Texas Instruments MSP432: Cortex™-M4 Tutorial
Using the MSP432P401R LaunchPad Board
and ARM Keil MDK 5 Toolkit

Winter 2018 V 3.8 bob.boys@arm.com

The latest version of this document is here: www.keil.com/appnotes/docs/apnt_276.asp

MSP432E401Y board ETM tutorial is here: www.keil.com/appnotes/docs/apnt_309.asp

Introduction:

The purpose of this lab is to introduce you to the TI MSP432 processor using the ARM® Keil® MDK toolkit featuring the IDE µVision®. At the end of this tutorial, you will be able to confidently work with this processor and Keil MDK.

Keil MDK supports TI Cortex-M processors. Check the Keil Device Database® on www.keil.com/dd2 for the complete list. Software Packs for MSP432, LM3S, LM4F and Tiva C are available here: www.keil.com/dd2/pack/.

See www.keil.com/TI for more details. For MDK 5.20, see www.keil.com/mdk5/version520. Keil also has TI 8051 support.

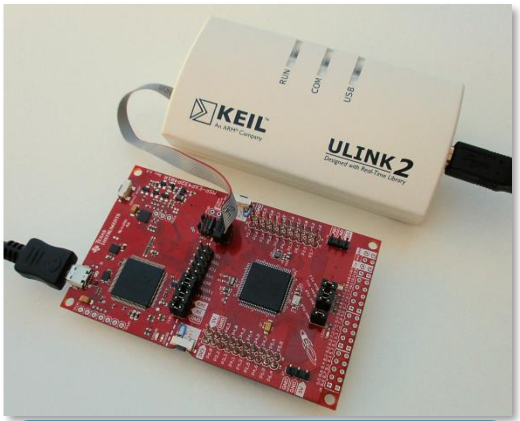
Cortex-A: TI Cortex-A processors are supported by ARM DS-5™. DS-5 is Linux and Android aware. www.arm.com/ds5.

Keil MDK-Lite™ is a free evaluation version that limits code size to 32 Kbytes. Nearly all Keil examples will compile within this 32K limit. The addition of a valid license number will turn MDK into a specific commercial version.

Why Use Keil MDK ?

MDK provides these features particularly suited for TI Cortex-M users:

1. µVision IDE with Integrated Debugger, Flash programmer and the ARM Compiler/assembler/linker toolchain.
   MDK is a complete turn-key "out-of-the-box" tool solution.
2. You can use the ARM Compiler 5, ARM Compiler 6 (LLVM) or GCC in µVision. ELF/DWARF is supported.
3. Dynamic Syntax Checking and Code Completion.
4. **Compiler Safety Certification Kit:** www.keil.com/safety/
5. **TÜV** certified. SIL3 (IEC 61508) and ASILD (ISO 26262).
6. MISRA C/C++ support using PC-Lint. www.gimpel.com
7. A full feature Keil RTX RTOS with source code. RTX has a BSD license. www.keil.com/rtx FreeRTOS is supported.
8. ARM CoreSight™ debugging technology is supported.
9. Serial Wire Viewer (SWV) with any Keil ULINK™ and J-Link.
10. ETM trace on selected TI processors supported with ULINK*pro*.
11. Debug Adapters: Launchpad XDS110, Keil ULINK™2, ULINK*plus*, ULINK-ME, ULINK*pro* and J-Link.
12. ***NEW !*** **ULINK*plus*:** Power Measuring and fast SWV support. www.keil.com/ulinkplus
13. Keil Technical Support is included for one year and is easily renewable. This helps your project get completed faster.
14. Affordable perpetual and term licensing. Contact Keil sales for pricing options. See the last page in this document.
15. ***NEW !*** Event Recorder: Instrument your code. www.keil.com/pack/doc/compiler/EventRecorder/html/cv_use.html



The MSP432 LaunchPad connected to a Keil ULINK2. This lab will use the onboard TI XDS110 debug adapter except for those portions using Serial Wire Viewer (SWV).

This document includes details on these features plus more:

1. Real-time read and write operations to memory locations in the Watch, Memory and Peripheral windows. These are non-intrusive to your program. No CPU cycles are stolen. No instrumentation code is added to your source files.
2. Six Hardware Breakpoints (can be set/unset on-the-fly) and one Watchpoint (also known as Access Breaks).
3. Two RTX kernel awareness windows for RTX that update while your program is running.
4. A DSP example program using ARM CMSIS-DSP libraries including ITM *printf*.
5. Two LoPower examples. With and without RTX. The MSP432 goes into LMP3 mode consuming little power.
6. How to create your own µVision projects using Software Packs with or without RTX.

General Information:

Install Keil MDK and Software Packs:

Connecting a Debug Adapter:

Blinky Example and Debugging Features:

RTX_Blinky Example with Keil RTX:

DSP Sine Example:

Low Power Example:

Creating your own MDK 5 Blinky projects from scratch:

Other Useful Information:

1) CoreSight™ Definitions: *It is useful to have a basic understanding of these terms:*

Various Cortex-M processors will have specific features. Consult your TI datasheet for feature implementation.

ETM requires a 20 pin CoreSight connector that is not usually installed on evaluation boards. Most boards have the legacy 20 pin JTAG and 10 pin CoreSight connectors. See www.keil.com/coresight/coresight-connectors It is possible to make a simple adapter to connect the ETM signals from the processor to the ULINK*pro* 20 pin CoreSight connector.

1. **JTAG:** Provides access to the CoreSight debugging module located on the Cortex processor. It uses 4 to 5 pins.
2. **SWD:** Serial Wire Debug is a two pin alternative to JTAG and has about the same capabilities except Boundary Scan is not possible. SWD is referenced as SW in the µVision Cortex-M Target Driver Setup. The SWJ box must be selected in ULINK2/ME or ULINK*pro*. Serial Wire Viewer (SWV) must use SWD because the JTAG signal TDIO shares the same pin as SWO. The SWV data normally comes out the SWO pin or optionally out the 4 bit Trace Port.
3. JTAG and SWD are functionally equivalent. The signals and protocols are not directly compatible.
4. **DAP:** Debug Access Port. This is a component of the ARM CoreSight debugging module that is accessed via the JTAG or SWD port. One of the features of the DAP are the memory read and write access which provides on-the-fly memory accesses without the need for processor core intervention. µVision uses the DAP to update Memory, Watch, System Viewer (peripherals) and RTOS kernel awareness windows in real-time while the processor is running. You can also modify variable values on the fly. No CPU cycles are used and no source code stubs are used.
   You do not need to configure or activate DAP. µVision configures DAP when you select a function that uses it.
   Do not confuse this with CMSIS_DAP which is an ARM on-board debug adapter standard.
5. **SWV:** Serial Wire Viewer: a Data trace providing display of reads, writes, exceptions, PC Samples and printf.
6. **ITM:** Instrumentation Trace Macrocell: As used by µVision, ITM has thirty-two 32 bit memory addresses (Port 0 through 31) that when written to, will be output on either the SWO or Trace Port. This is useful for printf type operations. µVision uses Port 0 for printf and Port 31 for the RTOS Event Viewer. The data can be saved to a file.
7. **SWO:** Serial Wire Output: SWV frames usually come out this one pin output. It shares the JTAG signal TDIO.
8. **Trace Port:** A 4 bit port that ULINK*pro* uses to collect ETM frames and optionally SWV (rather than SWO pin).
9. **ETM:** Embedded Trace Macrocell: Displays all the executed instructions. The ULINK*pro* provides ETM using a suitable Cortex-M3/M4/M7 processor. ETM requires a special 20 pin CoreSight connector mounted on the target hardware. ETM also provides Code Coverage, Execution Timing and Performance Analysis. Many Cortex-M processors have ETM. Check your specific datasheet for details of features implemented by the manufacturer.
10. **MTB:** Micro Trace Buffer. A portion of the device internal RAM is used for an instruction trace buffer. Only on certain Cortex-M0+ processors. No special debug adapter such as a ULINK*pro* is needed. Just JTAG or SWD.
11. **Hardware Breakpoints:** Most Cortex-M0+ have 2 breakpoints. Most Cortex-M3, M4 and M7 have 6. These can be set/unset on-the-fly without stopping the processor. They are no skid: they do not execute the instruction they are set on when a match occurs. The exact number of breakpoints is selected by the manufacturer at design time.
12. **WatchPoints:** Both Cortex-M0+, Cortex-M4 and Cortex-M7 have 2 Watchpoints. These are conditional breakpoints. They stop the program when a specified value is read and/or written to a specified address or variable. They are also referenced as Access Breakpoints in Keil documents.

The MSP432 LaunchPad board connected to a Keil ULINK*pro*. A 20 to 10 pin adapter cable is used. This cable is normally provided with the ULINK*pro*.

2) TI MSP432 LaunchPad Board & Keil Evaluation Software:

Keil provides board support for many TI Cortex-M3 and Cortex-M4 processors. This is contained in the Software Packs.

On the second last page of this document is an extensive list of resources that will help you successfully finish your projects. This list includes application notes, books, labs and tutorials.

We recommend you obtain the latest Getting Started Guide for MDK5: It is available here: www.keil.com/gsg/.

Community Forums: www.keil.com/forum and http://community.arm.com/groups/tools/content

3) Keil MDK 5 Information: This document used *MDK 5.20. You can use a later version.*

MDK 5 uses CMSIS compliant software. CMSIS is an ARM standard that downloads software including middleware, header and configuration files, RTX and documents from a webserver. These Packs are downloaded with "Pack Installer", the versions selected with "Select Software Packs" and components are selected with "Manage Run Time Environment" (MRTE).

Most TI Cortex-M processors have a Software Pack. See www.keil.com/dd2/pack for the current list.

Example Files: The examples are available where this document is stored: www.keil.com/appnotes/docs/apnt_276.asp.

RTX: RTX is a Keil RTOS that is provided with all source files and a BSD license. This makes it free. RTX, as is distributed with MDK 5, is CMSIS-RTOS compliant. You can use other RTOSs with MSP432 and MDK.

This tutorial used Software Pack 2.2.0. It is better to not have any other earlier MSP432 Packs installed including 1.0.3.

4) USB Debug Adapters: For Serial Wire Viewer (SWV) use a Keil ULINK2, ULINK*pro* or J-Link.

Keil manufactures and supports several adapters. These are listed below with a brief description.

1. **Keil ULINK2 and ULINK-ME:** ULINK2 is pictured on page 1. ULINK-ME is offered only as part of certain evaluation board packages. These are electrically the same and both support Serial Wire Viewer (SWV). SWV and DAP update non-intrusively while the program is running. Run-time memory reads and writes for the Watch, Memory and System Viewer windows are non-intrusive. Hardware breakpoints can be set/unset on-the-fly.
2. ***NEW !*** **ULINK*plus*:** High SWV performance plus Power Measurement. It is pictured on page 1. See www.keil.com/ulink/ulinkplus/ for details.
3. **Keil ULINK*pro*:** ULINK*pro* is pictured on page 3. It supports all SWV features and adds ETM Instruction Trace support. ETM records all executed instructions and provides Code Coverage, Execution Profiling and Performance Analysis features. The MSP432 does not have ETM. The MPS432E401 does have ETM. See www.keil.com/appnotes/docs/apnt_309.asp ULINK*pro* also provides fast Flash programming times.
4. **TI XDS110 Emulator:** MSP432 LaunchPad contains an on-board XDS110 debug adapter. Currently, XDS110 does not support Serial Wire Viewer. Use any ULINK or a J-Link to utilize this useful feature.
5. **Segger J-Link:** J-Link Version 6 (black case) or later supports Serial Wire Viewer. SWV data reads and writes are not currently supported with a J-Link. Program execution must be stopped to view the trace window. J-Link USB drivers must be manually installed. See C:\Keil_v5\ARM\Segger\USBDriver\ J-Link SWV configuration windows are slightly different from a Keil ULINK. Segger offers a 20 pin to 10 pin connector adapter needed to connect to the LaunchPad to the 10 pin EXT CoreSight Debug connector J102. Contact www.segger.com to obtain this adapter. This adapter is shown here:

CMSIS-DAP: This is an ARM standard for debug adapters. SWV is a new option. CMSIS-DAP can be used to put a debug adapter on a custom board. See CMSIS-DAP is widely supported by development tools. The XDS110 does not currently support CMSIS-DAP. ***New*** See https://github.com/ARM-software/CMSIS_5

Serial Wire Viewer (SWV): This is an ARM CoreSight feature. SWV provides data trace, the ability to graph four variables and see exceptions and interrupts plus other events occurring in real-time. No code stubs are needed in your code. Keil ULINK2, ULINK-ME, ULINK*plus*, ULINK*pro* and J-Link all provide SWV. TI XDS110, at this time, does not.

ETM Instruction Trace: ETM collects all the instructions executed. This is useful for program flow debugging. Code Coverage and Performance Analysis is provided. A Keil ULINK*pro* or Segger J-Trace debug adapter is needed. The TI MSP432-E401 series has ETM. See www.keil.com/appnotes/docs/apnt_309.asp

5) Download and Install Keil MDK Core Software:

Download MDK-Core Version 5

1. Download MDK 5.25 or later from the Keil website. www.keil.com/mdk5/install
2. Install MDK into the default folder. You can install into any folder, but this tutorial uses the default C:\Keil_v5.
3. We recommend you use the default folders for this tutorial. We will later put the examples in C:\00MDK\.
4. If you install MDK or the examples into a different folder, you will have to adjust for the folder location differences.
5. You do not need a debug adapter: just the LaunchPad board, a USB cable and MDK installed on your PC. With a Keil ULINK2, ULINK-ME, ULINK*pro* or a Segger J-Link, you can use Serial Wire Viewer. A ULINK*pro* adds ETM instruction trace. A ULINLK*plus* adds Power Measurement.
6. You do not need an MDK license for this tutorial. All examples will compile within the 32 K limit.
7. For projects larger than 32K or to evaluate Keil Middleware, you can obtain a one-time free 7 day license in File/License Management. If you are eligible, this button is visible: Evaluate MDK Professional
8. If you need additional time for evaluation purposes, please contact Keil Sales as listed on the last page.

