

Code Composer Studio™ 6.1 for MSP432

This manual describes the use of the TI Code Composer Studio™ IDE (CCS) version 6.1 with the MSP432™ low-power microcontrollers. This manual describes only CCS for Windows® operating systems. The versions of CCS for Linux® and Mac OS X® operating systems are similar and, therefore, are not described separately.

Contents

1	Installing Code Composer Studio IDE	3
2	Updating Code Composer Studio IDE	3
3	Creating an MSP432 Project	5
4	Additional MSP432 Examples and Documentation	7
5	Debugging Your Application	8
6	EnergyTrace™ Technology	11
7	Device Security	22
8	Low-Power Debug	26
9	Frequently Asked Questions	29
10	Additional Code Composer Studio Information	30
11	References	30

List of Figures

1	Check for Updates	3
2	Available Updates	4
3	Creating a New CCS Project	5
4	New Project Wizard	6
5	New Project Files	7
6	Project Properties	8
7	Choose Debugger Connection	9
8	Launch Debug Session	9
9	Debug Session	10
10	Pulse Density and Current Flow	11
11	EnergyTrace™ Technology Preferences	12
12	EnergyTrace™ Technology Control Bar	13
13	Debug Session With EnergyTrace+ Graphs	14
14	Profile Window	14
15	States Window	15
16	Power Window	15
17	Energy Window	16
18	Debug Session With EnergyTrace Graphs	16
19	EnergyTrace Profile Window	17
20	Zoom Into Power Window	17

Code Composer Studio, MSP432, E2E, LaunchPad are trademarks of Texas Instruments.
 ARM is a registered trademark of ARM Ltd.
 OS X is a registered trademark of Apple Inc.
 IAR Embedded Workbench is a trademark of IAR Systems.
 Linux is a registered trademark of Linus Torvalds.
 Windows is a registered trademark of Microsoft Corporation.

21	Current Profile (Blue) With Recorded Profile (Yellow).....	18
22	Energy Profile of the Same Program in Resume (Yellow Line) and Free Run (Green Line)	18
23	Comparing Profiles in EnergyTrace+ Mode	19
24	Comparing Profiles in EnergyTrace Mode	20
25	Show Target Configuration View	22
26	List of Target Configurations.....	23
27	Launch Selected Target Configuration	23
28	Debug View After Launching Target Configuration	23
29	Show All Cores	24
30	List of All Cores in the MSP432	24
31	Manually Connecting to the DAP.....	25
32	DAP is Connected.....	25
33	Executing the Factory Reset Script	25
34	Mass Erase Script Console Output	26
35	Properties Menu	27
36	Enabling Low Power Run	28
37	CPU Core Status Display Indicating Deep Sleep Mode	28
38	Program Counter Located at WFI Instruction.....	29
39	Change Debugger Settings to SWD	29

Read This First

How to Use This Manual

This manual describes only those CCS features that are specific to the 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 MSP432 Documents on the Web

The primary sources of MSP432 information are the device-specific data sheets and user's guides. The MSP432 web site (www.ti.com/msp432) contains the most recent version of these documents.

Documents describing 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 CCS-specific Wiki page (FAQ) is available at processors.wiki.ti.com/index.php/Category:CCS, and the Texas Instruments E2E™ Community support forums at e2e.ti.com 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 web site at www.ti.com/support. The Texas Instruments E2E Community support forums for the MSP432 at e2e.ti.com provide open interaction with peer engineers, TI engineers, and other experts. Additional device-specific information can be found on the MSP432 web site at www.ti.com/msp432.

1 Installing Code Composer Studio IDE

The Code Composer Studio IDE can be obtained from TI website under www.ti.com/tool/ccstudio. Note that MSP432 low-power microcontrollers are supported by Code Composer Studio 6.1 and higher versions. Previous versions do not support MSP432.

2 Updating Code Composer Studio IDE

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

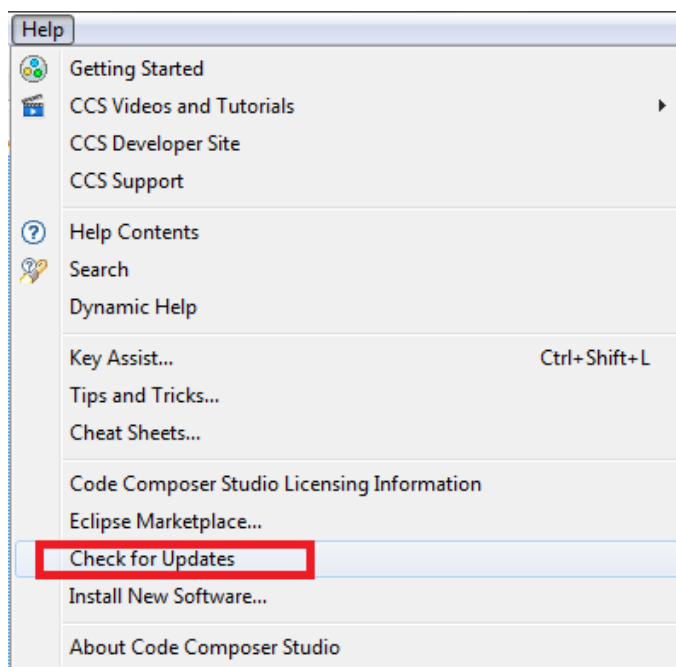


Figure 1. Check for Updates

CCS now automatically 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 (see [Figure 2](#)).

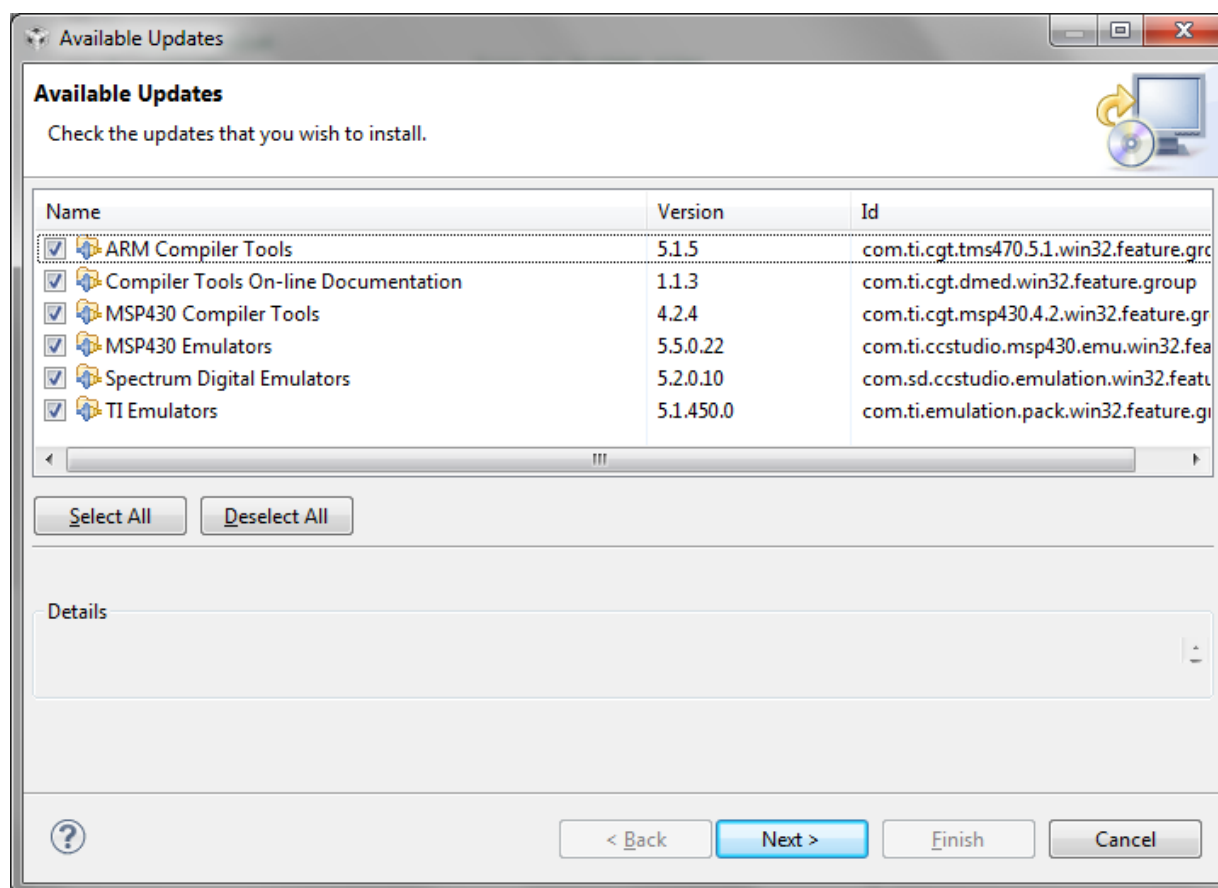


Figure 2. Available Updates

3 Creating an MSP432 Project

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

To create a new project in that workspace:

