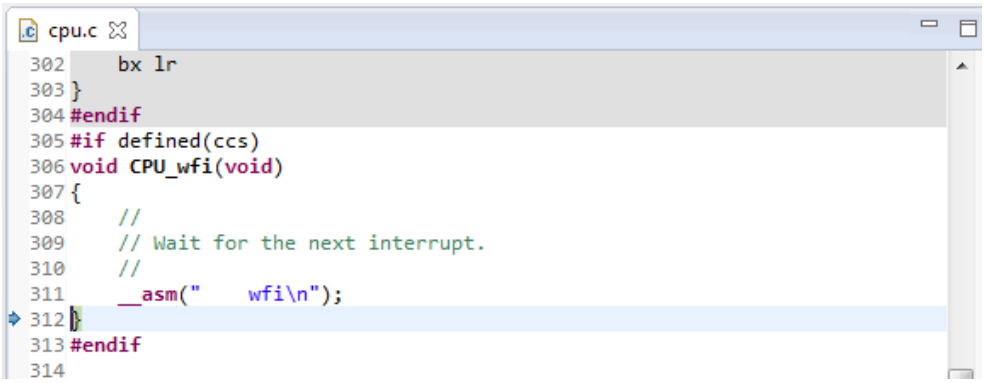

Figure 45. Program Counter Located at WFI Instruction

10 Frequently Asked Questions

Q: I cannot program my LaunchPad™ kit; the IDE cannot connect to target. What's wrong?

A: Check the following:

- Is the JTAG switch (S101) in the correct orientation?

  Switch to left for XDS110-ET onboard debugger

  Switch to the right for external debugger connection
- Check the debugger settings: change to *SWD Mode – Aux COM port is target TDO pin*. When the settings of Port J (PJSEL0 and PJSEL1 bits) are changed, full JTAG access is prevented on these pins. Changing to use SWD allows access through the dedicated debug pins only. Figure 46 shows how to configure the debugger to use SWD instead of JTAG by modifying the MSP432P401R.ccxml file.

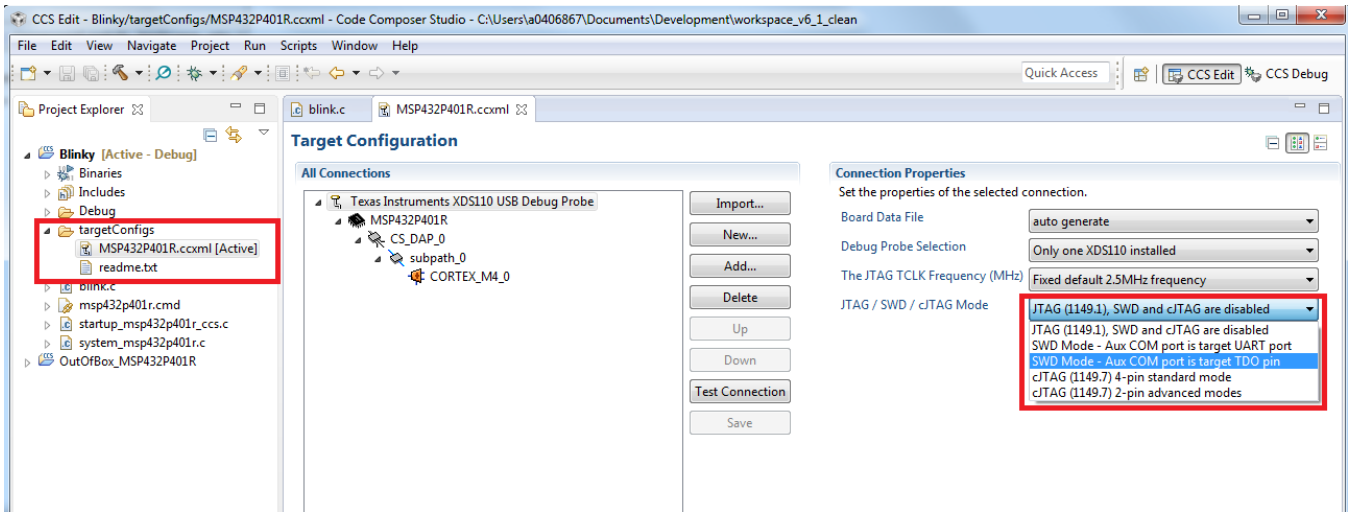

Figure 46. Change Debugger Settings to SWD

- If even this cannot connect, reset the device to factory settings. Review Section 8 for information on how to perform a factory reset on the device.