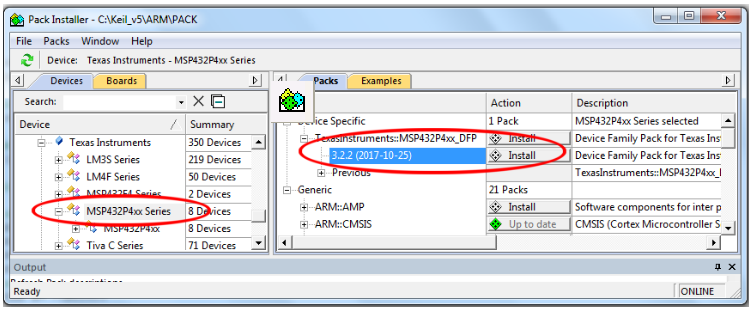
6) Install the MSP432 Software Pack: Version 3.2.2 or later:

A Software Pack contains components such as headers, Flash programming, documents and other files used in a project. The first time µVision is started after initial installation, the "Master Pack" must downloaded into the Pack Installer from the web.

TIP: A Pack is an ordinary zip file with the extension .pack. A few IT departments do not allow an application such as Pack Installer to download a zip file. In this case, download the Pack directly from www.keil.com/dd2/pack or contact Keil tech support to obtain it. Then, install it manually by double clicking on the Pack or select File/Import in Pack Installer.

1. Start µVision.
2. Open the Pack Installer (PI) by clicking on its icon: This window opens up:
1. Select the Devices tab. Highlight Texas Instruments and then MSP432P4xx Series as shown:

TIP: What is entered in the Devices or Boards tabs filters what is displayed in the Packs and Examples tabs.


Pack Installer - C:\Keil_v5\ARM\PACK
File
Packs
Window
Help
Device: Texas Instruments - MSP432P4xx Series
Devices
Boards
Search:
Packs
Examples
Output
Ready
ONLINE

Device	Summary
Texas Instruments	350 Devices
LM3S Series	219 Devices
LM4F Series	50 Devices
MSP432E4 Series	2 Devices
MSP432P4xx Series	8 Devices
MSP432P4xx	8 Devices
Tiva C Series	71 Devices

	Action	Description
Device Specific	1 Pack	MSP432P4xx Series selected
TexasInstruments::MSP432P4xx_DFP	Install	Device Family Pack for Texas Ins
3.2.2 (2017-10-25)	Install	Device Family Pack for Texas Ins
Previous		TexasInstruments::MSP432P4xx_I
Generic	21 Packs	
ARM::AMP	Install	Software components for inter p
ARM::CMSIS	Up to date	CMSIS (Cortex Microcontroller S

2. Select the Packs tab. Install
3. Click on the 3.2.2 Pack (or later) Install.
4. Close the Packs Installer. You can open it any time by clicking on its icon.
5. If a dialog box opens stating: "Software Pack folder has been modified. Reload Packs?" Click Yes.

TIP: A Pack's status will be indicated by the "Up to date" icon: Up to date

The "Update" icon means there is an updated Software Pack available for download. Update

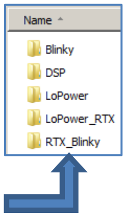
TIP: Select "Check for Updates" or File/Refresh in PI to check for Packs updates. You must be connected to the Internet.

7) Install the Keil MSP432 Examples in C:\00MDK\

1. The MSP432 examples are available on the Keil website where this document is stored.

TIP: For simplicity, we will use the default folder C:\00MDK\ in this tutorial. You can use any folder.

2. Obtain the zip file from www.keil.com/appnotes/docs/apnt_276.asp.
3. Create the folder: C:\00MDK\Boards\TI\MSP432\. Extract the .zip file into it to create these folders:



Blinky
DSP
LoPower
LoPower_RTX
RTX_Blinky

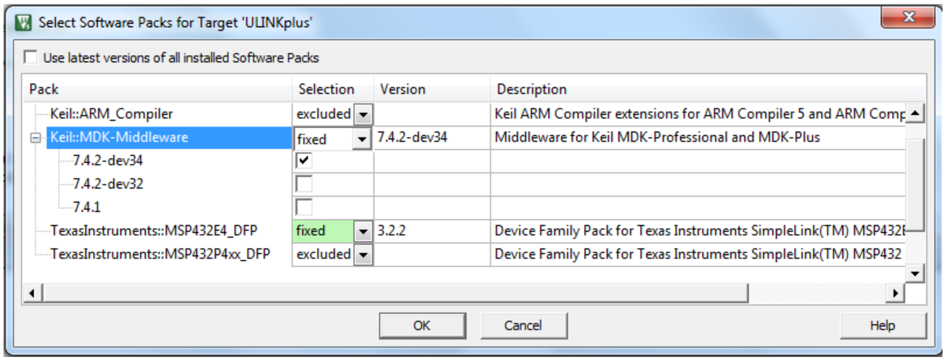
TIP: µVision icon meanings are found here: www.keil.com/support/man/docs/uv4/uv4_ca_filegrp_att.htm

8) Other features of CMSIS-Pack Software Packs: (for reference)

A) Select Software Packs (version selection): Use this to select the Pack version you want to use.

This feature provides the ability to choose various Software Pack versions installed in your computer. You can select the versions you want to use. "Use the latest versions" is the default. This is easily overridden for custom version selection.

1. Open Select Software Pack by clicking on its icon: You can also it open with Project/Manage/Select Software Packs.
2. This window opens up. Note Use latest versions … is selected.
3. Unselect this setting. Expand one of the entries – Keil::MDK – Middleware is a good one to try.

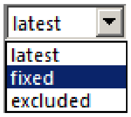


Select Software Packs for Target 'ULINKplus'

☐ Use latest versions of all installed Software Packs

Pack	Selection	Version	Description
Keil::ARM_Compiler	excluded		Keil ARM Compiler extensions for ARM Compiler 5 and ARM Comp
Keil::MDK-Middleware	fixed	7.4.2-dev34	Middleware for Keil MDK-Professional and MDK-Plus
7.4.2-dev34	☑		
7.4.2-dev32	☐		
7.4.1	☐		
TexasInstruments::MSP432E4_DFP	fixed	3.2.2	Device Family Pack for Texas Instruments SimpleLink(TM) MSP432E
TexasInstruments::MSP432P4xx_DFP	excluded		Device Family Pack for Texas Instruments SimpleLink(TM) MSP432

1. The Middleware Pack has three versions installed on this computer at this time: You may see different versions. If there were more than one Pack for the MSP series, they would be visible and selectable.
2. Select **fixed** and then the version you want to use. In this case, 7.4.2dev34.
3. **Re-select *Use latest versions of …***
4. Close this window with Cancel. **Do not make any changes at this time.** We will use the latest version



B) Manage Run-Time Environment: Use this to add various software components to your project.

1. Select Project/Open Project and select C:\00MDK\Boards\TI\MSP432\Blinky\Blinky.uvprojx
2. Click on the Manage RTE icon: The next window opens:
3. Expand various headers and note the selections you can make. A selection made here will automatically select and insert the appropriate source files into your project for your convenience. Core and Startup are very important files.
4. Do not make any changes. Click Cancel to close this window.

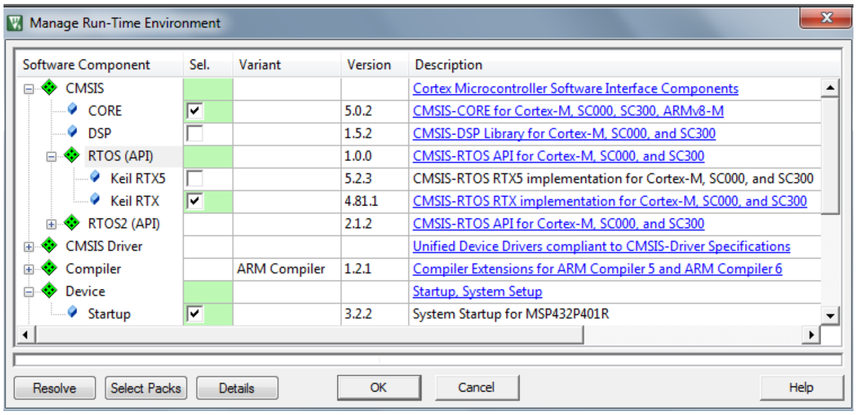
TIP: Different colors represent messages:

🟩☑ Green: all required files are located.

🟨☑ Yellow: some files need to be added. Click the Resolve button to add them automatically or add them manually.

🟥☑ Red: some required files could not be found. Obtain these files or contact Keil technical support for assistance.

The Validation Output area at the bottom of this window gives some information about needed files and current status.



Manage Run-Time Environment

Software Component	Sel.	Variant	Version	Description
CMSIS				Cortex Microcontroller Software Interface Components
CORE	☑		5.0.2	CMSIS-CORE for Cortex-M, SC000, SC300, ARMv8-M
DSP	☐		1.5.2	CMSIS-DSP Library for Cortex-M, SC000, and SC300
RTOS (API)			1.0.0	CMSIS-RTOS API for Cortex-M, SC000, and SC300
Keil RTX5	☐		5.2.3	CMSIS-RTOS RTX5 implementation for Cortex-M, SC000, and SC300
Keil RTX	☑		4.81.1	CMSIS-RTOS RTX implementation for Cortex-M, SC000, and SC300
RTOS2 (API)			2.1.2	CMSIS-RTOS API for Cortex-M, SC000, and SC300
CMSIS Driver				Unified Device Drivers compliant to CMSIS-Driver Specifications
Compiler		ARM Compiler	1.2.1	Compiler Extensions for ARM Compiler 5 and ARM Compiler 6
Device				Startup, System Setup
Startup	☑		3.2.2	System Startup for MSP432P401R

9) Install the XDS110 Drivers and Update the LaunchPad XDS firmware:

If you are going to use the on-board XDS110 debug adapter, the XDS110 drivers ***must*** be installed at C:\ti\.

If you are going to use a Keil ULINK2, ULINK-ME, ULINK*pro* or a J-Link, you do not need to install these drivers.

Download and Install the XDS110 Emulation Software Package (emupack):

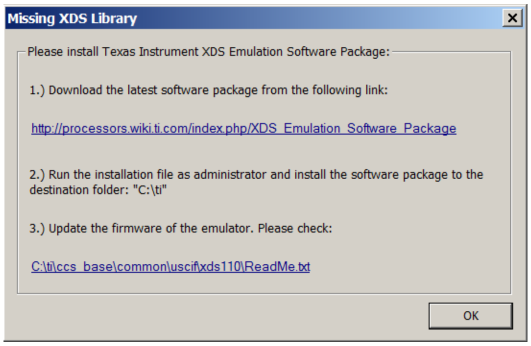
1. Go to http://processors.wiki.ti.com/index.php/XDS_Emulation_Software_Package
2. On this website, download the latest XDS Emulation Software for Windows. You need a free TI account.
3. The filename will be similar to **ti_emupack_setup_6.0.228.1_win_32.exe** *or later*.
4. Run this executable. It will install files to the default folder C:\ti\. Do not change this default folder. µVision looks in this folder for the needed XDS110 driver software.

Install/Update the Firmware in the LaunchPAD XDS110 Processor:

1. Open the ReadMe file: C:\ti\ccs_base\common\uscif\xds110\ReadMe.txt
2. This file contains instructions on how to update the Launchpad firmware using the Windows Command prompt.
3. If the green LED LED102 does not light when the USB cable is connected, the firmware needs to be installed.

TIP: It is a good idea to use the same versions of software and firmware.

TIP: If the XDS110 debug adapter is selected, and if the appropriate XDS110 files are not present in C:\ti\: when you attempt to enter Debug mode in µVision this alert window opens: The above instructions are repeated here to install the drivers and to update the LaunchPad firmware.



Missing XDS Library

Please install Texas Instrument XDS Emulation Software Package:

1.) Download the latest software package from the following link:

http://processors.wiki.ti.com/index.php/XDS_Emulation_Software_Package

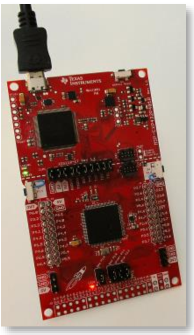
2.) Run the installation file as administrator and install the software package to the destination folder: "C:\ti"

3.) Update the firmware of the emulator. Please check:

C:\ti\ccs_base\common\uscif\xds110\ReadMe.txt

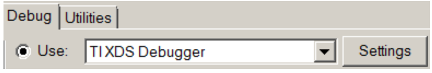
Using the LaunchPad XDS110 Emulator with µVision:

1. Start µVision if it is not already running.
2. Select Project/Open Project and select Blinky.uvprojx in C:\00MDK\Boards\TI\MSP432\Blinky\.
3. Select XDS in the µVision Target Selector: XDS
4. Connect a USB cable to the LaunchPad USB connector and your PC as shown here:
5. J101 must have all 10 jumpers populated.
6. The green LED LED102 will illuminate. Test the XDS110 connection as described below:



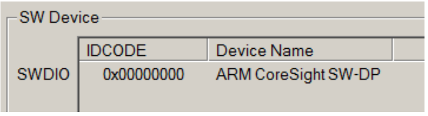
Testing the Debug Connection: You need any valid µVision project open for this step.

1. Select Target Options or ALT-F7 and select the Debug tab: Select your debugger. This selection is for the XDS110 on-board debug adapter:



○ Use: TI XDS Debugger

2. Click on Settings: and the Target Driver Setup window opens:
3. Select SW in the Port box.
4. An IDCODE and Device name will then be displayed indicating a valid connection to the CoreSight DAP.
This means the debug adapter is working correctly !
You can continue with the tutorial.