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

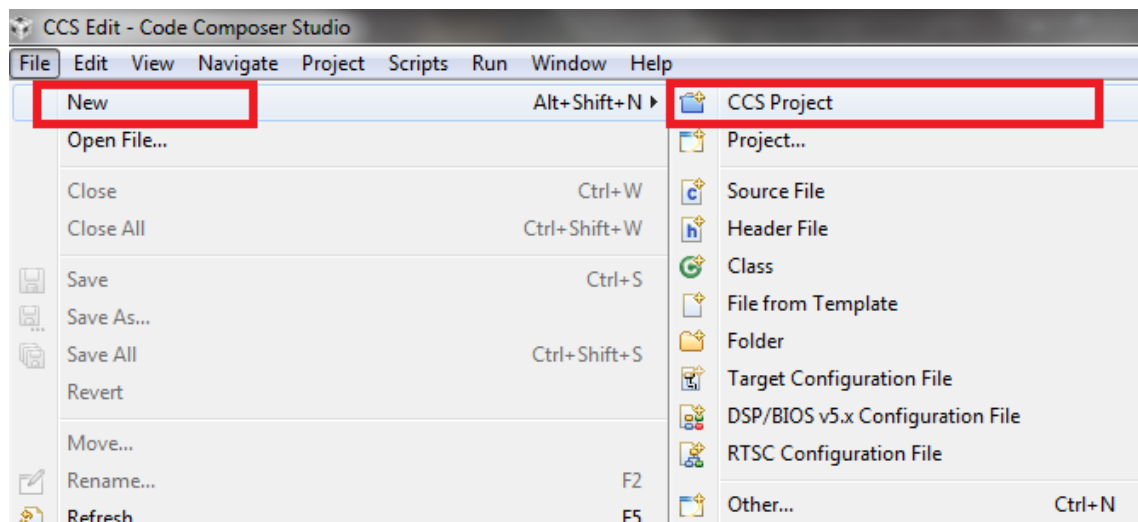


Figure 3. Creating a New CCS Project

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

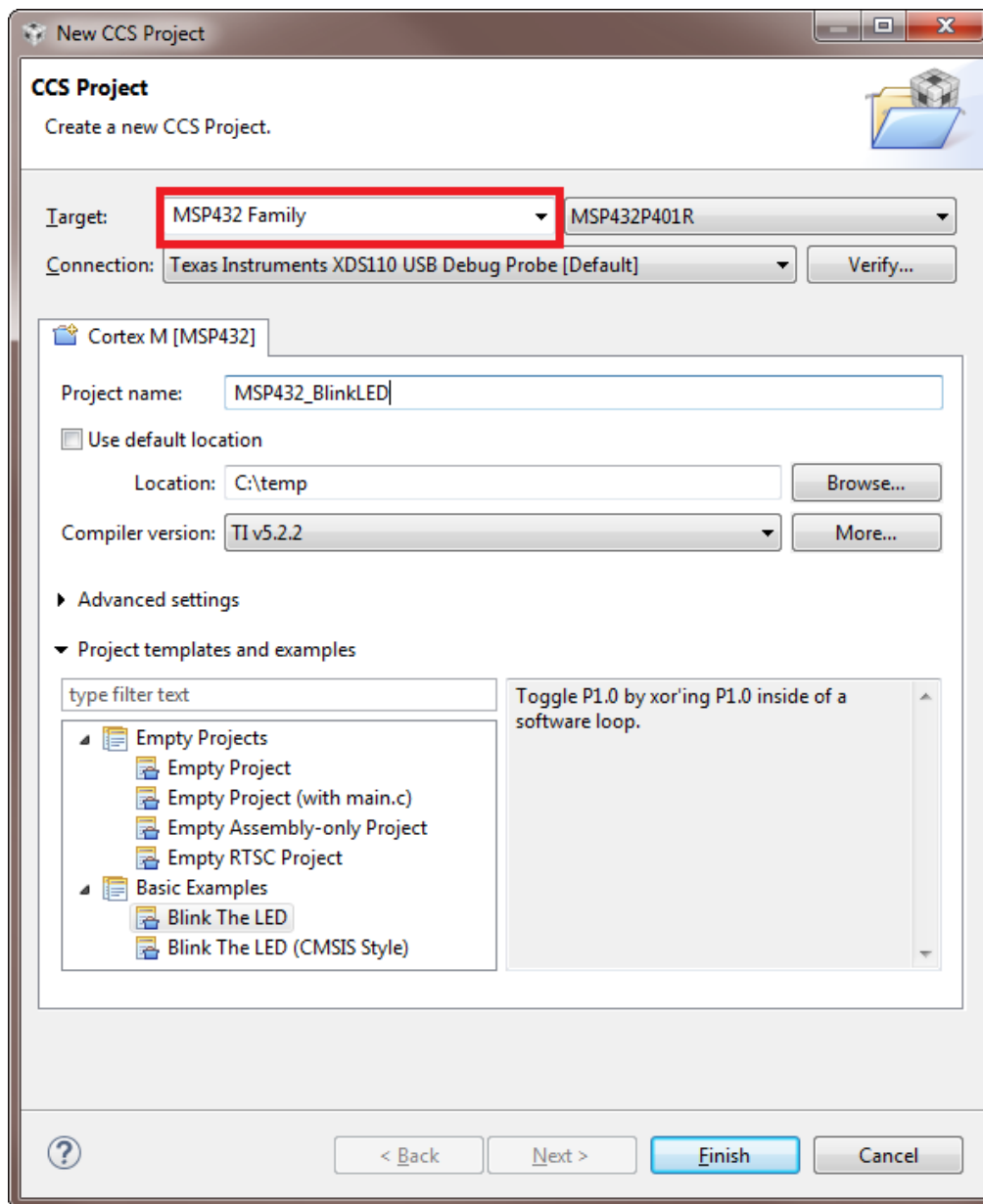


Figure 4. New Project Wizard

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

- A basic main.c file
- 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

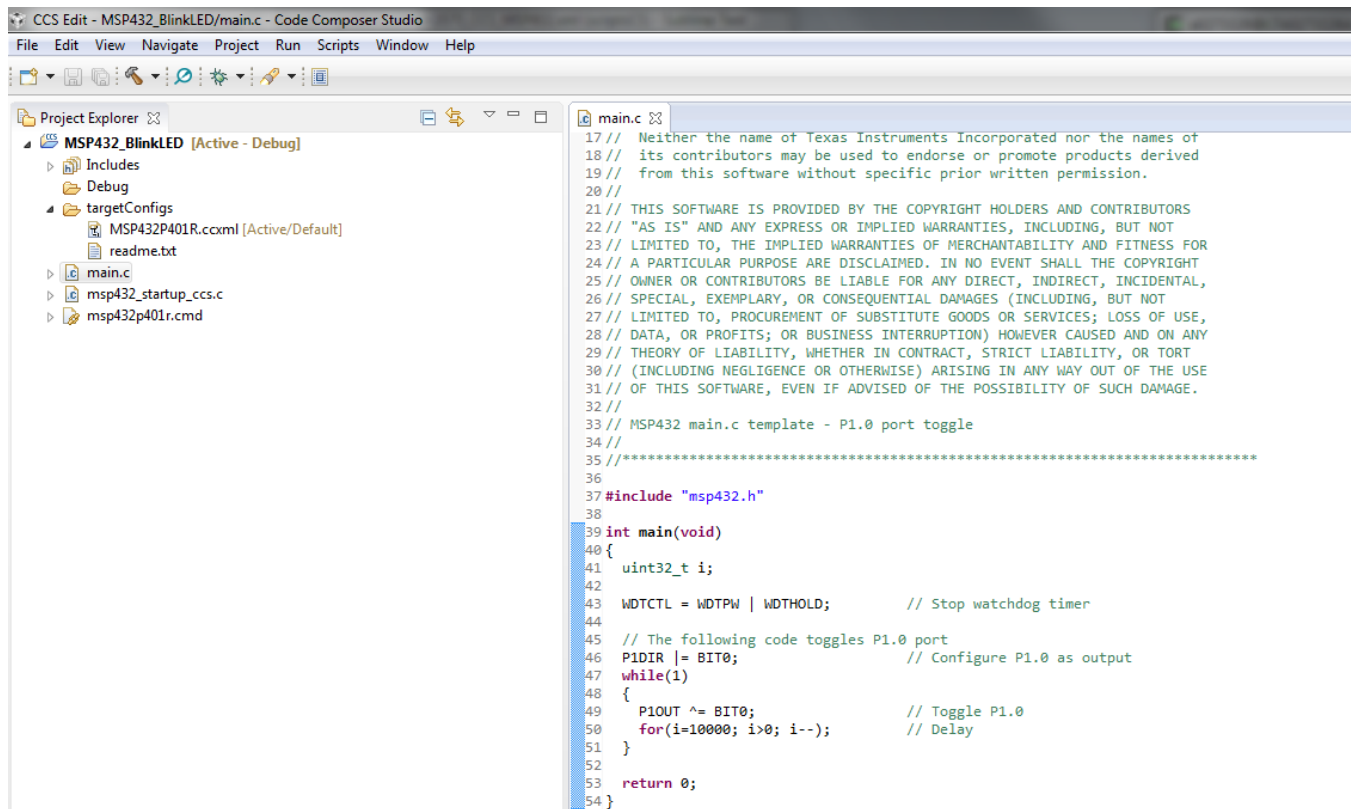


Figure 5. New Project Files

Now you can go on and start developing code.

4 Additional MSP432 Examples and Documentation

The Texas Instruments MSP team releases a software package called MSPWare that contains many software examples, projects, documentation as well as application notes and training for all MSP devices. This includes example projects for Code Composer Studio that work with MSP432. For more information, visit www.ti.com/tool/mspware.

5 Debugging Your Application

The following debug probes have been tested successfully with CCS.

- Texas Instruments XDS100v2, XDS100v3, XDS200, XDS110
- Segger J-Link (for more information, visit the TI J-Link Support Emulator Wiki Page at http://processors.wiki.ti.com/index.php/J-Link_Emulator_Support)

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 6).

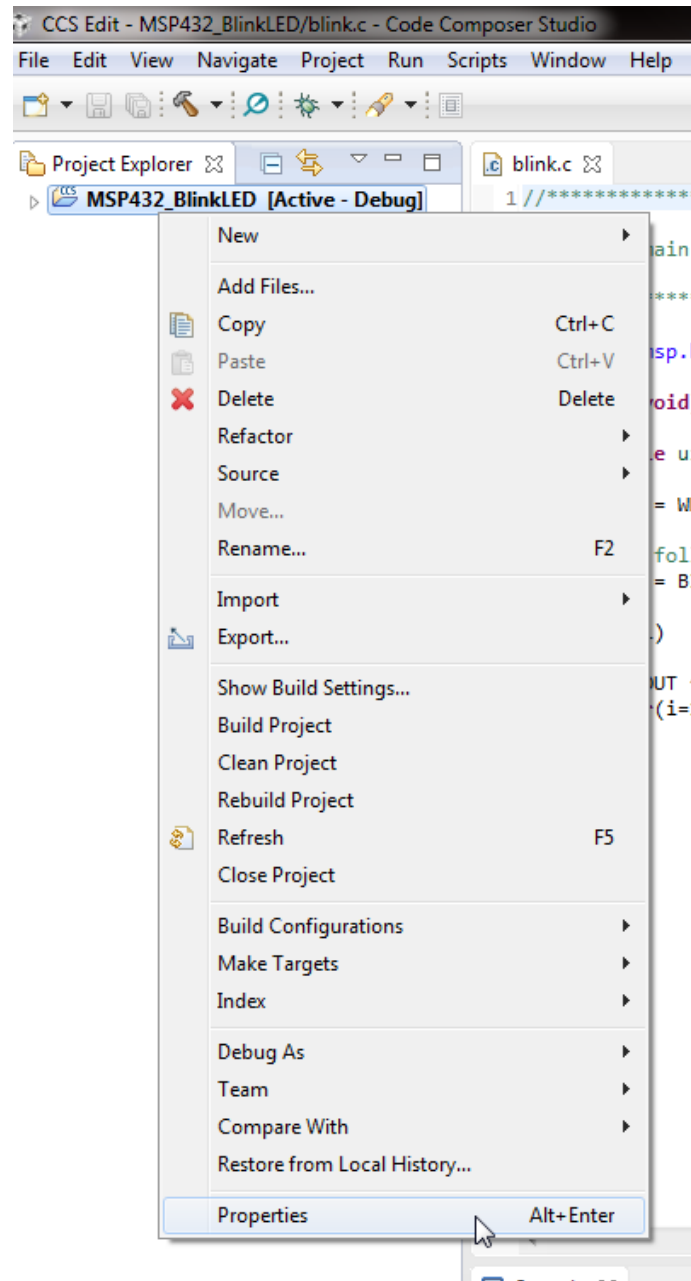


Figure 6. Project Properties

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**.