Copyright © 2015–2017, Texas Instruments Incorporated

Q: Why doesn't the backchannel UART on the MSP432 LaunchPad work with my serial terminal program at speeds faster than 56000 baud?

A: Certain serial terminal programs such as HTerm or the Code Composer Studio IDE built-in terminal might not work with the MSP432 LaunchPad at specific baud rates, resulting in the software not being able to open the virtual COM port or in the baud rate getting configured incorrectly. An issue with the LaunchPad's emulator firmware has been identified and will be fixed in the next release. Until the update is available, use Tera Term, ClearConnex, or HyperTerminal instead or reduce the baud rate to speeds of 38400 baud or lower.

Q: Problems plugging the MSP432 LaunchPad into a USB3.0 Port

A: It has been observed that when the MSP432 LaunchPad is connected to USB3.0 ports provided by a certain combination of USB3.0 host controller hardware and associated device drivers that the IDE is unable to establish a debug session with the LaunchPad, resulting in an error message like "CS_DAP_0: Error connecting to the target: (Error -260 @ 0x0) An attempt to connect to the XDS110 failed" in the case of Code Composer Studio. In this case the Code Composer Studio IDE-provided low-level command line utility ‘xdsdfu' will also not be able to establish a connection with the LaunchPad.

Specifically, this issue was observed on PCs running Windows 7 that show the "Renesas Electronics USB 3.0 Host Controller" and the associated "Renesas Electronics USB 3.0 Root Hub" in the device manager. After updating the associated Windows USB drivers to more recent versions obtained from the hardware vendor the issue went away. There might be other USB3.0 hardware and device driver combinations that will lead to the same issue. If you think you might be affected, contact the PC vendor or locate and install more recent versions of the USB3.0 device drivers. Alternatively, connect the LaunchPad to an USB2.0 port on the PC if available.

Q: I cannot get the backchannel UART to connect. What's wrong?

A: Check the following:

- Do the baud rate in the host's terminal application and the eUSCI settings match?
- Are the appropriate jumpers in place on the isolation jumper block?
- Probe on RXD and send data from the host. If you don't see data, it might be a problem on the host side.
- Probe on TXD while sending data from the MSP432. If you don't see data, it might be a configuration problem with the eUSCI module.
- Consider the use of the hardware flow control lines (especially for higher baud rates).

Q: How can I easily unlock the SYS_CTL register block?

A: For the TI ARM compiler, there is a macro define available for doing so. See `#define UNLOCK_DEVICE` in the corresponding device header.

11 Additional Code Composer Studio IDE Information

For more information about Code Composer Studio IDE, see the following links:

- Code Composer Studio Information
- Code Composer Studio v7 Training
- Code Composer Studio v7 Wiki

Copyright © 2015–2017, Texas Instruments Incorporated

12 References

1. SimpleLink MSP432 SDK
2. J-Link Emulator Support
3. *MSP432™ Debugging Tools: Using Serial Wire Output With CCS Trace Analyzer*
4. MSP-FET for MSP432 Microcontrollers
5. SimpleLink MSP432 Security and Update Tool
6. XDS110 JTAG Debug Probe
7. Debuggers for MSP432 Microcontrollers
8. Migration Guide for SimpleLink MSP432 SDK

Copyright © 2015–2017, Texas Instruments Incorporated

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from September 10, 2016 to March 6, 2017 **Page**

- Changed title of this user's guide 1
- Changed the Code Composer Studio version from 6.1 to 7.1 throughout document 1
- Added "SimpleLink" branding and updated titles of referenced documents as necessary 1
- Added Code Composer Studio IDE version requirements for MSP432 and SimpleLink SDK support in Section 1, *Installing Code Composer Studio IDE* 4
- Removed former Figure 2, *Available Updates* 4
- Added the paragraph that starts "You can either create a new project..." in Section 3, *Creating a Basic SimpleLink MSP432 Project* 5
- Updated Figure 3, *New Project Wizard* 6
- Updated Figure 4, *New Project Files* 6
- Replaced former Section 4, *Additional MSP432 Examples and Documentation*, with Section 4, *Importing SimpleLink Examples From TI Resource Explorer* 7
- Added the first three paragraphs in Section 5.2, *Debugging ROM Driver Library* 12
- Updated Section 12, *References* 39

Copyright © 2015–2017, Texas Instruments Incorporated

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/sampterms.htm).

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