SW Device

	IDCODE	Device Name
SWDIO	0x00000000	ARM CoreSight SW-DP

5. If nothing or an error is displayed in this SW Device box, this ***must*** be corrected before you can continue. Check your connections and settings. Reinstall the firmware. All 10 jumpers of J101 must be installed for XDS110 operation. Remove RST through TDI for an external adapter such as any Keil ULINK or Segger J-Link.
6. Click on OK twice to return to the µVision main menu.

TIP: You can use these steps to test the connection of any debug adapter such as any ULINK or a J-Link if there is a conflict.

TIP: The TI XDS110 Emulator does not yet support Serial Wire Viewer (SWV). For SWV, use a Keil ULINK2, ULINK-ME, ULINK*pro* or a J-Link as described on the next page. SWV is a very useful debugging tool and is worth using.

10) Connecting External Debug Adapters:

You can use an external debug adapter. The main benefit is the ability to use Serial Wire Viewer (SWV). SWV is data trace and will display exceptions and interrupts, date read and write operations, graphical display of variables and more. . ETM instruction trace is a powerful debugging technology and provides program flow debugging, Performance Analysis and Code Coverage. The source windows display code as it was written and the ETM window shows it as it was executed.

A) Keil ULINK2:

1. Select ULINK2 in the Target Selector:
UKINK2
2. Install the provided 10 pin connector cable. You will need to remove the ULINK2 cover to install this cable.
7. Remove jumpers RST, TMS, TCK, TDO and TDI on J101 to disconnect the onboard XDS110 debug adapter.
8. Connect the ULINK2 to J8 as shown on page 1. Power the LaunchPad with a USB cable.
3. Test the connection as described on the previous page.

B) Keil ULINK*pro*:

1. Install the provided 20 to 10 pin connector cable. You will need to remove the ULINK*pro* cover to install this cable.
2. **Make sure you do not remove or disturb the internal ULINK*pro* battery.**
3. Select ULINKpro in the Target Selector:
ULINKpro
4. Remove jumpers RST, TMS, TCK, TDO and TDI on J101 to disconnect the onboard XDS110 debug adapter.
5. Connect the ULINK*pro* to J8 as shown on page 3. Power the LaunchPad with a USB cable.
6. Test the connection as described on the previous page.

C) Segger J-Link:

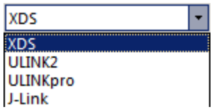
1. J-Link USB drivers must be manually installed (unless they are already installed). The drivers are located here: C:\Keil_v5\ARM\Segger\USBDriver\InstDrivers.exe or C:\Keil_v5\ARM\Segger\USBDriver\CDC\.
2. No indication this install is completed is given by the install program. www.segger.com also provides install software.
3. Select J-Link:
J-Link
4. Remove jumpers RST, TMS, TCK, TDO and TDI on J101 to disconnect the onboard XDS110 debug adapter.
5. Connect the J-Link with the 10 pin adapter. Power the LaunchPad with a USB cable to J8.
6. Test the connection as described on the previous page.

Adding a Target hardware configuration: (for reference)

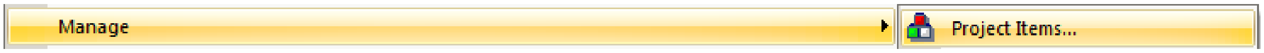
The Target Options configures many items related to your project and processor.

All these can be individually saved and recalled with the Select Target drop down menu as shown above.

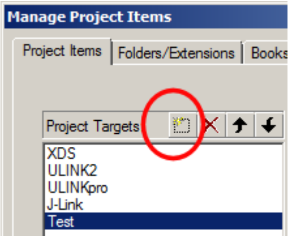
1. In Edit mode, in the Select Target box, choose the setting you want to be the template for the new one you are about to create.



2. Select Open Project/Manage and then select Project Items… or click



3. Click on the Insert icon or the Insert key and enter a name:
4. Click on close and this name will now be available in the Select Target box.
5. Any changes you make in any Target Options tab will be saved and recalled by this target name.



11) *Blinky* example program using the MSP432 LaunchPad:

Using the example Blinky, we will run our first program on the MSP432 Launchpad.

1. Select Project/Open Project and select Blinky.uvprojx in C:\00MDK\Boards\TI\MSP432\Blinky\.
2. Select your debug adapter in the Select Target box in µVision as described on the previous two pages.
3. Connect an external debug adapter to J8 if used. Remove J101 jumpers RST, TMS, TCK, TDO and TDI to disconnect the onboard XDS110 debug adapter. For the on-board XDS110, all ten J101 jumpers must be installed.
4. Power the board with a USB cable from your PC.
5. Compile the source files by clicking on the Rebuild icon. You can also use the Build icon beside it.
6. Enter Debug mode by clicking on the Debug icon. The Flash memory will be programmed. Progress will be indicated in the Output Window. Select OK if the Evaluation Mode box appears.
7. Click on the RUN icon. **Note:** you stop the program with the STOP icon.

The four LEDs will now blink in sequence on the LaunchPad board.

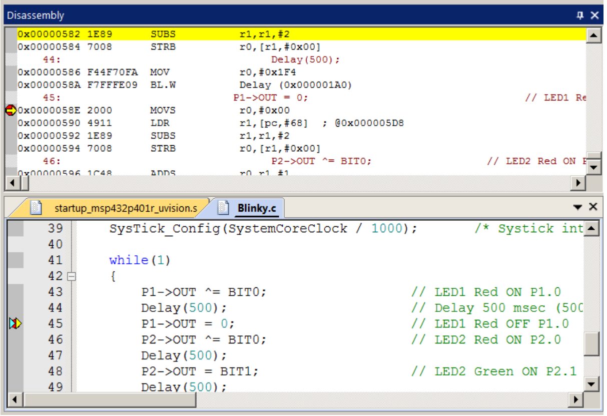
Now you know how to compile a program, program it into the TI processor Flash, run it and stop it !

Note: The board will start Blinky stand-alone. Blinky is now permanently programmed in the Flash until reprogrammed.

12) Hardware Breakpoints:

The MSP432 has six hardware breakpoints that can be set or unset on-the-fly while the program is running. This works with a XDS110, Keil ULINK2, ULINK-ME, ULINK*pro* or a J-Link debug adapter.

1. With Blinky running, in the Blinky.c window, click on a darker grey block on the left on a suitable part of the source code. This grey box indicates assembly instructions are present at these points. Inside the while (1) loop inside the main() function between near lines 43 through 52 are suitable places: You can also click in the Disassembly window to set or unset a breakpoint on a specific assembly instruction.
2. A red circle will appear and the program will presently stop if the CPU tries to access this instruction.
3. Note the breakpoint is displayed in both the Disassembly and Blinky.c windows as shown here:
4. Set a second breakpoint in the while() loop as before.
5. Every time you click on the RUN icon the program will run until the breakpoint is again encountered.
6. The yellow arrow is the current program counter value.
7. Open Debug/Breakpoints or Ctrl-B and you can see any breakpoints set. You can unselect them or delete them here.
8. Delete all breakpoints and close the Breakpoint window.
9. Clicking in the source window will indicate the appropriate code line in the Disassembly window and vice versa. This relationship is indicated by the cyan arrow and the yellow highlight:
10. **Make sure you remove the breakpoints !**



```
0x00000582 1E89      SUBS     r1,r1,#2
0x00000584 7008      STRB     r0,[r1,#0x00]
    44:                           Delay(500);
0x00000586 F44F70FA  MOV      r0,#0x1F4
0x0000058A F7FFFE09  BL.W     Delay (0x000001A0)
    45:                      P1->OUT = 0;                              // LED1 Re
0x0000058E 2000      MOVS     r0,#0x00
0x00000590 4911      LDR      r1,[pc,#68]  ; @0x000005D8
0x00000592 1E89      SUBS     r1,r1,#2
0x00000594 7008      STRB     r0,[r1,#0x00]
    46:                           P2->OUT ^= BIT0;                // LED2 Red ON P
0x00000596 1C48      ADDS     r0,r1,#1
```

```
39     SysTick_Config(SystemCoreClock / 1000);        /* Systick int
40
41     while(1)
42     {
43         P1->OUT ^= BIT0;                           // LED1 Red ON P1.0
44         Delay(500);                                // Delay 500 msec (500
45         P1->OUT = 0;                               // LED1 Red OFF P1.0
46         P2->OUT ^= BIT0;                           // LED2 Red ON P2.0
47         Delay(500);
48         P2->OUT = BIT1;                            // LED2 Green ON P2.1
49         Delay(500);
```

TIP: ARM hardware breakpoints do **not** execute the instruction they are set to and land on. CoreSight hardware breakpoints are no-skid. This is an important feature for effective debugging as is the ability to set/unset them while the program runs.

Single-Stepping:

1. With Blinky.c in focus (click anywhere inside Blinky.c), click on the Step In icon or F11 a few times: The program counter steps one C line at a time. The yellow arrow indicates the next C line or instruction to be executed.
2. Click on the top margin of the Disassembly window to bring it into focus. Clicking Step Into now jumps the program counter PC one assembly instruction at a time.

13) Watch and Memory Windows and how to use them:

The Watch and Memory windows will display updated variable values in real-time. It does this using ARM CoreSight DAP debugging technology that is part of Cortex-M processors. It is also possible to “put” or insert values into a Memory window in real-time. You can right click on a variable and select Add *varname* to.. and select the appropriate window. DAP works with all debug adapters including XDS110.

TIP: It is possible to change a variable n a Watch window while the program runs if the value is not changing too fast. If it is too difficult to do this, just use a Memory window as described below.

Watch window:

Add a global variable: Call Stack, Watch and Memory windows can’t see local variables unless stopped in their function.

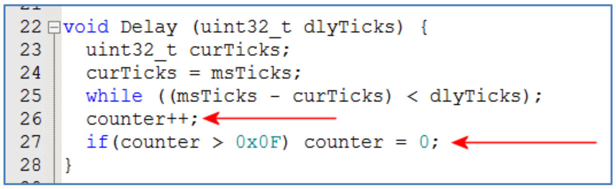
1. Stop the processor and exit Debug mode.
2. In Blinky.c, declare a global variable called `counter` near line 11:

   unsigned int counter = 0;

3. Add these statements near Line 26 in the Delay function:

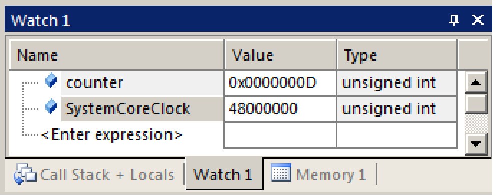
   counter++;

   if (counter > 0x0F) counter = 0;



```
22 void Delay (uint32_t dlyTicks) {
23   uint32_t curTicks;
24   curTicks = msTicks;
25   while ((msTicks - curTicks) < dlyTicks);
26   counter++;
27   if(counter > 0x0F) counter = 0;
28 }
```

4. Select File/Save All.
5. Click on Rebuild . There will be no errors or warnings: If there are, please fix them before continuing.
6. Enter Debug mode. The Flash will be programmed. Click on RUN . You can configure a Watch or Memory window while the program is running. The program must be stopped to remove a variable from a Watch window.
7. In Blinky.c, right click on `counter` and select Add 'counter' to … and select Watch 1. Watch 1 will automatically open. `counter` will be displayed as shown here:
8. `counter` is immediately updated in real time !
9. Double click on <Enter expression> and enter **SystemCoreClock.** Press Enter. Right click on SystemCoreClock and unselect **Hexadecimal Display**. The processor clock speed is displayed: 48 MHz. SystemCoreClock is created in system_msp432p401r.c

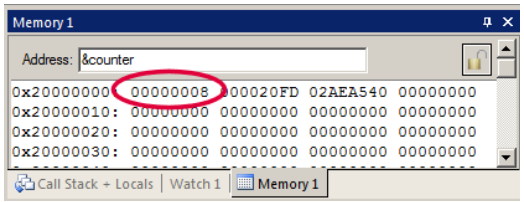


Name	Value	Type
counter	0x0000000D	unsigned int
SystemCoreClock	48000000	unsigned int
<Enter expression>		

TIP: A Watch or Memory window can display and update global and static variables, structures, raw addresses and peripheral addresses while the program is running. These are unable to display local variables because these are typically stored in a CPU register which cannot be read by µVision in real-time. To view a local variable in these windows, convert it to a static or global variable.

Memory window:

1. Right click on `counter` in Blinky.c and select Add 'counter' to … and select Memory 1.
2. Note the value of `counter` is displaying its value in Memory 1 as if it is a pointer. This is useful to see what address a pointer is pointing to, but this not what we want to see at this time.
3. Add an ampersand “&” in front of the variable name and press Enter. The physical address here is 0x2000_0000.
4. Right click in the Memory window and select Unsigned/Int.
5. The data contents of `counter` is displayed as shown here:
6. Both the Watch and Memory windows are updated in real-time.
7. Right-click the mouse cursor over the desired data field and select Modify Memory. You can change a memory or variable on-the-fly while the program is still running.



Address: &counter

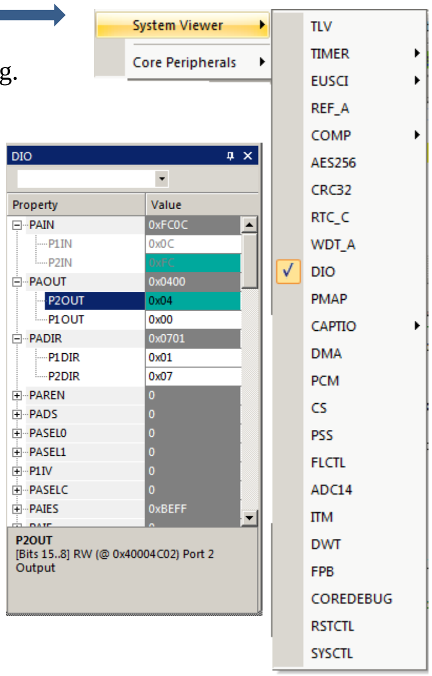
```
0x20000000: 00000008 000020FD 02AEA540 00000000
0x20000010: 00000000 00000000 00000000 00000000
0x20000020: 00000000 00000000 00000000 00000000
0x20000030: 00000000 00000000 00000000 00000000
```

TIP: No CPU cycles are usually used to perform these operations.