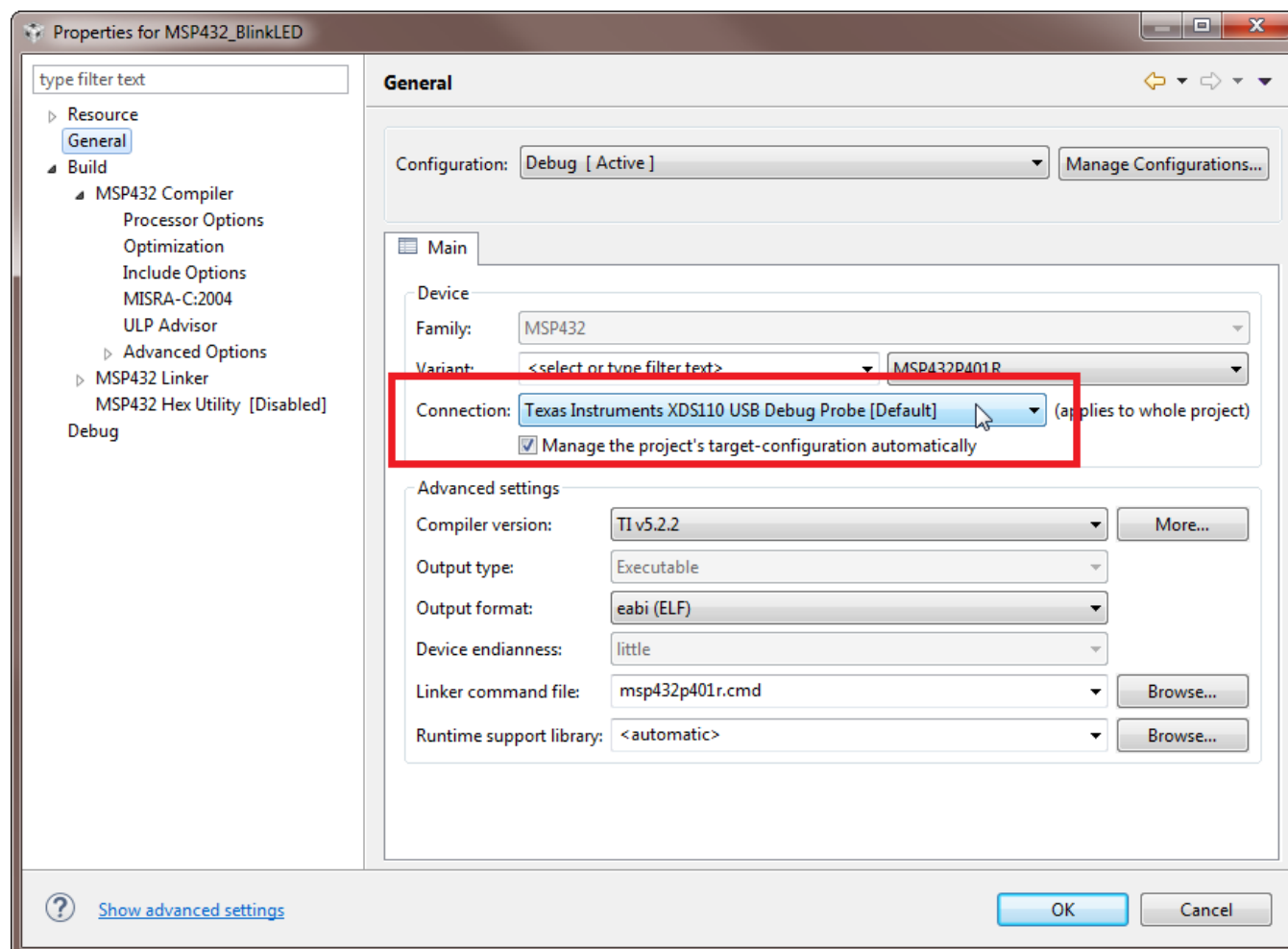



Figure 7. Choose Debugger Connection

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 8).

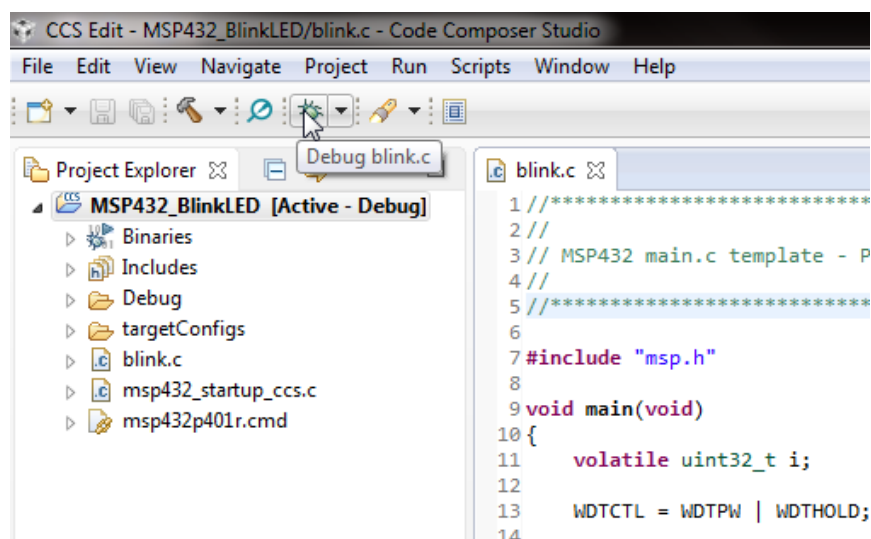


Figure 8. Launch Debug Session

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.

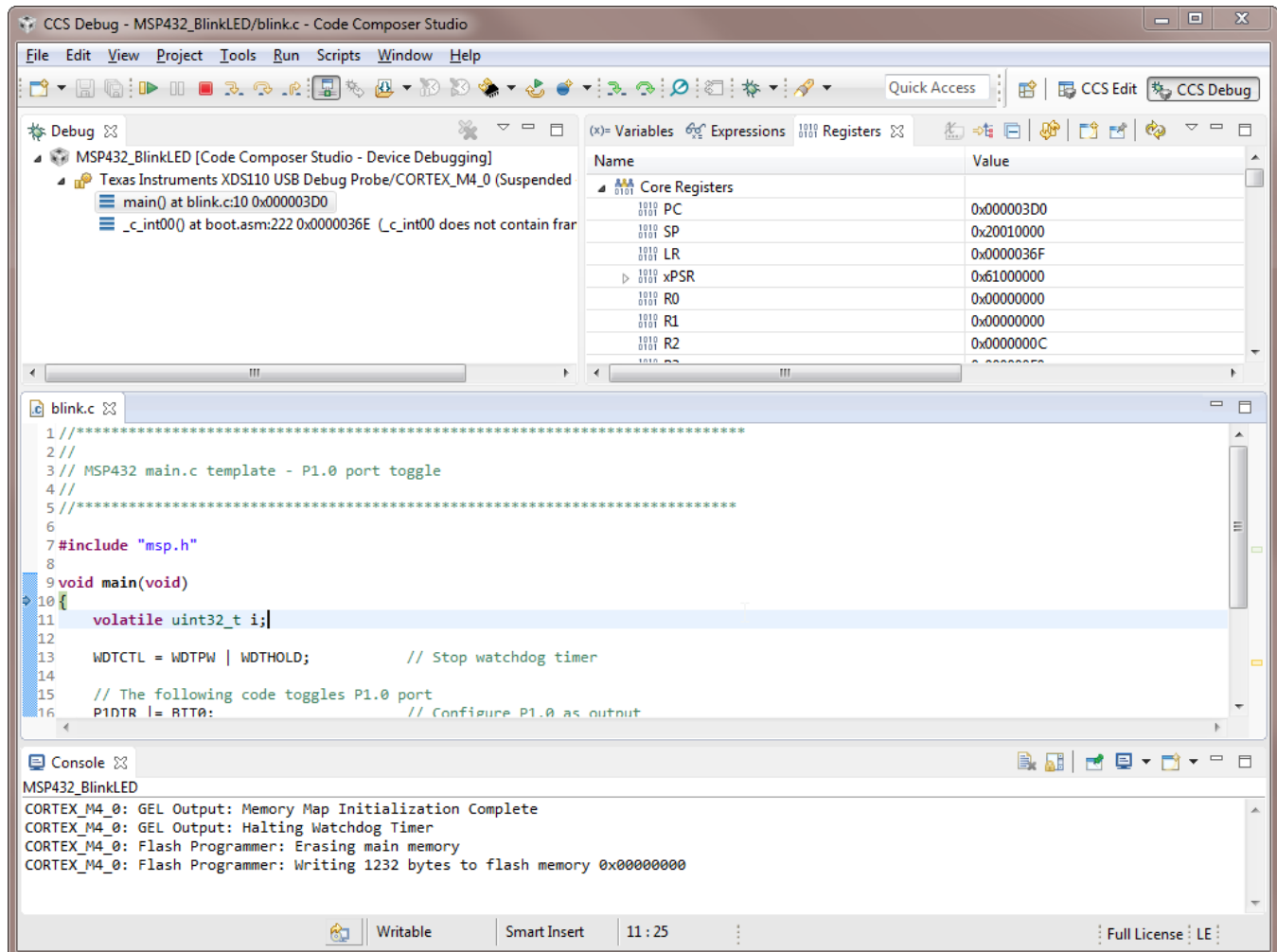


Figure 9. Debug Session

6 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 it to internal device information. The EnergyTrace mode is available for all MSP432 devices with selected debuggers, including CCS.