TIP: To view variables and their location use the Symbols window. Select View/Symbols Window while in Debug mode.

14) System Viewer (SV):

System Viewer provides the ability to view registers in the CPU core and in peripherals. In most cases, these peripherals are updated in real-time while your program is running. These Views are available only while in Debug mode. There are two ways to access these Views: **a)** View/System Viewer and **b)** Peripherals/System Viewer. In the Peripheral/Viewer menu, the Core Peripherals are also available: Note the various peripherals available.



Property	Value
PAIN	0xFC0C
P1IN	0x0C
P2IN	0xFC
PAOUT	0x0400
P2OUT	0x04
P1OUT	0x00
PADIR	0x0701
P1DIR	0x01
P2DIR	0x07
PAREN	0
PADS	0
PASEL0	0
PASEL1	0
P1IV	0
PASELC	0
PAIES	0xBEFF

P2OUT
[Bits 15..8] RW (@ 0x40004C02) Port 2 Output

1. Click on RUN. You can open SV windows when your program is running.

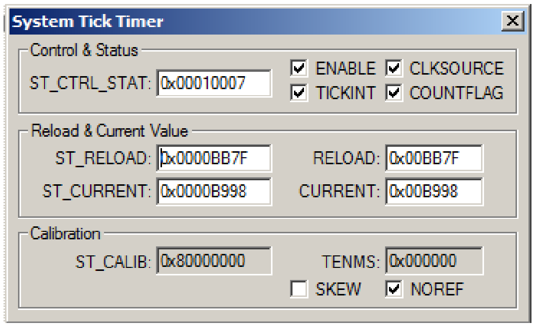
GPIO Port P1 and P2:

2. Select Peripherals/System Viewer and then DIO.
3. This window opens up. Expand PAIN, PAOUT and PADIR:
4. You can now see P1 and P2 update as the LEDs blink in succession:
5. These are updated periodically, not when the value changes. To view a register when it changes, you need to use Serial Wire Viewer and the Logic Analyzer. A ULINK2, ULINK*pro* or a J-Link is needed.
6. You can change the values in the System Viewer on-the-fly. In this case, the values are updated quickly so it is hard to see the change.

TIP: If you click on a register in the Property column, a description about this register will appear at the bottom of the SV window. P2OUT is shown here:

SysTick Timer: This program uses the SysTick timer as a timer for the Delay function.. This is configured and activated in Blinky.c near line 39: SysTick_Config(SystemCoreClock / 1000);

1. Select Peripherals/Core Peripherals and then select System Tick Timer.
2. The System Tick Timer window shown below opens:
3. Note it also updates in real-time while your program runs. These windows use the same CoreSight DAP technology as the Watch and Memory windows. Do not confuse this DAP (Debug Access Port) with CMSIS-DAP.
4. Note the ST_RELOAD and RELOAD registers. This is the reload register value set by the SysTick_Config function.
5. Note that it is set to 0xBB7F. This is the hex value of 48,000,000/1000-1 that is programmed into SysTick_Config() in main() in Blinky.c. The CPU clock in this example is 48 MHz. Changing the variable passed to this function is how you change how often the SysTick timer creates its interrupt 15. See the TIP below for SystemCoreClock.
6. In the RELOAD register in the SysTick window, *while the program is running*, type in 0x1000, press Enter and click inside ST_RELOAD ! (or the other way around)
7. The blinking LEDs will speed up. This will convince you of the power of ARM CoreSight debugging.
8. Replace RELOAD with 0x5DBF. A CPU RESET and RUN will also do this.
9. You can look at other Peripherals contained in the System View windows.
10. When you are done, stop the program and close all the System Viewer windows that are open.
11. Exit Debug mode.



ST_CTRL_STAT: 0x00010007 ☑ ENABLE ☑ CLKSOURCE ☑ TICKINT ☑ COUNTFLAG

ST_RELOAD: 0x0000BB7F RELOAD: 0x00BB7F

ST_CURRENT: 0x0000B998 CURRENT: 0x00B998

ST_CALIB: 0x80000000 TENMS: 0x000000

☐ SKEW ☑ NOREF

TIP: It is true: you can modify values in the System Viewer while the program is running. This is very useful for making slight timing value changes instead of the usual modify, compile, program, run cycle.

You must make sure a given peripheral register allows for and will properly react to such a change. Changing such values indiscriminately is a good way to cause serious and difficult to find problems.

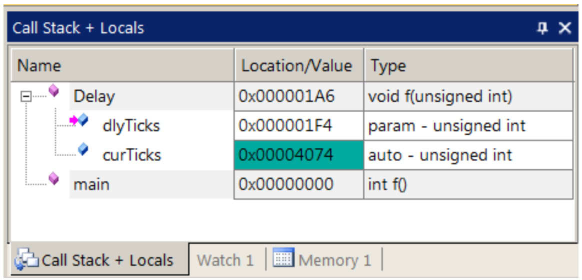
TIP: You can enter the global variable SystemCoreClock in the Watch window to see its frequency. SystemCoreClock is created in the CMSIS-CORE file system_msp432p401r.c.

15) Call Stack + Locals Window:

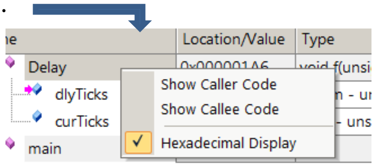
The Call Stack and Locals windows are incorporated into one integrated window. Whenever the program is stopped, the Call Stack + Locals window will display call stack contents as well as any local variables located in the active function.

If possible, the values of the local variables will be displayed and if not the message <not in scope> will be displayed. The Call + Stack window presence can be toggled by selecting View/Call Stack Window in the main µVision window when in Debug mode.

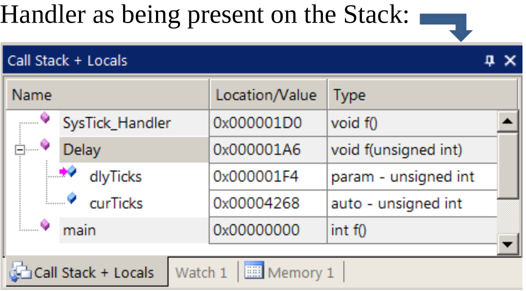
1. Use the same Blinky project as from the previous page.
2. Enter the Debug mode . The Flash will be programmed only if changes to the program executable were made.
3. Click on RUN. After a second or two, stop the CPU.
4. Click on the Call Stack + Locals tab if necessary to open it. Expand some of the entries.
5. Shown is a Call Stack + Locals window similar to this one: The program will most likely stop in the Delay function.
6. The functions are displayed in the order they were called.
7. Note the two local variables in Delay() are displayed with their values since the program is stopped in this function.
8. Right click on the Delay name and select either Callee or Caller code and this will be highlighted in the source and Disassembly windows.



Name	Location/Value	Type
Delay	0x000001A6	void f(unsigned int)
dlyTicks	0x000001F4	param - unsigned int
curTicks	0x00004074	auto - unsigned int
main	0x00000000	int f()



9. Click on Step Out and the Delay() function will not be on the stack as evidenced in the Call Stack window.
10. Set a breakpoint in the SysTick handler near line 16 or 17 in Blinky.c.
11. Click on RUN and the program will run to SysTick_Handler() and stop.
12. The Call Stack window will now show main(), Delay() and SysTick_Handler as being present on the Stack:
13. Remove any breakpoints. Click on each breakpoint to remove it or select Ctrl-B, Kill All and then Close.
14. Stop the CPU and exit Debug mode. .
15. Disconnect the USB cable for the next page.



Name	Location/Value	Type
SysTick_Handler	0x000001D0	void f()
Delay	0x000001A6	void f(unsigned int)
dlyTicks	0x000001F4	param - unsigned int
curTicks	0x00004268	auto - unsigned int
main	0x00000000	int f()

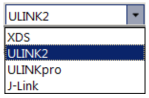
TIP: You can modify a variable value in the Call Stack & Locals window only when the program is stopped.

TIP: This window is only valid when the processor is halted. It does not update while the program is running. Any local variable values are visible only when they are in scope. This is also true for locals in the Watch and Memory windows.

16) Watchpoints: *Conditional Breakpoints* **Watchpoints are only supported with a ULINK or J-Link.**

The MSP432 Cortex-M4 processor has two Watchpoints. Watchpoints can be thought of as conditional breakpoints. Watchpoints are also referred to as Access Breaks in Keil documents. Cortex-M3/M4/M7 Watchpoints are not intrusive as they are implemented in CoreSight™ hardware. XDS110 does not currently support Watchpoints in µVision.

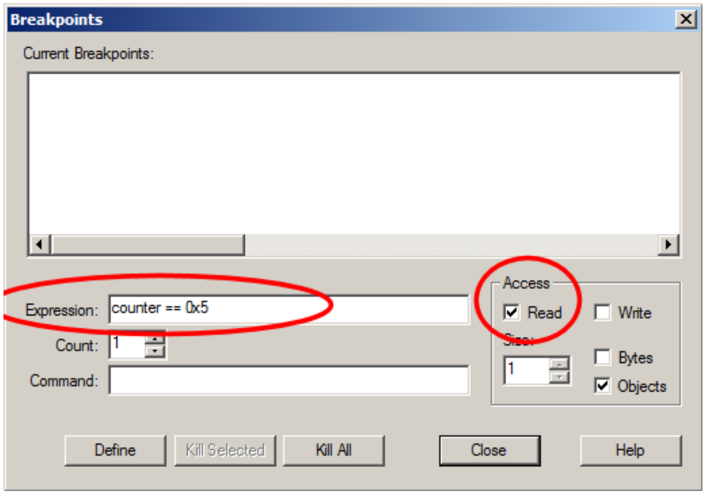
1. Remove J101 jumpers RST, TMS, TCK, TDO and TDI to disconnect the onboard XDS110 debug adapter.
2. Connect a ULINK2, ULINK-ME, ULINK*pro* or a J-Link to the CoreSight connector J8.
3. Power the TI board with a USB cable to your PC.
4. In the Target dialog box, select your debug adapter:



5. Use the same Blinky configuration as the previous page. Enter Debug mode .

TIP: You can configure Watchpoints while the program is running if you are using any Keil ULINK.

6. We will use the global variable **`counter`** you created in Blinky.c to explore Watchpoints.
7. In the main µVision window, select Debug/Breakpoints or press Ctrl-B.
8. Select Access to Read.
9. In the Expression box enter: "**`counter`** == **0x5**" without the quotes. This window will display:



Expression: counter == 0x5

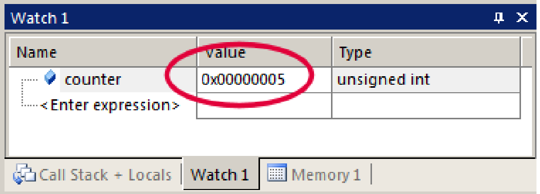
Count: 1

Command:

Access: ☑ Read □ Write

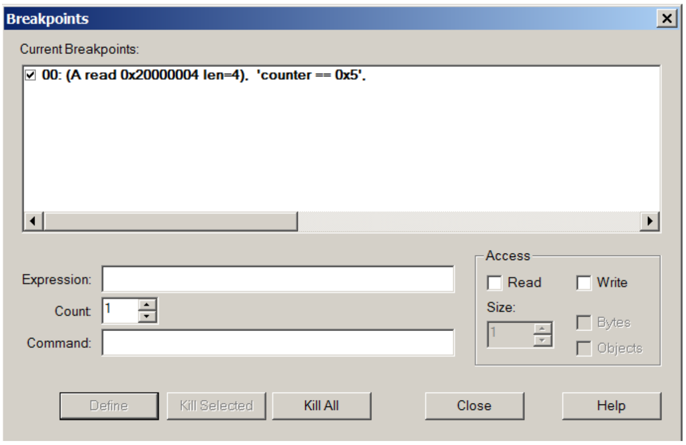
Size: 1 □ Bytes ☑ Objects

10. Click on Define or press Enter and the expression will be accepted as shown below in the Breakpoints window at the bottom of this page:
11. Click on Close.
12. Enter the variable **`counter`** in Watch 1 if it is not already there.
13. Click on RUN .
14. When **`counter`** equals 0x5, the Watchpoint will stop the program. See Watch 1 shown below:
15. Currently you can enter an equality test (= =) as a Watchpoint. Entering no test will stop the processor when the address is accessed with no regard to its value.
16. To repeat this exercise, change **`counter`** to something other than 0x05 in the Watch window and click on RUN. Or click on RUN a few times to clear past the Watchpoint comparator.



Name	Value	Type
counter	0x00000005	unsigned int
<Enter expression>		

17. Stop the CPU if it is running.
18. Select Debug/Breakpoints (or Ctrl-B) and delete the Watchpoint with Kill All.
19. Select Close.
20. Exit Debug mode.



☑ 00: (A read 0x20000004 len=4). 'counter == 0x5'.

Expression:

Count: 1

Command:

Access: □ Read □ Write

Size: 1 □ Bytes □ Objects

TIP: To edit a Watchpoint, double-click on it in the Breakpoints window and its information will be dropped down into the configuration area. Clicking on Define will create another Watchpoint. You must then delete the old one by highlighting it and click on Kill Selected or try the next TIP:

TIP: The checkbox beside the expression allows you to temporarily unselect or disable a Watchpoint without deleting it.

TIP: Raw addresses can be used with a Watchpoint. An example is: *((unsigned long *)0x20000004)

17) Serial Wire Viewer (SWV): (you need a ULINK2, ULINK*pro* or a J-Link for this page)

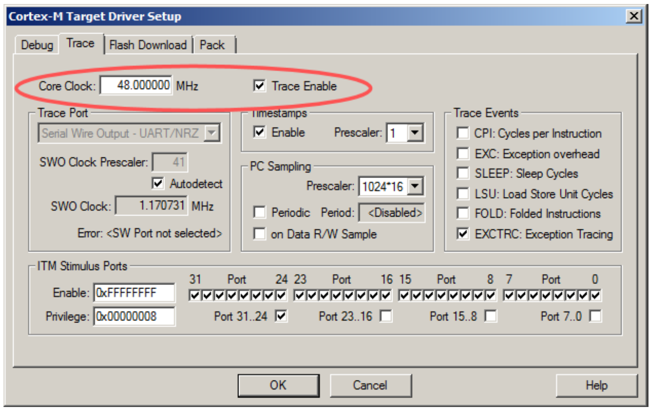
Serial Wire Viewer is a data trace found on most Cortex-M3/M4/M7 processors and is a separate technology from the DAP reads and writes used in the Watch and other windows. It can display variables in a graphical format, data reads and writes, exceptions (including interrupts), PC sampler and various hardware counters. The ITM printf feature uses SWV.

Configure SWV: A ULINK2, ULINK*pro* and J-Link each have slightly different but similar menus.

1. Connect a ULINK2, ULINK*pro* or J-Link to J8 which is a CoreSight 10 pin debug connector.
2. Remove J101 jumpers RST, TMS, TCK, TDO and TDI to disconnect the onboard XDS110 debug adapter.
3. Select your adapter in the Select Target box. This is for ULINK2: ULINK2
4. Click on the Options icon next to the Target box in the main µVision window.
5. Select the Debug tab and then click the Settings box next to ULINK2/ME Cortex Debugger dialog.
6. In the Debug window that opens, set Port: to SW. SWV will not work with JTAG. Only with SWD.
7. Click on the Trace tab to open the Trace window.
8. Select Trace Enable.
9. Set Core Clock: to 48 MHz. With a ULINK2 or J-Link, this ***must*** be set to the exact CPU frequency.
10. Leave everything else at default as shown here:
11. Click on OK twice to return to the µVision main menu. ***The Serial Wire Viewer is now configured in µVision.***
12. Click on File/Save All or
13. Power the board with a USB cable to your PC.
14. Enter Debug mode . Click on RUN .

Configure the Logic Analyzer (LA) with counter:

1. In Blinky.c, right click on `counter` and select Add 'counter' to … and select Logic Analyzer. The LA window will open.



Cortex-M Target Driver Setup

Debug | Trace | Flash Download | Pack

Core Clock: 48.000000 MHz ☑ Trace Enable

Trace Port: Serial Wire Output - UART/NRZ

SWO Clock Prescaler: 41 ☑ Autodetect

SWO Clock: 1.170731 MHz

Error: <SW Port not selected>

Timestamps: ☑ Enable Prescaler: 1

PC Sampling: Prescaler: 1024*16 ☐ Periodic Period: <Disabled> ☐ on Data R/W Sample

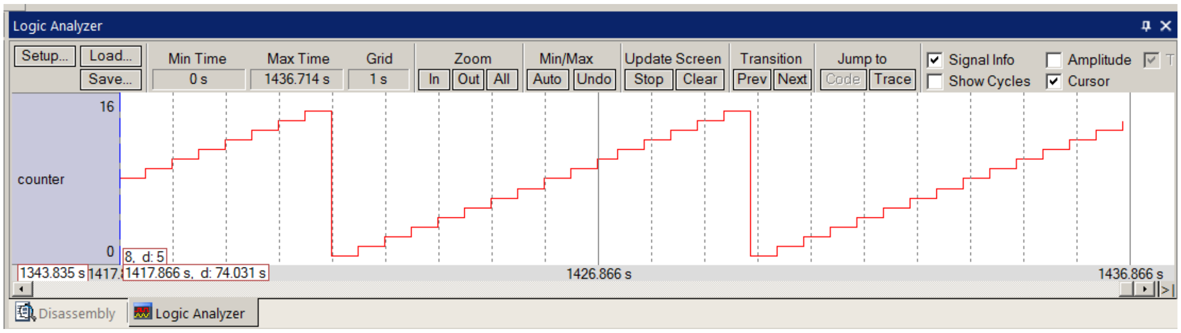
Trace Events:
☐ CPI: Cycles per Instruction
☐ EXC: Exception overhead
☐ SLEEP: Sleep Cycles
☐ LSU: Load Store Unit Cycles
☐ FOLD: Folded Instructions
☑ EXCTRC: Exception Tracing

ITM Stimulus Ports: Enable: 0xFFFFFFFF Privilege: 0x00000008

Port 31..24 ☑ Port 23..16 ☐ Port 15..8 ☐ Port 7..0 ☐

TIP: If you are unable to enter a variable in the LA, this usually means SWV is not configured properly. Number one reason is the Core Clock: value is incorrectly set for ULINK2 and J-Link. A ULINK*pro* will work with the wrong clock value but timing values displayed in µVision will be incorrect. This is because ULINK*pro* uses Manchester encoding instead of UART.

2. In the Logic Analyzer (LA), click on Setup:
3. Set the Display Range Max: to 0x10. Close this window.
4. Click on Zoom out to get an appropriate display as shown below:
5. counter is displayed and updated in real-time. You can display up to four variables.
6. Select Cursor and Signal Info to make time and voltage measurements. Stop the Update Screen if needed. This enables you to examine the LA contents and have the program continue running.



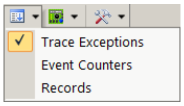
Logic Analyzer

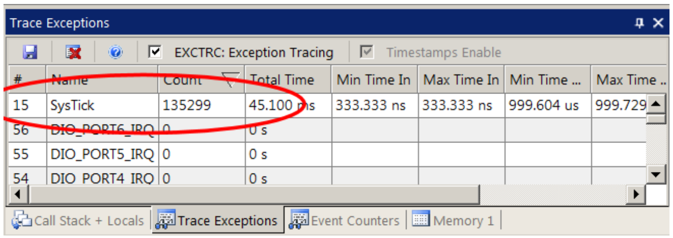
Min Time: 0 s Max Time: 1436.714 s Grid: 1 s

☑ Signal Info ☐ Show Cycles ☐ Amplitude ☑ Cursor

Display Exceptions (including interrupts):

1. Open the Trace Exceptions window by clicking on its tab. If it is not visible, select View/Trace/Trace Exceptions. You can also click on the small arrow beside the Trace icon and select Trace Exceptions.
2. Click on the Count column header and SysTick will come to the top as shown below:
3. Note other interrupts are labelled by name along with their interrupt numbers.
4. This window updates in real-time and uses SWV.





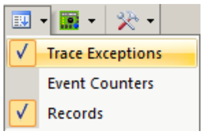
☑ EXCTRC: Exception Tracing ☑ Timestamps Enable

#	Name	Count	Total Time	Min Time In	Max Time In	Min Time ...	Max Time ..
15	SysTick	135299	45.100 ms	333.333 ns	333.333 ns	999.604 us	999.729
56	DIO_PORT6_IRQ	0	0 s				
55	DIO_PORT5_IRQ	0	0 s				
54	DIO_PORT4_IRQ	0	0 s				

> **Note:** The directions below are for using a ULINK2. A Keil ULINK*pro* or a Segger J-Link will require slightly different operations. Additionally, the program must be stopped to display any trace frames and started again to show now ones. The J-Link does not currently display data reads or writes.

Display Trace Records:

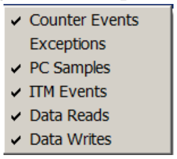
1. Select the small arrow beside the Trace icon and select Records as shown here: (for ULINK2)



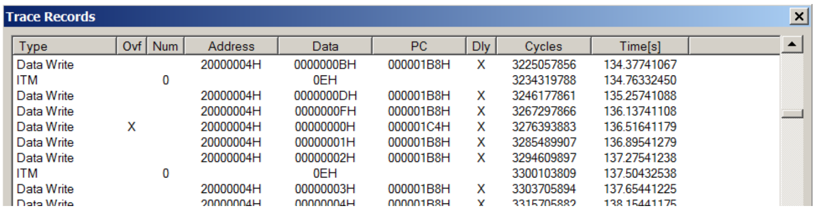
2. SysTick Exception 15 will display in the Trace Records window. There are data writes present but they are swamped out by the more numerous SysTick interrupts..

TIP: With ULINK*pro* or J-Link, the program must be stopped to see the Trace Records window.

3. For ULINK2, right click and unselect Exceptions.
4. For ULINK*pro*, select Data Writes.



5. Now the Data Writes to counter are displayed (they were swamped out before and are not easy to see.). See below:
6. Note the numerous "x" in DLY and Ovf columns. This indicates data overflow out the single bit SWO port.
7. In the Trace Exceptions window, unselect EXCTRC:.
8. The collection of exceptions including interrupts will not be sent out the SWO port. This will reduce overloading.
9. Double click in the ULINK2 Trace Records to clear it. The "x" are gone indicating much less overloading.



Type	Ovf	Num	Address	Data	PC	Dly	Cycles	Time[s]
Data Write			20000004H	0000000BH	000001B8H	X	3225057856	134.37741067
ITM		0		0EH			3234319788	134.76332450
Data Write			20000004H	0000000DH	000001B8H	X	3246177861	135.25741088
Data Write			20000004H	0000000FH	000001B8H	X	3267297866	136.13741108
Data Write	X		20000004H	00000000H	000001C4H	X	3276393883	136.51641179
Data Write			20000004H	00000001H	000001B8H	X	3285489907	136.89541279
Data Write			20000004H	00000002H	000001B8H	X	3294609897	137.27541238
ITM		0		0EH			3300103809	137.50432538
Data Write			20000004H	00000003H	000001B8H	X	3303705894	137.65441225
Data Write			20000004H	00000004H	000001B8H	X	3315705882	138.15441175

10. In the main µVision menu, select Debug/Debug Settings and select the Trace tab.
11. Select on Data R/W Sample. This will add the location of the write in the program. Click Close.
12. Allow "take on new values" and click on RUN.
13. Double click in the Trace Records window to clear it. Now the PC column in Trace Records will now be filled.
14. In the Trace Records window above, the first line means:

    A Data Write occurred to memory 0x2000 0004, with the data value 0x0B by the assembly instruction located at 0x 0000 01B8 at the Cycles or Time indicated.

15. Select Click on STOP . Exit Debug mode. .

TIP: You must stop program execution with a ULINK*pro* or a J-Link to update trace frames. The LA and Exceptions windows still update in real time. ULINK2 updates Trace Records continuously.

a ULINK*pro* provides the best SWV operation with least amount of overloading.

18) *RTX_Blinky* Example Program with Keil RTX RTOS: A Stepper Motor example

Keil provides RTX, a full feature RTOS as a component of MDK. RTX comes with a BSD license. RTX with source code is provided in all versions of MDK as is all documentation. C:\Keil_v5\ARM\Pack\ARM\CMSIS\x.x.x\CMSIS\RTOS\RTX.

For CMSIS 5, see https://github.com/ARM-software/CMSIS_5

Running the RTX Program:

RTX_Blinky has target options pre-configured for the TI XDS110, ULINK2, ULINK*pro* and J-Link.

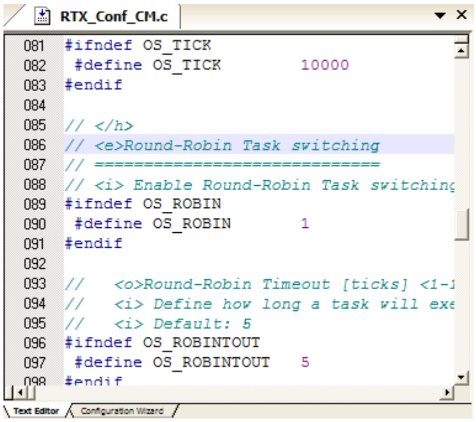
1. If you are using the on-board TI XDS110: make sure all J101 jumpers are populated.
2. If using a ULINK2, ULINK*pro* or a J-Link, connect it to J8. Remove J101 jumpers RST, TMS, TCK, TDO and TDI. This action disconnects the XDS110 from the processor.
3. Connect a USB cable to the LaunchPad board and to your PC to power the board.

1. Start µVision and select Project/Open Project.
2. Open C:\00MDK\Boards\TI\MSP432\RTX_Blinky\Blinky.uvprojx.
3. Select your debug adapter: This is for the TI XDS110: XDS
4. Compile the source files . They will compile with no errors or warnings.
5. Enter Debug mode . The Flash will be programmed with a progress bar.
6. Click on RUN.
7. Four LEDs will blink in sequence indicating the four waveforms of a stepper motor driver changing.
8. Click on STOP .

TIP: A blog on RTX: https://e2e.ti.com/blogs_/b/msp430blog/archive/2015/06/09/using-arm-keil-rtx-with-msp432-mcus

The Configuration Wizard for RTX:

1. Click on the RTX_Conf_CM.c source file tab as shown below on the left. You can open it with File/Open.
2. Click on Configuration Wizard at the bottom and your view will change to the Configuration Wizard.
3. Open up the individual directories to show the various configuration items available. Do not change anything !
4. See www.keil.com/support/docs/2735.htm for instructions on how to add Configuration Wizard to your own sources.