6.1 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 10 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.

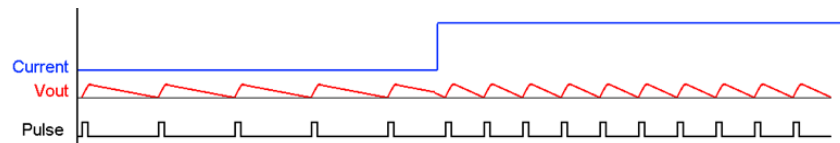


Figure 10. Pulse Density and Current Flow

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.

6.2 Code Composer Studio™ Integration

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.

6.3 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 11).

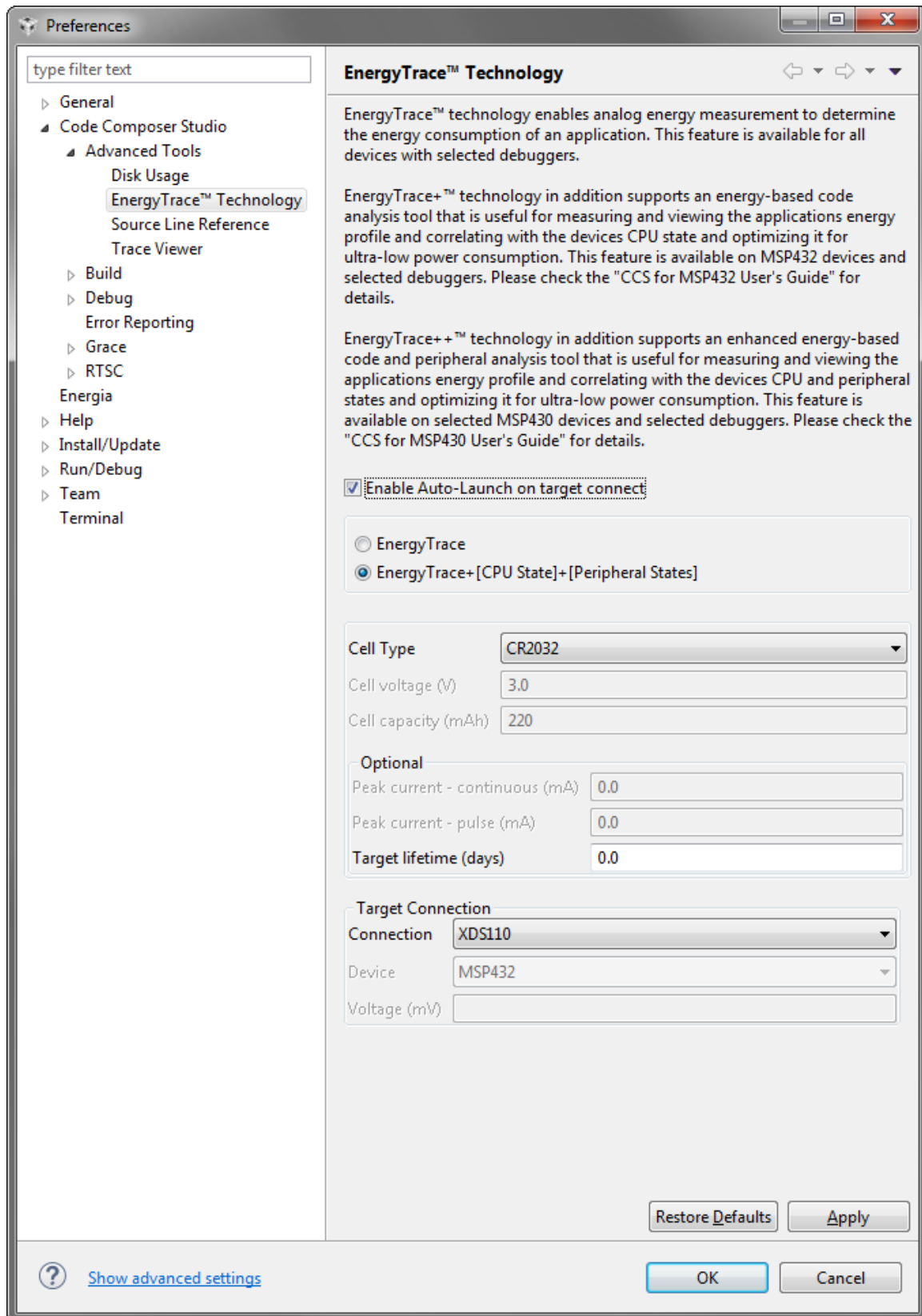



Figure 11. EnergyTrace™ Technology Preferences

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, refer to the device-specific data sheet, the *MSP432 Hardware Tools User's Guide*, 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?

6.4 Controlling EnergyTrace Technology

EnergyTrace Technology can be controlled using the control bar icons in the **Profile** window (see [Figure 12](#)). [Table 1](#) describes the function of each of these buttons.

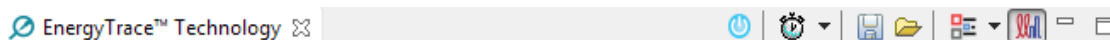








Figure 12. EnergyTrace™ Technology Control Bar

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 will start with "EnergyTrace_D" followed by a timestamp. When saving an EnergyTrace profile, the default filename will start 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

6.5 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

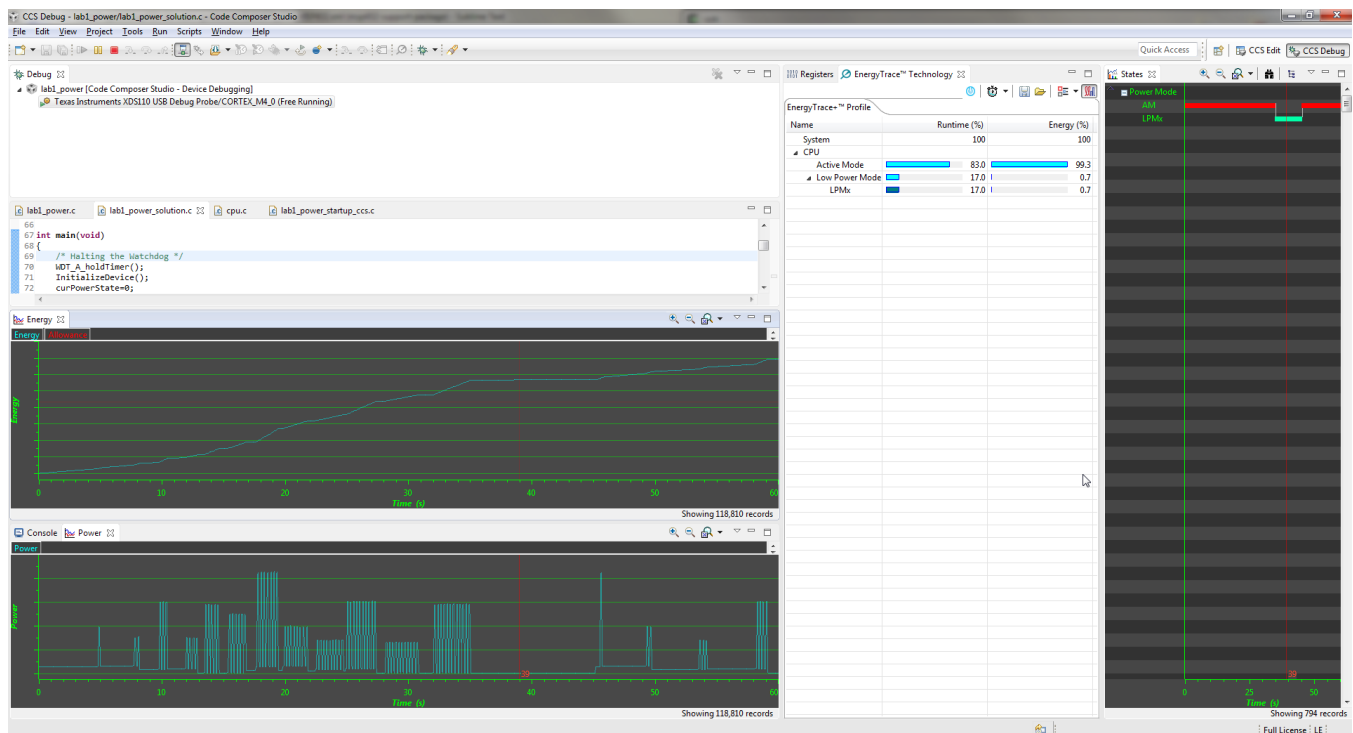


Figure 13. Debug Session With EnergyTrace+ Graphs

The **Profile** window (see Figure 14) 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.

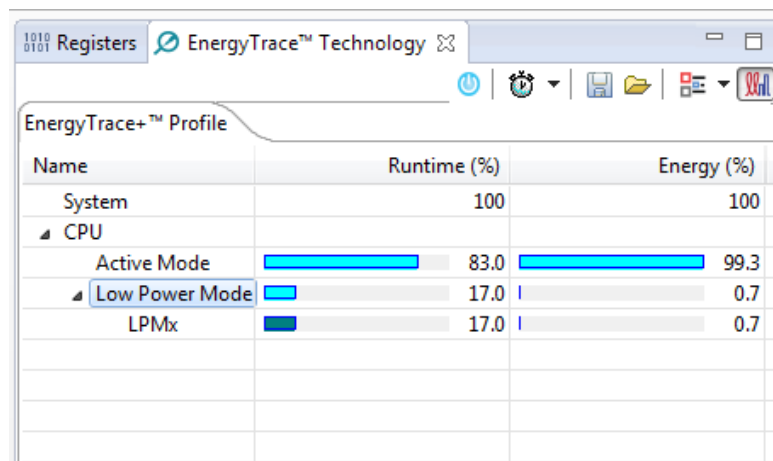


Figure 14. Profile Window

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 15](#)) 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 15](#) 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.