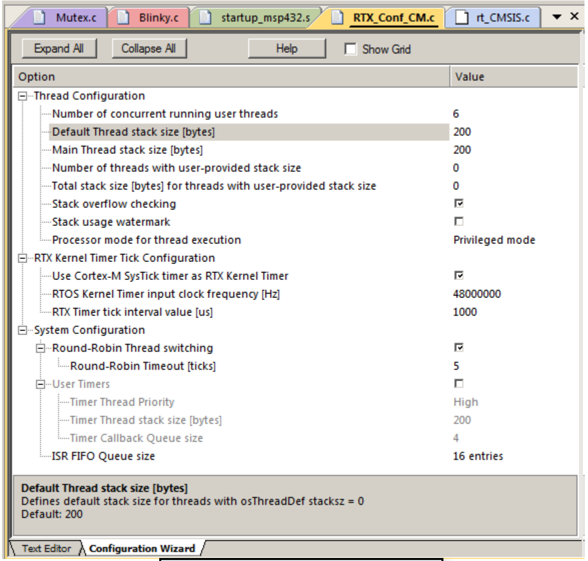


```
#ifndef OS_TICK
 #define OS_TICK        10000
#endif

// </h>
// <e>Round-Robin Task switching
// =============================
// <i> Enable Round-Robin Task switching
#ifndef OS_ROBIN
 #define OS_ROBIN       1
#endif

//   <o>Round-Robin Timeout [ticks] <1-1
//   <i> Define how long a task will exe
//   <i> Default: 5
#ifndef OS_ROBINTOUT
 #define OS_ROBINTOUT   5
#endif
```

Text Editor: Source Code



Option	Value
⊟ Thread Configuration	
Number of concurrent running user threads	6
Default Thread stack size [bytes]	200
Main Thread stack size [bytes]	200
Number of threads with user-provided stack size	0
Total stack size [bytes] for threads with user-provided stack size	0
Stack overflow checking	☑
Stack usage watermark	□
Processor mode for thread execution	Privileged mode
⊟ RTX Kernel Timer Tick Configuration	
Use Cortex-M SysTick timer as RTX Kernel Timer	☑
RTOS Kernel Timer input clock frequency [Hz]	48000000
RTX Timer tick interval value [us]	1000
⊟ System Configuration	
⊟ Round-Robin Thread switching	☑
Round-Robin Timeout [ticks]	5
⊟ User Timers	□
Timer Thread Priority	High
Timer Thread stack size [bytes]	200
Timer Callback Queue size	4
ISR FIFO Queue size	16 entries

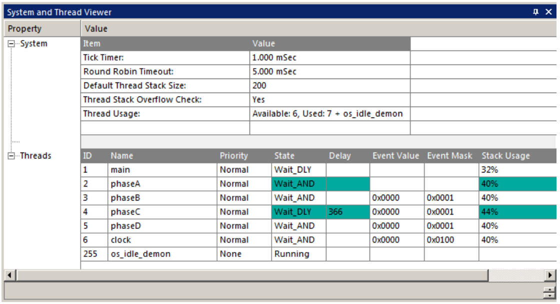
Default Thread stack size [bytes]
Defines default stack size for threads with osThreadDef stacksz = 0
Default: 200

Configuration Wizard

19) RTX Kernel Awareness

Users often want to know the number of the current operating thread and the status of the other threads. This information is usually stored in a structure or memory area by the RTOS. µVision provides two Kernel Awareness windows for RTX.

1. Run RTX_Blinky by clicking on the Run icon.
2. Open Debug/OS Support and select System and Thread Viewer. The window on the right opens up. You might have to grab the window and move it into the center of the screen. Note these values are updated in real-time using the same DAP technology as used in the Watch and Memory windows.
3. Open Debug/OS Support and select Event Viewer. There is probably no data displayed because SWV is not configured.



System and Thread Viewer

Property: System

Item	Value
Tick Timer:	1.000 mSec
Round Robin Timeout:	5.000 mSec
Default Thread Stack Size:	200
Thread Stack Overflow Check:	Yes
Thread Usage:	Available: 6, Used: 7 + os_idle_demon

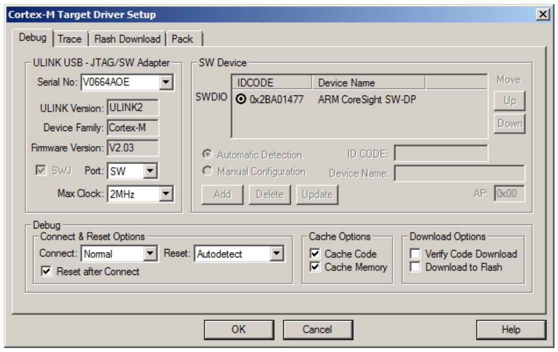
Property: Threads

ID	Name	Priority	State	Delay	Event Value	Event Mask	Stack Usage
1	main	Normal	Wait_DLY				32%
2	phaseA	Normal	Wait_AND				40%
3	phaseB	Normal	Wait_AND		0x0000	0x0001	40%
4	phaseC	Normal	Wait_DLY	366	0x0000	0x0001	44%
5	phaseD	Normal	Wait_AND		0x0000	0x0001	40%
6	clock	Normal	Wait_AND		0x0000	0x0100	40%
255	os_idle_demon	None	Running				

Event Viewer: ***available only with any Keil ULINK or J-Link:***

We must activate Serial Wire Viewer to get the Event Viewer working. These steps are for ULINK2. Others are different.

15. Stop the CPU and exit Debug mode.
16. Connect a ULINK2, ULINK*pro* or J-Link to J8. Remove J101 jumpers RST, TMS, TCK, TDO and TDI. Cycle the power to the board.
17. Select your adapter in the Select Target box. ULINK2
18. Click on the Options for Target next to the Target box.
19. Select the Debug tab and then click the Settings box next to ULINK2/ME Cortex Debugger dialog.
20. In the Debug window as shown here, set Port: to SW.



Cortex-M Target Driver Setup

Debug | Trace | Flash Download | Pack

ULINK USB - JTAG/SW Adapter
Serial No: V0664AOE
ULINK Version: ULINK2
Device Family: Cortex-M
Firmware Version: V2.03
☑ SWJ Port: SW
Max Clock: 2MHz

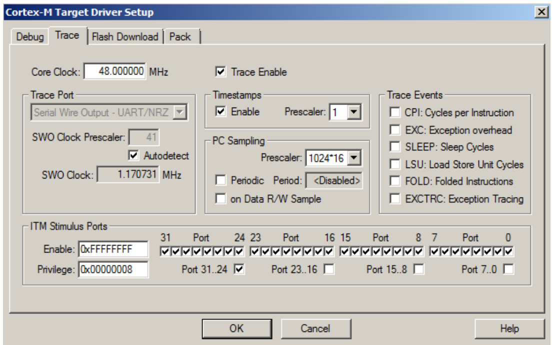
SW Device

	IDCODE	Device Name
SWDIO	● 0x2BA01477	ARM CoreSight SW-DP

● Automatic Detection ID CODE:
○ Manual Configuration Device Name:
Add Delete Update AP: 0x00

Debug
Connect & Reset Options
Connect: Normal Reset: Autodetect
☑ Reset after Connect
Cache Options
☑ Cache Code
☑ Cache Memory
Download Options
☐ Verify Code Download
☐ Download to Flash

21. Click on the Trace tab to open the Trace window.
22. Select Trace Enable. Set Core Clock: to 48 MHz. With a ULINK2 or J-Link, this ***must*** be set to the CPU frequency.
23. Unselect the Periodic and EXCTRC boxes as shown here:
24. ITM Stimulus Port 31 must be checked. This is the port used to output the kernel awareness information to the Event Viewer. ITM is slightly intrusive from calls made by RTX.
25. Click on OK twice to return to the µVision main menu.
The Serial Wire Viewer is now configured in µVision.



Cortex-M Target Driver Setup

Debug | Trace | Flash Download | Pack

Core Clock: 48.000000 MHz ☑ Trace Enable

Trace Port
Serial Wire Output - UART/NRZ
SWO Clock Prescaler: 41
☑ Autodetect
SWO Clock: 1.170731 MHz

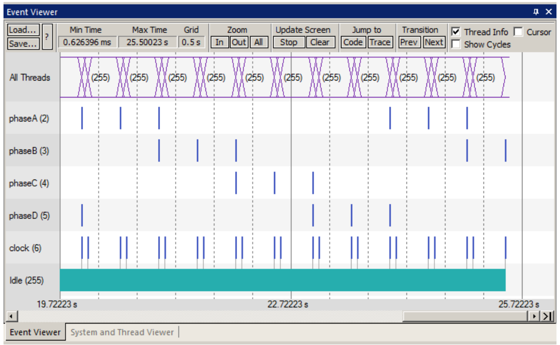
Timestamps
☑ Enable Prescaler: 1

PC Sampling
Prescaler: 1024*16
☐ Periodic Period: <Disabled>
☐ on Data R/W Sample

Trace Events
☐ CPI: Cycles per Instruction
☐ EXC: Exception overhead
☐ SLEEP: Sleep Cycles
☐ LSU: Load Store Unit Cycles
☐ FOLD: Folded Instructions
☐ EXCTRC: Exception Tracing

ITM Stimulus Ports
Enable: 0xFFFFFFFF 31 Port 24 ☑☑☑☑☑☑☑☑ 23 Port 16 ☑☑☑☑☑☑☑☑ 15 Port 8 ☑☑☑☑☑☑☑☑ 7 Port 0 ☑☑☑☑☑☑☑☑
Privilege: 0x00000008 Port 31..24 ☑ Port 23..16 ☐ Port 15..8 ☐ Port 7..0 ☐

26. Click on File/Save All or
27. Enter Debug mode . Click on RUN .
28. Select "System and Threads" tab: note the display is updated.
29. Click on the Event Viewer tab.
30. This window displays the threads in a graphical format as shown in the RTX Kernel window below. You probably have to change the Range to about 0.5 seconds by clicking on the ALL and then the + and – icons.
31. Stop the CPU and exit Debug mode.



Event Viewer

Min Time: 0.626396 ms Max Time: 25.50023 s Grid: 0.5 s
☑ Thread Info ☐ Cursor
☐ Show Cycles

TIP: If Event Viewer doesn't work, open up the Trace Records and confirm there are good ITM 31 frames present. The most probable cause for no valid ITM frames is the Core Clock: is not set correctly

You do not have to stop the program to view this data. Your program runs at nearly full speed. No instrumentation code need be inserted into your source. Event Viewer is a component of RTX. You will find this feature very useful to see if RTX is behaving as you expect it to and to view results of your configuration modifications.

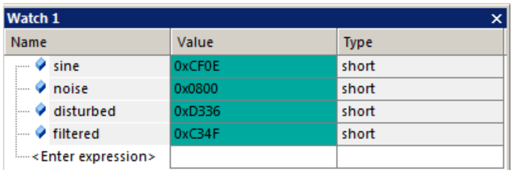
20) DSP SINE example using ARM CMSIS-DSP Libraries and RTX:

ARM CMSIS-DSP libraries are offered for Cortex-M0, Cortex-M0+, Cortex-M3, Cortex-M4 and Cortex-M7 processors. DSP libraries are provided in MDK in C:\Keil_v5\ARM\Pack\ARM\CMSIS\x.x.x\CMSIS\. For documentation see www.keil.com/pack/doc/CMSIS/DSP/html/. For CMSIS 5 see https://github.com/ARM-software/CMSIS_5
This example creates a sine wave to which noise is added, and then the noise is filtered out leaving the original sine wave.

This example incorporates Keil RTX RTOS. RTX is available free with a BSD type license. Source code is provided.

a) Running the DSP Example:

1. Stop the CPU and exit Debug mode. Select Project/Open Project.
2. Open the project file in C:\00MDK\Boards\TI\MSP432\DSP\ sine.uvprojx
3. Select the debugger you are using in the in the Select Target box: This is for the XDS110: XDS
4. For the on-board XDS110-ET, populate all positions of jumper J101.
5. Connect a debugger if used to J8, select it in the Select target box.
6. Remove J101 jumpers RST, TMS, TCK, TDO and TDI. This action disconnects the XDS110 from the processor.
7. Build the files. There will be no errors or warnings.
8. Enter Debug mode by clicking on the Debug icon. The Flash will be programed.
9. Click on the RUN icon.
10. Open Watch 1 by selecting View/Watch Windows/Watch 1.
11. Four global variables will be displayed in Watch 1 as shown here: If these variables are changing, the program is working properly.



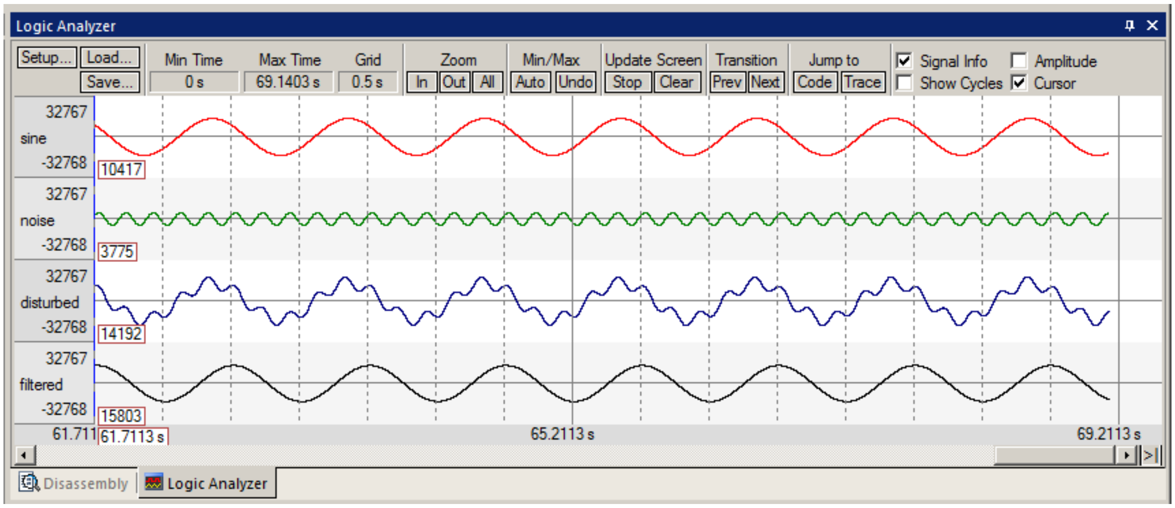
Name	Value	Type
sine	0xCF0E	short
noise	0x0800	short
disturbed	0xD336	short
filtered	0xC34F	short
<Enter expression>		

b) Serial Wire Viewer (SWV) (requires a Keil ULINK2, ULINK-ME, ULINK*pro* or J-Link):

If you are using a ULINK2, ULINK-ME, ULINK*pro* or a J-Link, you will see the screen below. This has been configured by default in this project. You can open this window by selecting View/Analysis Windows/Logic Analyzer or . The Cortex-M3, Cortex-M4 and Cortex-M7 processors have Serial Wire Viewer (SWV).

The values of the global variables shown above will be displayed graphically in the Logic Analyzer as shown below. There are many other SWV features you can use to debug your programs. Currently the TI XDS110 does not support SWV.

TIP: If there are distortions in the waveforms, it is usually because of SWO overload. Solutions are to use a ULINK*pro* or turn off one of the four waveforms in the LA Setup window. Limit the SWV features configured to those you really need.



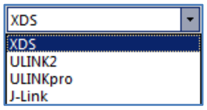
This is the end of the DSP example. Stop the CPU and exit Debug mode.

21) LMP3 Low Power Mode 3 demonstration:

This example puts the MSP432 into LMP3 mode. At this point, it will draw less than 4 uA. Pressing S1 P1.1 creates an exception. The exception handler toggles the red LED1. Refer to the file main.c to examine how this program works.

Load the LoPower software example:

1. Start µVision.
2. Select Project/Open Project. Open the project C:\00MDK\Boards\TI\MSP432\LoPower\LoPower.uvprojx.
3. Select your debug adapter:



Build and Load the Example:

1. Set J101 jumpers according to the debug adapter you are using.
   For the on-board XDS110, populate all positions of jumper J101. For an external adapter such as a ULINK2, remove J101 jumpers RST, TMS, TCK, TDO and TDI. This action disconnects the XDS110 from the processor.

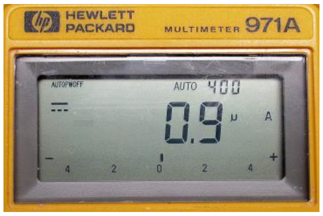
4. Power the board with a USB cable. Build the source files. There will be no errors or warnings.
5. Program the MSP432 Flash by clicking on the Load icon: Progress will be indicated in the lower left corner.
6. The LoPower executable is now permanently loaded (until the next Flash program cycle) into the MSP432 Flash.

Configure the Board:

7. Remove the USB cable attached to the USB connector.
8. Remove the external debug adapter if one is connected. You can debug this program with any adapter but this action activates the ARM CoreSight debug module which will consume more power.
9. Remove all jumpers except for Red LED1 P1.0, 3V3 and GND. This is to minimize power consumption.
10. Connect a quality current meter across the 3V3 jumper pins. Be careful to not short these to the 5V jumpers.

Run the Example:

11. Plug a USB cable to power the board. The LoPower program will run stand-alone without µVision.
12. The MSP432 will go into low power mode with a wfi instruction (wait for interrupt) near line 143 in main.c.
13. Your current meter will read less than 1 uA as seen here:
14. Press S1 P1.1 and the Red LED will come on. Much more power will be consumed because of the LED being on. In my case, it was 1423 uA.
15. Each time you press the button, the red LED is toggled. An interrupt 51 is generated. We will display this event on the next page using Serial Wire Viewer (SWV).



TIP: During the execution of the interrupt handler, normal processor processing power consumption will occur. You may need an oscilloscope to view this power consumption.

What is happening Here: Each time S1 is pressed an interrupt occurs which calls the function PORT1_IRQHandler. This toggles the red LED.

The interrupt is ExtIRQ 35 which is also known as # 51. It is associated with GPIO 1.1

The button S1 is connected to P1.1 and generates ExtIRQ 35 a.k.a # 51.

Note: This value was obtained with one board. The ammeter was not calibrated. Typical results are around 660nA-750nA.

Notes on Using Serial Wire Viewer (SWV): ***See the next page for instructions:***

1. SWV frames are output on the Single Wire Output (SWO) which is a 1 pin output port shared with JTAG signal TDO. This means Serial Wire Debug must be used instead of JTAG. SWD and JTAG are functionally equivalent.
2. ULINK2 and J-Link use the UART mode on the SWO (Serial Wire Output) pin. You must specify the Core Clock: frequency. If you see weird trace frames or none at all, the most usual cause is an incorrect Core Clock: frequency. In the Trace Records window, the only valid ITM frames possible are 31 and 0. All others are bogus.
3. The on-board XDS110 debug adapter does not yet support Serial Wire Viewer SWV.

Display Interrupt 51: A ULINK2, ULINK-ME, ULINK*pro* or J-Link is needed for Serial Wire Viewer support.

Configure the Serial Wire Viewer (SWV):

1. Connect your debug adapter to the board. Remove J101 jumpers RST, TMS, TCK, TDO and TDI.
2. Select your debug adapter in the Select Target box. This setting is for the ULINK2:

ULINK2

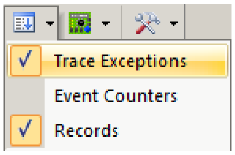
3. Click on Options for Target . Select the Debug tab. Select Settings on the right side:
4. The Target Driver Setup window will open and will display a connection to the CoreSight debug module.
5. Confirm the Port: setting is SW and not JTAG. SWV shares the SWO pin with JTAG TDO resulting in a conflict.
6. Select the Trace tab. Enter 1.5 for Core Clock (1.5 MHz). Select Trace Enable. Select EXCTRC.

TIP: Number one reason SWV doesn't work is the Core Clock: value is incorrectly set. This is the CPU core clock setting.

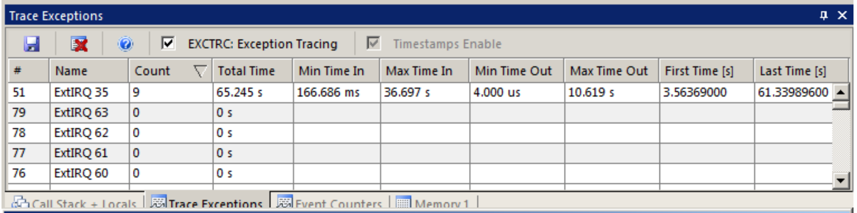
7. Click OK twice to return to the main µVision menu. The SWV data trace is now configured.
8. Click on File/Save All or

Run the Example:

1. Enter Debug mode. Click on the RUN icon.
2. Select the small arrow beside the Trace icon and select Trace Exceptions and Records:



3. In the Trace Exceptions window, click on the Count column or scroll until ExtIRQ 35 (# 51) appears as shown here:

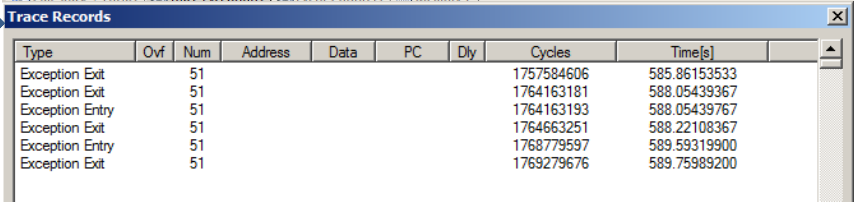


Trace Exceptions

☑ EXCTRC: Exception Tracing ☑ Timestamps Enable

#	Name	Count	Total Time	Min Time In	Max Time In	Min Time Out	Max Time Out	First Time [s]	Last Time [s]
51	ExtIRQ 35	9	65.245 s	166.686 ms	36.697 s	4.000 us	10.619 s	3.56369000	61.33989600
79	ExtIRQ 63	0	0 s						
78	ExtIRQ 62	0	0 s						
77	ExtIRQ 61	0	0 s						
76	ExtIRQ 60	0	0 s						

4. Press S1 P1.1 and the Exception will be recorded. This will also appear in the Trace Records window. It is updated in real-time.
5. The Trace Records window will display the Interrupt 51 as shown:



Trace Records

Type	Ovf	Num	Address	Data	PC	Dly	Cycles	Time[s]
Exception Exit		51					1757584606	585.86153533
Exception Exit		51					1764163181	588.05439367
Exception Entry		51					1764163193	588.05439767
Exception Exit		51					1764663251	588.22108367
Exception Entry		51					1768779597	589.59319900
Exception Exit		51					1769279676	589.75989200

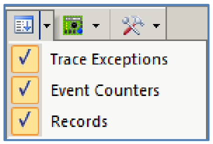
TIP: This is the display for the ULINK2. The ULINK*pro* and J-Link have slightly different trace windows. You must stop the program but the Trace Exceptions updates.

TIP: Stop the program to display the Trace with a ULINK*pro* or J-Link.

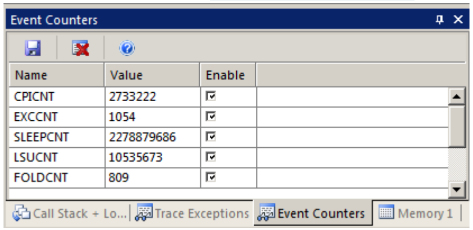
TIP: It is very easy to see when interrupts happen with SWV with Entry and Exit points. Any switch bounce shows as multiple entries.

Event Counters:

1. Select the small arrow beside the Trace icon and select Event Counters:



2. The Event Counters will open as shown below:
3. Enable each counter and they will start incrementing if appropriate: Press the S1 P1.1 button:



Event Counters

Name	Value	Enable
CPICNT	2733222	☑
EXCCNT	1054	☑
SLEEPCNT	2278879686	☑
LSUCNT	10535673	☑
FOLDCNT	809	☑

   1) **CPICNT:** Additional cycles to execute an instruction.
   2) **EXCNT:** Cycles spent entering and exiting exceptions.
   3) **SLEEPCNT:** CPU cycles spent in sleep mode.
   4) **LSUCNT:** Cycles spent waiting for loads and stores.
   5) **FOLDCNT:** Cycles saved with instructions (IT) using no extra cycles.

TIP: Right click in Trace Records and unselect Counter Events to filter these out and allow only interrupts to be displayed.

TIP: For more information see: www.keil.com/support/man/docs/ulinkpro/ulinkpro_trace_event_counter.htm

TIP: These counters are memory mapped and therefore are available to your program. See the CoreSight documentation.

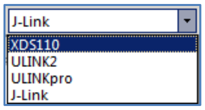
1. **Stop the program and exit Debug mode.**

22) RTX Tickless Low Power Mode RTOS demonstration:

This example uses the tickles mode of RTX RTOS. The idea is to put the processor into a sleep mode when no threads are active. The red LED1 will blink. Since this example does not completely stop like the preceding LoPower project, measuring current draw will require an oscilloscope. Refer to the file main.c to examine how this program works.

Load the RTX_LoPower software example:

1. Start µVision.
2. Select Project/Open Project. Go to C:\00MDK\Boards\TI\MSP432\RTX_LoPower\ and open LoPower.uvprojx.
3. Select your debug adapter: If you are using an external adapter such as a ULINK, attach it to the J8 CoreSight debug connector. This will allow the use of SWV to display the interrupts.



Build and Load the Example:

4. Set J101 jumpers according to the debug adapter you are using.
   For the on-board XDS110, populate all positions of jumper J101. For an external adapter such as a ULINK2, remove J101 jumpers RST, TMS, TCK, TDO and TDI. This action disconnects the XDS110 from the MSP432 processor.
5. Power the board with a USB cable. Build the source files. There will be no errors or warnings.
6. Program the MSP432 Flash by clicking on the Load icon: Progress will be indicated in the lower left corner.
7. The LoPower executable is now permanently loaded (until the next Flash program cycle) into the MSP432 Flash.

Configure the Board:

8. Remove the USB cable attached to the USB connector.
9. Remove an external debug adapter if one is connected. You can debug this program with any adapter but this will activate ARM CoreSight which will then consume more power.
10. Remove all jumpers except for Red LED1 P1.0, 3V3 and GND. This is to minimize power consumption.
11. Connect a quality oscilloscope with a shunt across the 3V3 jumper pins. Be careful to not short these to the 5V.

Run the Example:

12. Plug a USB cable to power the board. The LoPower program will run stand-alone without µVision.
13. The red LED1 will blink at a rate of approximately 2 Hz. When the LED1 is off, the processor is in low power mode.
14. Your current oscilloscope will display on and off currents.

TIP: During the execution of the interrupt handler, normal processor processing power consumption will occur. You may need an oscilloscope to view this power consumption.

15. **Stop the program and exit Debug mode .**

What is happening Here:

Using RTX Event Viewer (EV): EV uses Serial Wire Viewer (SWV): ***See the next page for instructions:***

1. RTX threads are displayed graphically. This is very useful for debugging your RTX project.
2. If you are using a ULINK*pro*, timing values for exceptions are also displayed. You can see the duration of all interrupts handler routines. Their times are easily measured. The next page displays this feature.
3. The on-board TI XDS110 debug adapter does not yet support Serial Wire Viewer SWV.