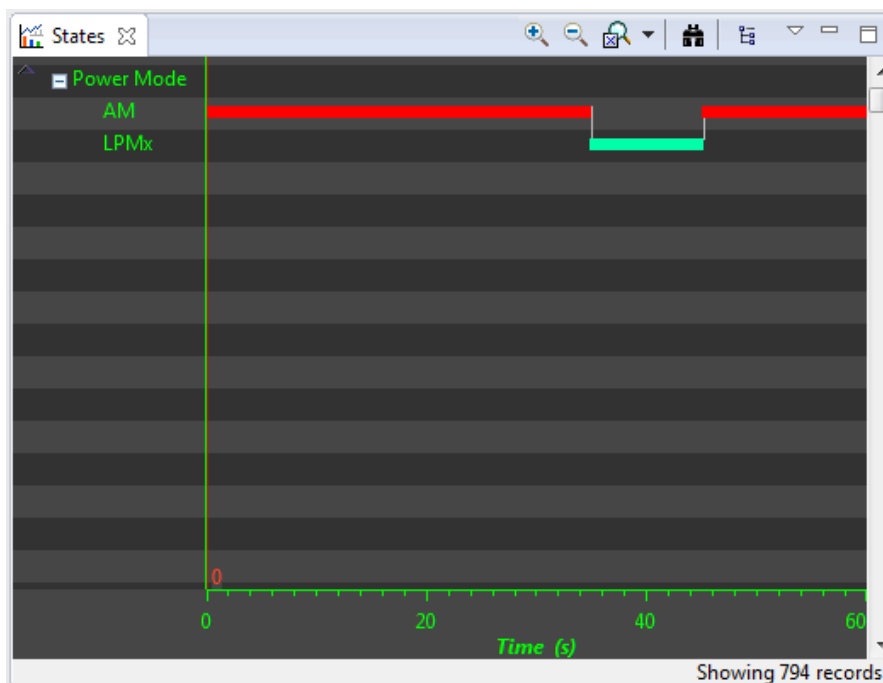


Figure 15. States Window

The **Power** window (see [Figure 16](#)) 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.

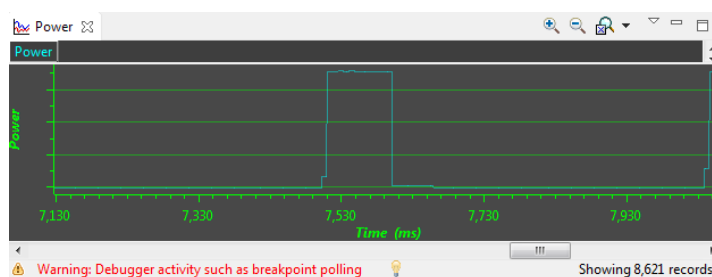


Figure 16. Power Window

The **Energy** window (see [Figure 17](#)) 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.

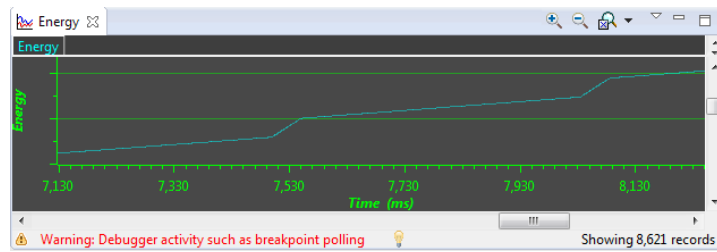


Figure 17. Energy Window

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.

6.6 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 18](#)):

- Profile
- Power
- Energy

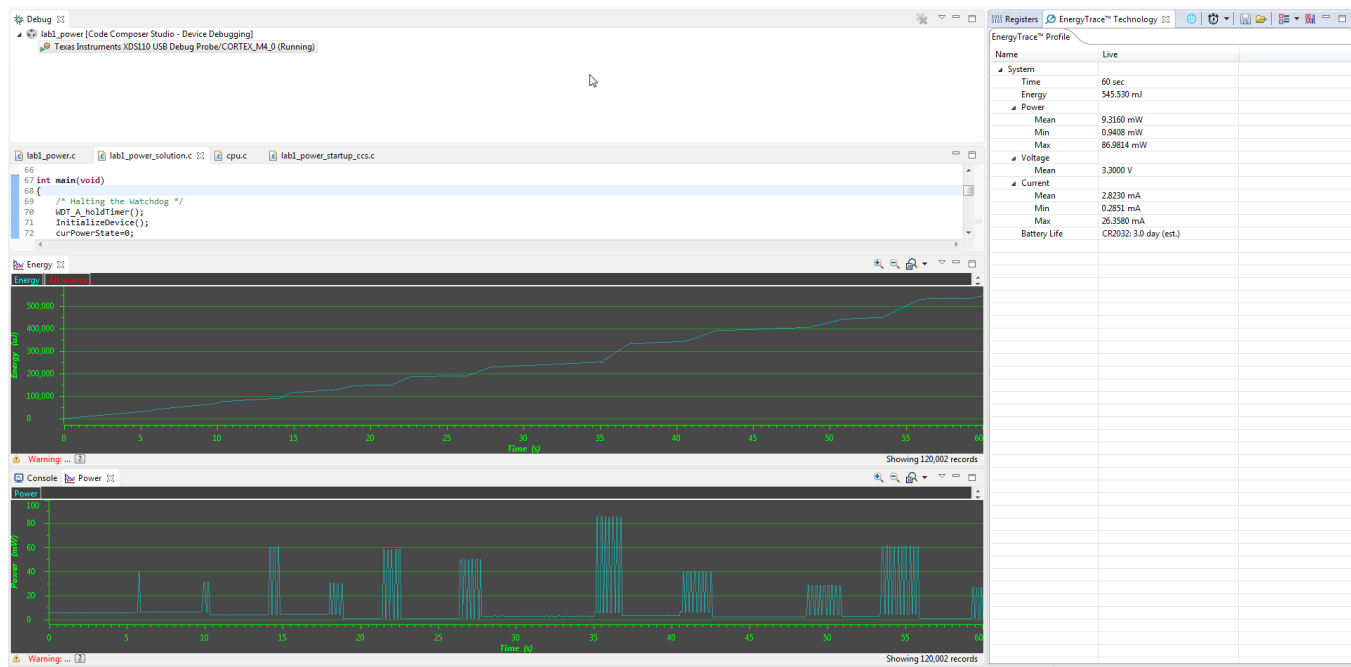


Figure 18. Debug Session With EnergyTrace Graphs

In the **EnergyTrace** mode, the **Profile** window shows statistical data about the application that has been profiled (see [Figure 19](#)). 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. It should also be noted that 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.

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 19. EnergyTrace Profile Window

The **Power** window (see [Figure 20](#)) 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.

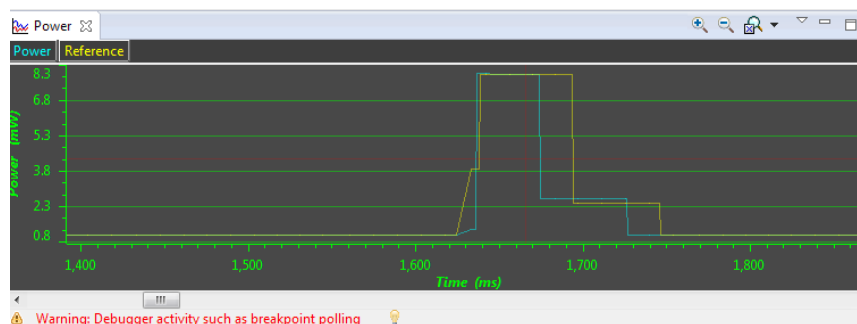


Figure 20. Zoom Into Power Window

The **Energy** window (see [Figure 21](#)) 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.

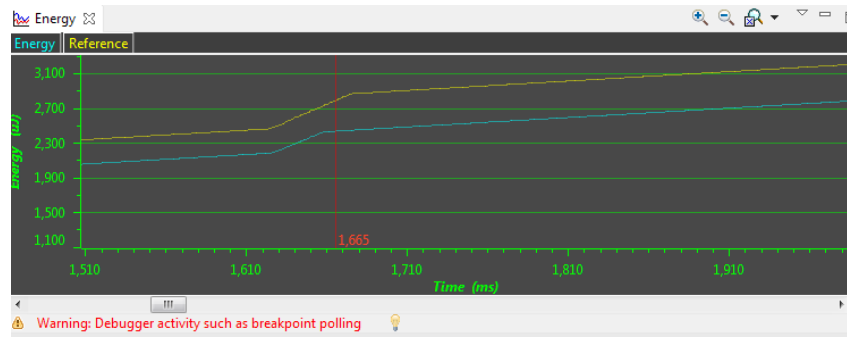


Figure 21. Current Profile (Blue) With Recorded Profile (Yellow)

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, it is recommended to use the **Free Run** mode. In Free Run mode, the debug logic of the target microcontroller is not accessed. See Figure 22 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.

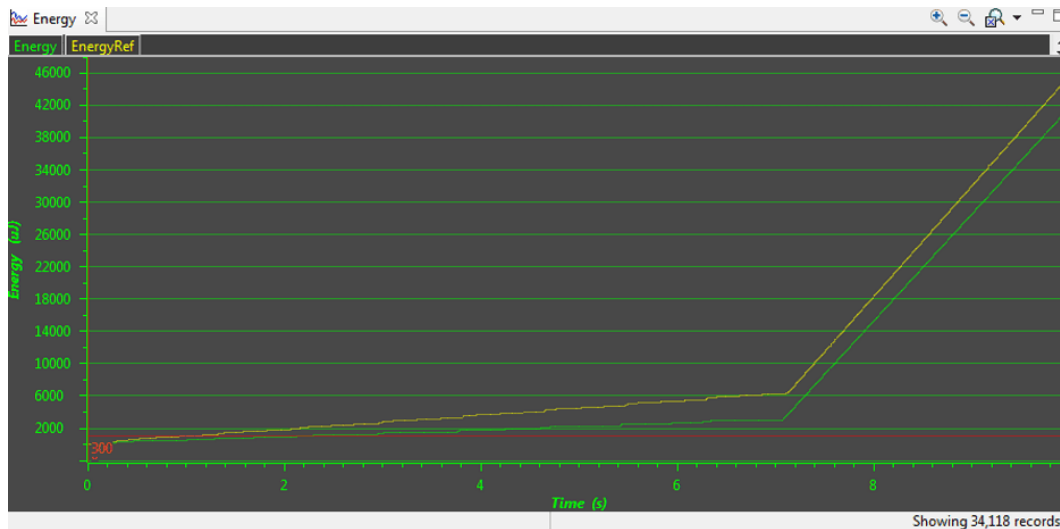


Figure 22. Energy Profile of the Same Program in Resume (Yellow Line) and Free Run (Green Line)

6.7 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 23](#)). 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.

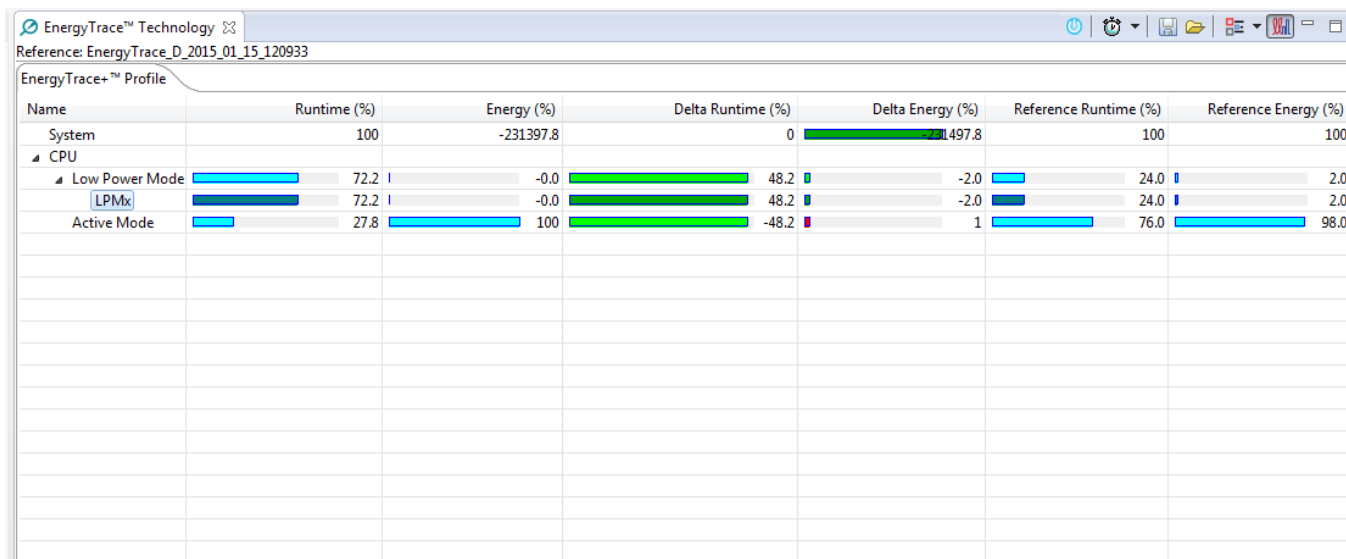


Figure 23. Comparing Profiles in EnergyTrace+ Mode

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 24](#)).

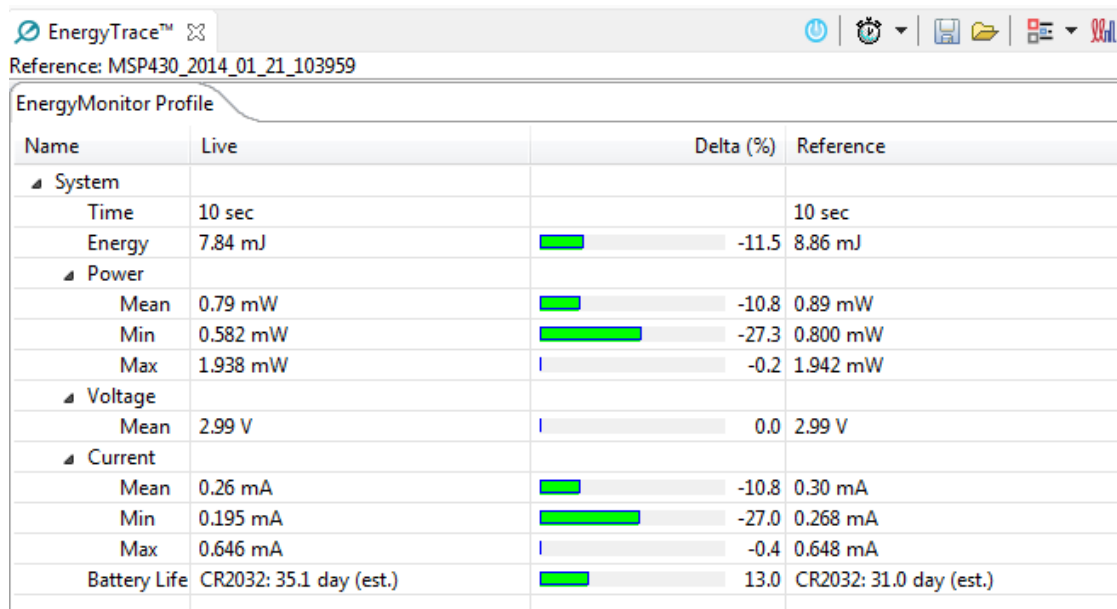


Figure 24. Comparing Profiles in EnergyTrace Mode

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

6.8 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 μ A, 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).

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 μ A.

A: Reading digital data from the target microcontroller consumes energy in the JTAG domain of the microcontroller. Hence, an average current of approximately 150 μ A 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.

7 Device Security

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 will be explained in detail next.

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

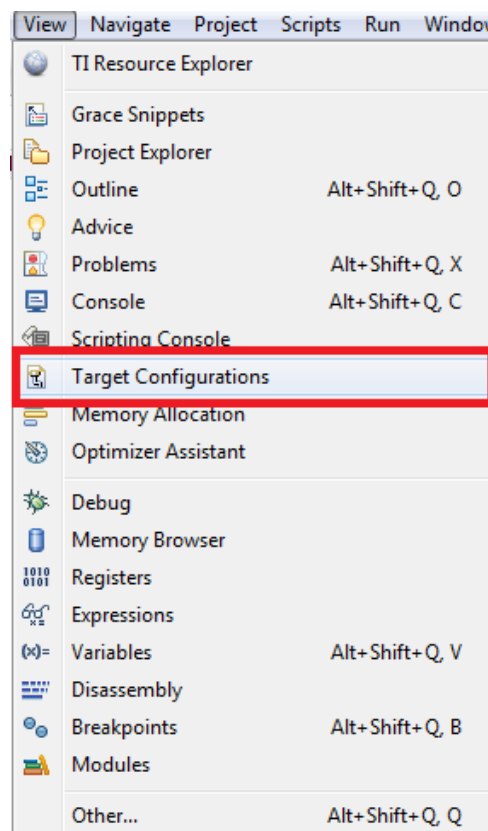


Figure 25. Show Target Configuration View

CCS will open a view showing the target configurations it can identify in the current workspace. Pick the one that works for the device and debugger. In our example we are using the configuration file for an XDS100v2 debugger.

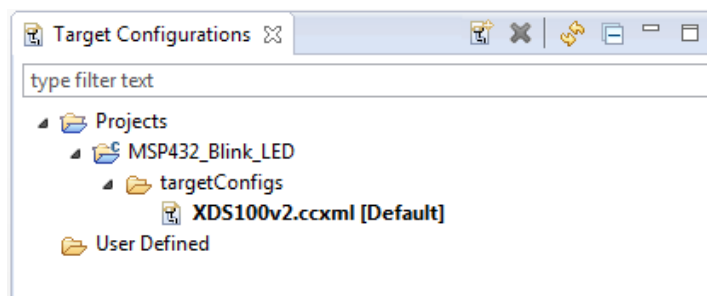


Figure 26. List of Target Configurations

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

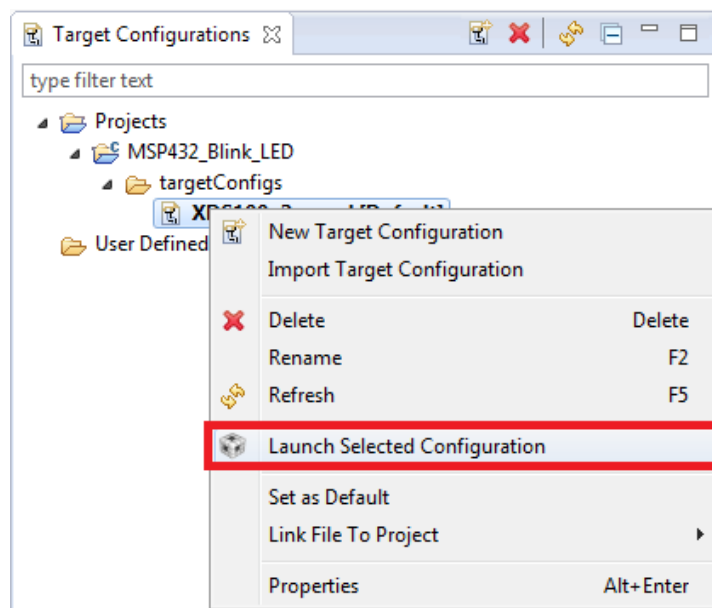


Figure 27. Launch Selected Target Configuration

The debugger will now connect 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 that is spanned shows the CPU core, but marks it as disconnected.

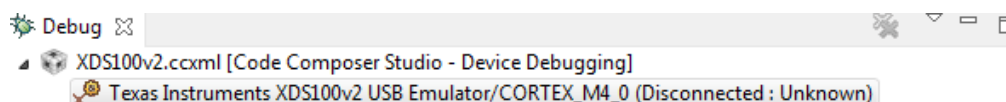


Figure 28. Debug View After Launching Target Configuration

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.

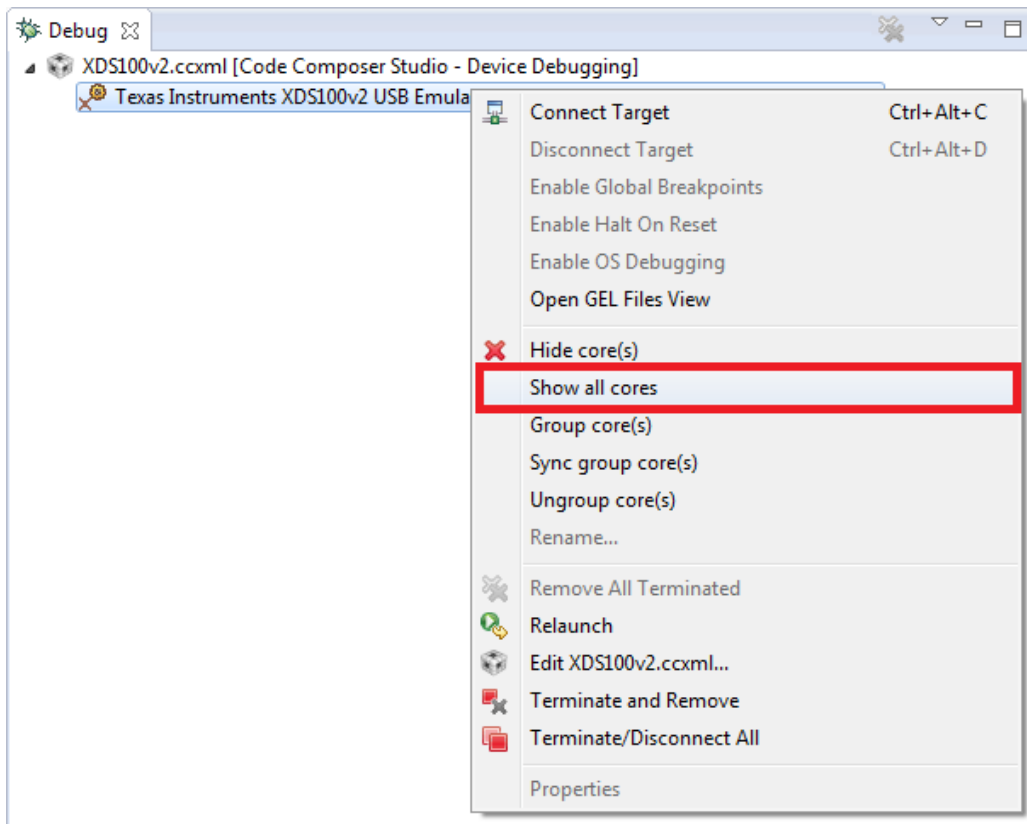


Figure 29. Show All Cores

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

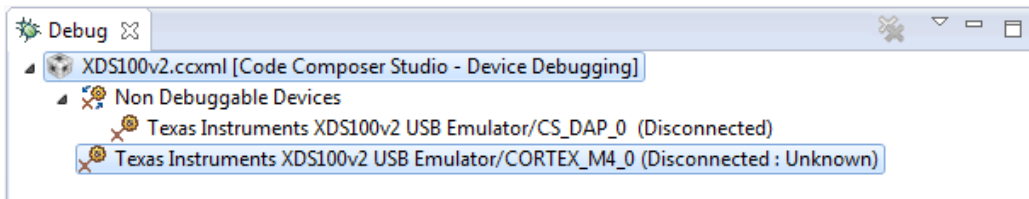


Figure 30. List of All Cores in the MSP432

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

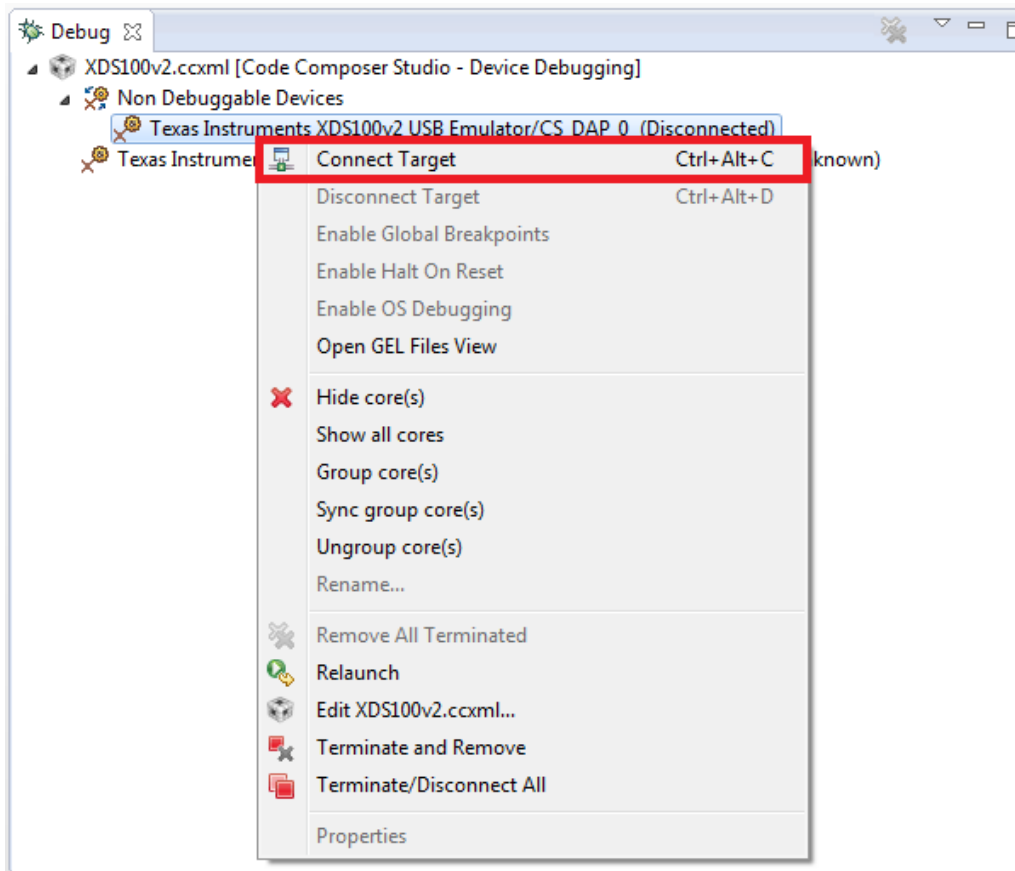


Figure 31. Manually Connecting to the DAP

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

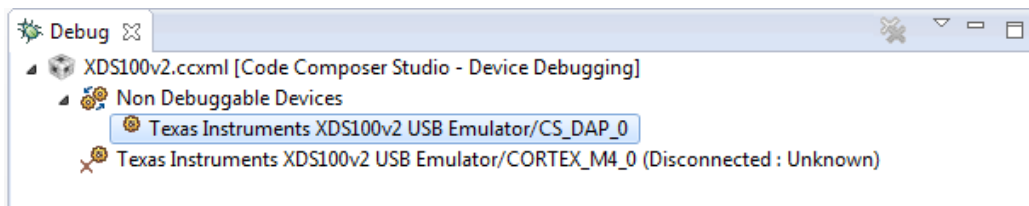


Figure 32. DAP is Connected

Now you need to execute a CCS script that performs triggers a reboot reset through the DAP. Go to **Scripts** → **default** → **MSP432_Factory_Reset** and click on it.

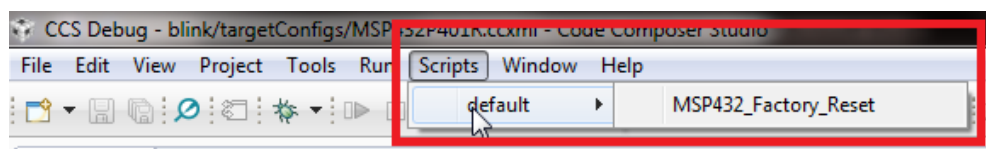
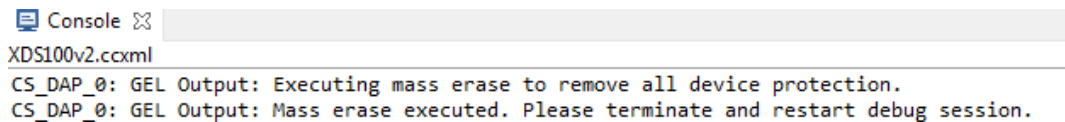


Figure 33. Executing the Factory Reset Script

As the script is executed, the Console window will show 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.



```

Console
XDS100v2.ccxml
CS_DAP_0: GEL Output: Executing mass erase to remove all device protection.
CS_DAP_0: GEL Output: Mass erase executed. Please terminate and restart debug session.

```

Figure 34. Mass Erase Script Console Output

8 Low-Power Debug

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

If users want to verify the current consumption and timing of their application while still being under basic debug control, MSP432 offers the **Low Power Run** feature. When enabled, the MCU will transition into exactly the low power mode the user has specified, with internal clocks being disabled and the power management module shutting down internal power domains. This has some implications:

- Previously set breakpoints will be hit, but the IDE will not automatically indicate that the device has halted. The user needs to manually hit the halt icon to enable the IDE to reconnect and show where the program has halted.
- Auto Run after loading a program will not work: The breakpoint that is set automatically by the IDE [for example, at main()] will be hit, but the IDE will not switch to halt automatically. When the user halts manually after program load, the program counter will be 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 CCS, right click on the active project in the Project Explorer and click on Properties.

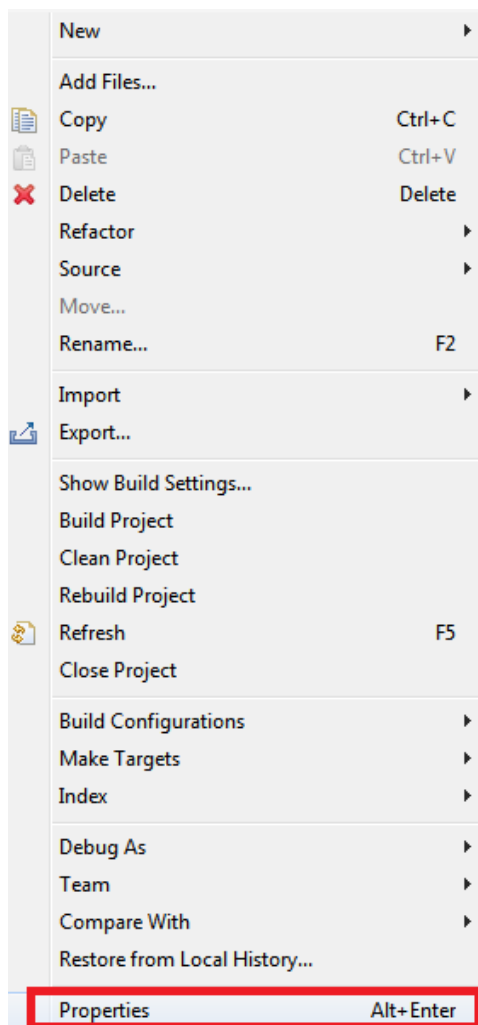


Figure 35. Properties Menu

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.

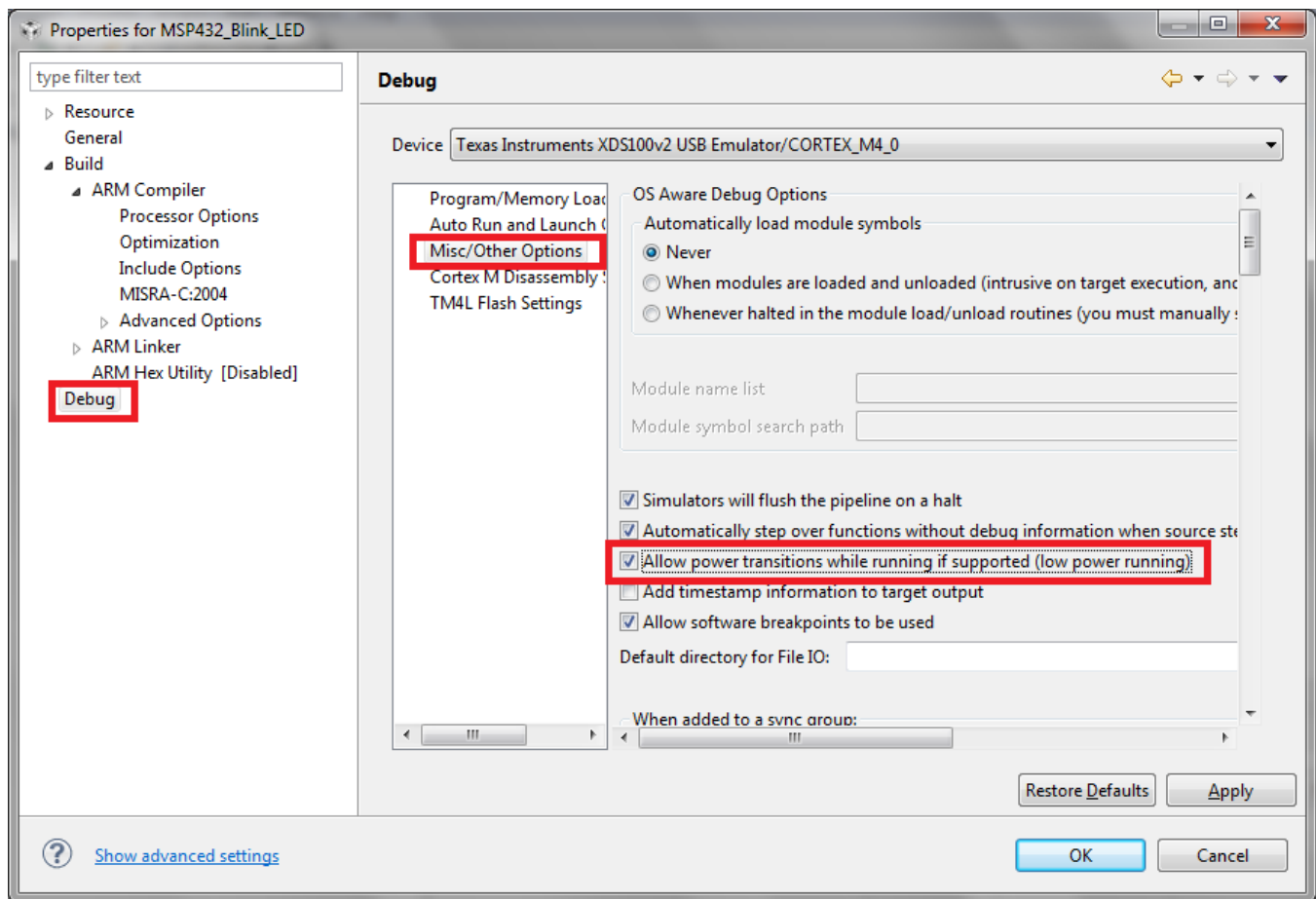


Figure 36. Enabling Low Power Run

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, or DAP.

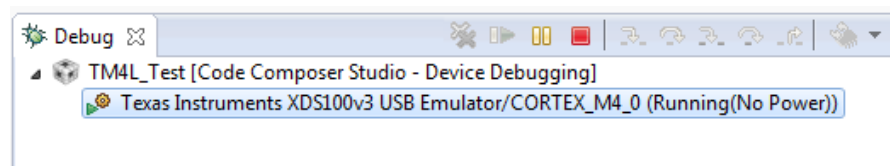


Figure 37. CPU Core Status Display Indicating Deep Sleep Mode

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.

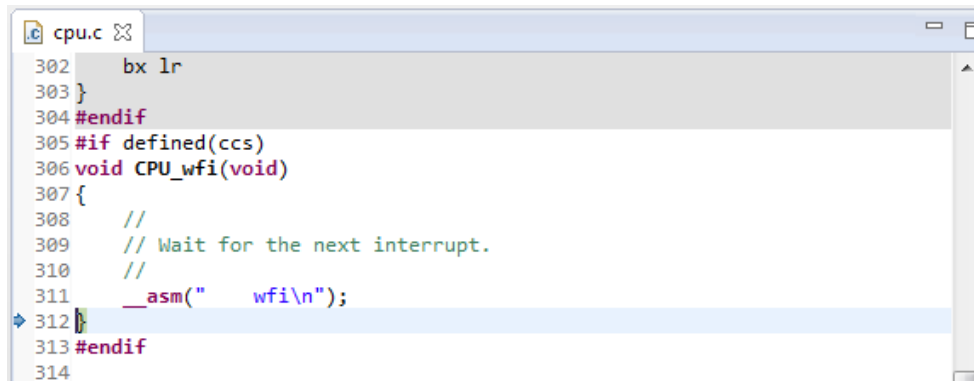


Figure 38. Program Counter Located at WFI Instruction

9 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 Serial Wire Debug (SWD) without SWO. 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 39](#) shows how to configure the debugger to use SWD instead of JTAG by modifying the MSP432P401R.ccxml file.

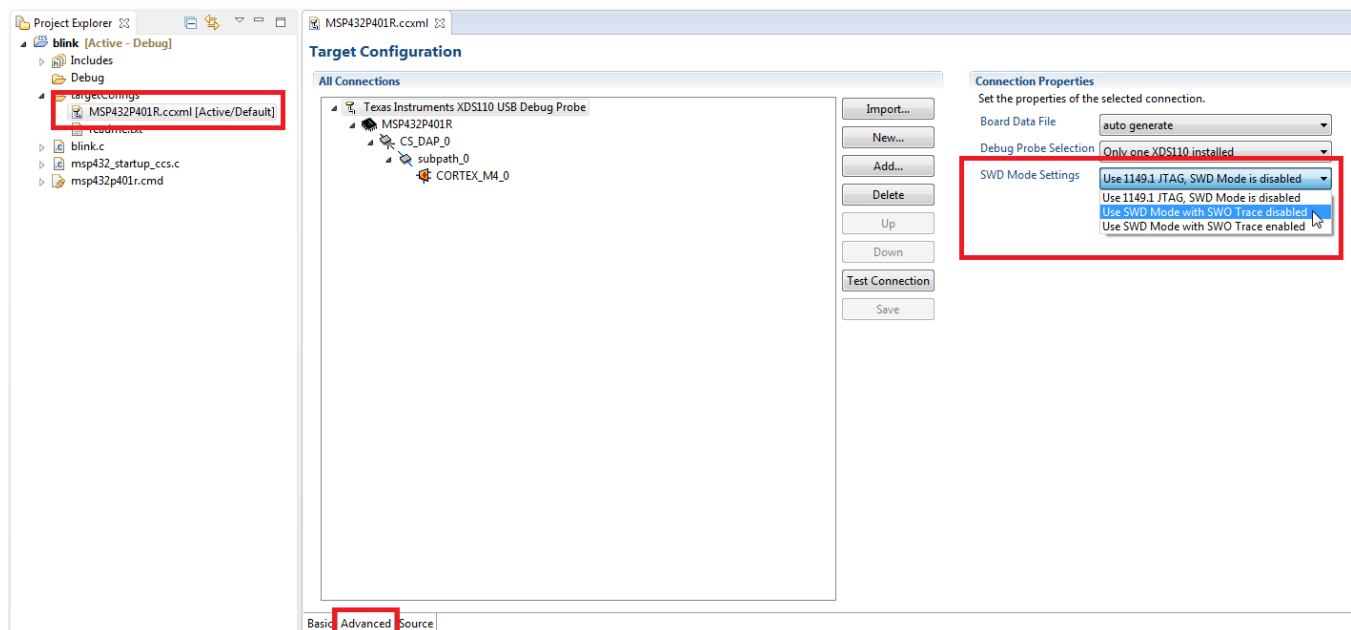


Figure 39. Change Debugger Settings to SWD

- If even this cannot connect, reset the device to factory settings. Review [Section 7](#) for information on how to perform a factory reset on the device.

Q: Why doesn't the back-channel 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 CCS 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 CCS-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).

10 Additional Code Composer Studio Information

For more information about Code Composer Studio, refer to the following links:

- Code Composer Studio Information (<http://www.ti.com/tool/ccstudio>)
- Code Composer Studio v6 Training (http://processors.wiki.ti.com/index.php/Category:CCSv6_Training)
- Code Composer Studio v6 Wiki (http://processors.wiki.ti.com/index.php/Category:Code_Composer_Studio_v6)
- Code Composer Studio Quick Start Guide (<http://downloads.ti.com/ccs/esd/documentation/CCSv6-QSG.pdf>)

11 References

1. MSPWare (www.ti.com/tool/mspware)
2. J-Link Emulator Support (http://processors.wiki.ti.com/index.php/J-Link_Emulator_Support)

Revision History

Changes from May 13, 2015 to May 20, 2015**Page**

-
- Added "Mac OS X" to introduction [1](#)
-

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

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information 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.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have **not** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products

Audio	www.ti.com/audio
Amplifiers	amplifier.ti.com
Data Converters	dataconverter.ti.com
DLP® Products	www.dlp.com
DSP	dsp.ti.com
Clocks and Timers	www.ti.com/clocks
Interface	interface.ti.com
Logic	logic.ti.com
Power Mgmt	power.ti.com
Microcontrollers	microcontroller.ti.com
RFID	www.ti-rfid.com
OMAP Applications Processors	www.ti.com/omap
Wireless Connectivity	www.ti.com/wirelessconnectivity

Applications

Automotive and Transportation	www.ti.com/automotive
Communications and Telecom	www.ti.com/communications
Computers and Peripherals	www.ti.com/computers
Consumer Electronics	www.ti.com/consumer-apps
Energy and Lighting	www.ti.com/energy
Industrial	www.ti.com/industrial
Medical	www.ti.com/medical
Security	www.ti.com/security
Space, Avionics and Defense	www.ti.com/space-avionics-defense
Video and Imaging	www.ti.com/video

TI E2E Community

e2e.ti.com