Display Interrupts and RTX Event Viewer: (any ULINK or J-Link is needed for Serial Wire Viewer support)

Configure the Serial Wire Viewer (SWV):

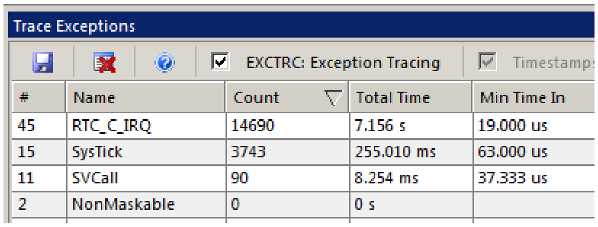
1. Click on the Options icon next to the Target box. Select the Debug tab. Select Settings on the right side:
2. The Target Options window will open and displays a connection to the CoreSight debug module.
3. Confirm the Port: setting is SW and not JTAG. SWV shares the SWO pin with JTAG TDO resulting in a conflict.
4. Select the Trace tab. Enter 3.0 for Core Clock (3 MHz). Select Trace Enable. Select EXCTRC and ITM bit 31.

TIP: Number one reason SWV doesn't work is the Core Clock: value is incorrectly set. This is the CPU core clock setting.

5. Click OK twice to return to the main µVision menu. The SWV data trace is now configured.
6. Click on File/Save All or

Run the Example:

1. Enter Debug mode. Click on the RUN icon.
2. Select the small arrow beside the Trace icon and select Trace Exceptions: You can also select it with View/Trace.
3. In the Trace Exceptions window, click on the Count column until RTC_C_IRQ (# 45) appears as shown here:
4. Also shown are the SysTick and SVCall exceptions..
5. The Trace Records window will also display these events.



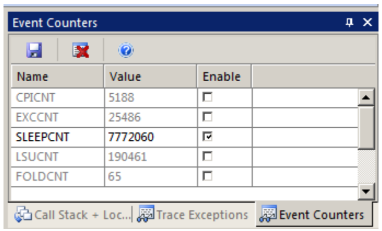
Trace Exceptions

☑ EXCTRC: Exception Tracing ☑ Timestamp

#	Name	Count	Total Time	Min Time In
45	RTC_C_IRQ	14690	7.156 s	19.000 us
15	SysTick	3743	255.010 ms	63.000 us
11	SVCall	90	8.254 ms	37.333 us
2	NonMaskable	0	0 s	

Event Counters:

1. Select the small arrow beside the Trace icon and select Event Counters:
2. The Event Counters will open as shown below:
3. Enable SLEEPCNT in the Enable column. It will start incrementing. SLEEPCNT is the number of CPU cycles spent in sleep mode. See www.keil.com/support/man/docs/ulinkpro/ulinkpro_trace_event_counter.htm



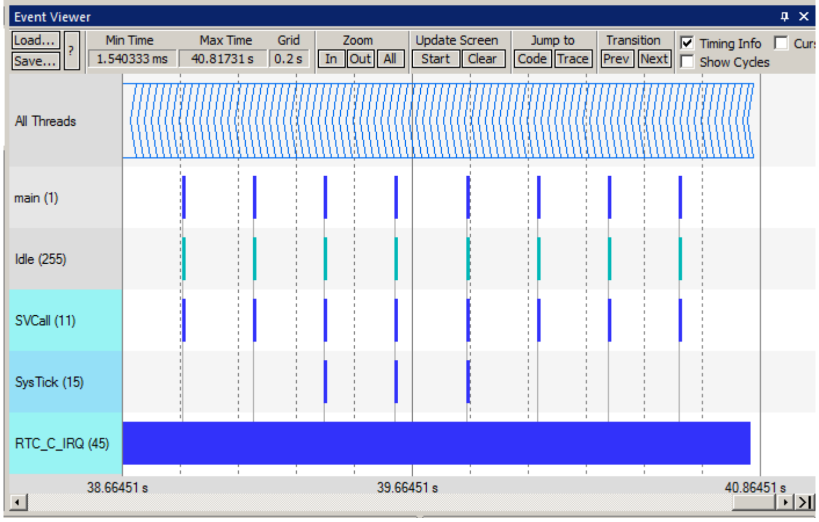
Event Counters

Name	Value	Enable
CPICNT	5188	☐
EXCCNT	25486	☐
SLEEPCNT	7772060	☑
LSUCNT	190461	☐
FOLDCNT	65	☐

Event Viewer: For ULINK2, unselect EXCTRC in Trace Exceptions window.

The Event Viewer offers an easy way to visualize the threads switching. With a ULINK*pro*, interrupt service routines are also displayed.

1. With the program running, select Debug/OS Support/Event Viewer.
2. The window below right opens: adjust Zoom In and Out for a proper window.
3. This example has only two threads: main and idle. You can easily add more.
4. Note the interrupt handlers: SVCall, SysTick and RTC_C_IRQ. This feature is available only with a ULINK*pro*.
5. Select Timing Info and Cursor.
6. Using your mouse, select a starting point and hover over a finish.
7. You can determine the times between a given thread is about 0.25 second.
8. Select Debug/OS Support/System and Thread Viewer.
9. You can see in this window the Threads switching in real-time.
10. Stop the program
11. Exit Debug mode.
12. **This is the end of the examples.**



Event Viewer

Min Time: 1.540333 ms Max Time: 40.81731 s Grid: 0.2 s

☑ Timing Info ☐ Cur ☐ Show Cycles

23) Creating your own MDK 5 project from scratch:

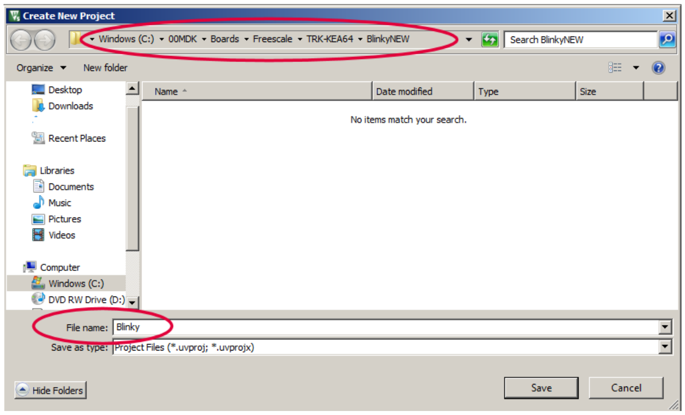
All examples provided by Keil are pre-configured. All you have to do is compile them. You can use them as a starting point for your own projects. However, we will start this example project from the beginning to illustrate how easy this process is. Once you have the new project configured; you can build, load and run a bare Blinky example. It will have a simple incrementing counter to monitor. However, the processor startup sequences are present and you can easily add your own source code and/or files. You can use this process to create any new project, including one using an RTOS such as RTX.

Install the MSP432 Software Pack for your processor:

1. Start µVision and leave it in Edit mode. Do not enter Debug mode. Any project must be loaded to start the process.
2. **Pack Installer:** The MSP432 Software Pack 2.2.0 or later must be installed. This has already been done on page 5.

Create a new Directory and a New Project:

3. In the main µVision menu, select Project/New/ µVision Project… Create New Project opens.
4. In this window, shown here, go to the folder C:\00MDK\Boards\TI\MSP432\
5. Right click in this window and select New and create a new folder. I called it BlinkyNEW.
6. Double click on BlinkyNew to open it or highlight it and select Open.
7. In the File name: box, enter Blinky. Click on Save.
8. This creates the MDK 4 project Blinky.uvproj.
9. The Select Device for Target…opens:


Create New Project

Windows (C:) ▸ 00MDK ▸ Boards ▸ Freescale ▸ TRK-KEA64 ▸ BlinkyNEW

Search BlinkyNEW

No items match your search.

File name: Blinky

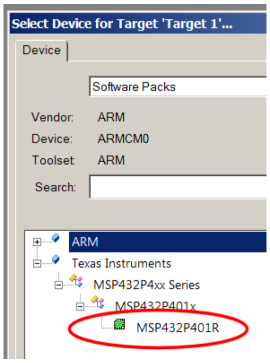
Save as type: Project Files (*.uvproj; *.uvprojx)

Select the Device you are using:

1. Expand Texas Instruments and then select MSP432P401R as shown here:

TIP: Make sure you select the deepest layer device or this will not work correctly.

2. Click OK and the Manage Run Time window shown below bottom right opens.


Select Device for Target 'Target 1'…

Software Packs

Vendor: ARM

Device: ARMCM0

Toolset: ARM

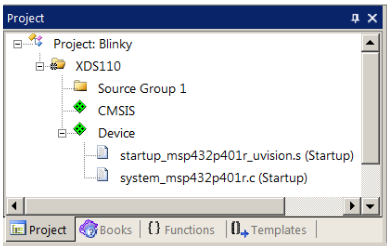
Search:

Select the CMSIS components you want:

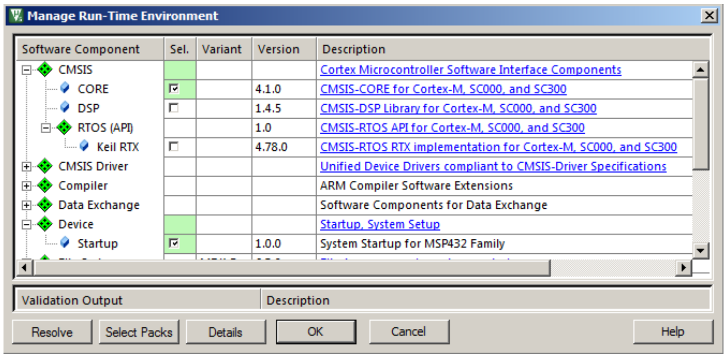
1. Expand CMSIS and Device. Select CORE and Startup as shown below. They will be highlighted in Green indicating there are no other files needed. Click OK to close.
2. Click on File/Save All or select the Save All icon:
3. The project Blinky.uvproj will now be changed to Blinky.uvprojx. (MDK 4 ➡ MDK 5)
4. You now have a new project list as shown on the bottom left below: The CMSIS files you selected have been automatically entered and configured into your project for your selected processor.
5. Note the Target Selector says Target 1. Highlight Target 1 in the Project window.
6. Click once on it and change its name to XDS110 and press Enter. The Select Target name will also change.

What has happened to this point:

You have created a blank µVision project using MDK 5 Software Packs. All you need to do now is add your own source files and select your debug adapter. The Software Pack has pre-configured many settings for your convenience.


Project

- Project: Blinky
  - XDS110
    - Source Group 1
    - CMSIS
    - Device
      - startup_msp432p401r_uvision.s (Startup)
      - system_msp432p401r.c (Startup)

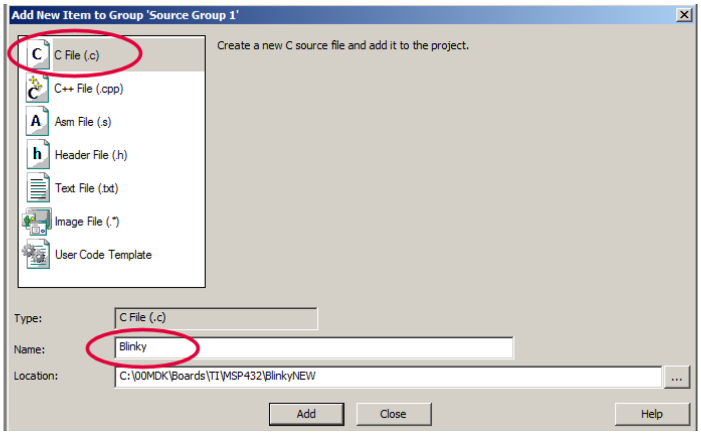

Manage Run-Time Environment

Software Component	Sel.	Variant	Version	Description
CMSIS				Cortex Microcontroller Software Interface Components
CORE	☑		4.1.0	CMSIS-CORE for Cortex-M, SC000, and SC300
DSP	□		1.4.5	CMSIS-DSP Library for Cortex-M, SC000, and SC300
RTOS (API)			1.0	CMSIS-RTOS API for Cortex-M, SC000, and SC300
Keil RTX	□		4.78.0	CMSIS-RTOS RTX implementation for Cortex-M, SC000, and SC300
CMSIS Driver				Unified Device Drivers compliant to CMSIS-Driver Specifications
Compiler				ARM Compiler Software Extensions
Data Exchange				Software Components for Data Exchange
Device				Startup, System Setup
Startup	☑		1.0.0	System Startup for MSP432 Family

Validation Output	Description

Create a blank C Source File:

1. Right click on Source Group 1 in the Project window and select Add New Item to Group 'Source Files'... .
2. This window opens up:
3. Highlight the upper left icon: C file (.c):
4. In the Name: field, enter Blinky.
5. Click on Add to close this window.
6. Click on File/Save All or
7. Expand Source Group 1 in the Project window and Blinky.c will now display. It is blank.



Create a new C source file and add it to the project.

Type: C File (.c)

Name: Blinky

Location: C:\00MDK\Boards\TI\MSP432\BlinkyNEW

Add Some Code to Blinky.c:

1. In the blank Blinky.c, add the C code below:
2. Click on File/Save All or
3. Build the files. There will be no errors or warnings if all was entered correctly.

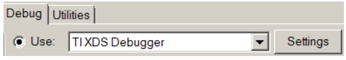
```
#include "msp.h"
unsigned int counter = 0;
/*---------------------------------------------
  MAIN function
 *---------------------------------------------*/
int main (void) {

  while(1) {
      counter++;
          if (counter > 0x0F) counter = 0;
}
}          //make sure you add a CR Carriage Return or Enter after the last parentheses.
```

TIP: You can also add existing source files: Add Existing Files to Group 'Source Files'...

Configure the Target XDS110: ***Please complete these instructions carefully to prevent unusual problems…***

1. Select the Target Options icon . Select the **Target** tab. Note the Flash and RAM addresses are already entered.
2. Select Use MicroLIB to optimize for smaller code size. An error will be generated if you cannot use this feature.
3. Select the **Linker** tab. Select Use Memory Layout from Target Dialog.
4. Click on the **Debug** tab. Select the debugger you are using in the Use: box:

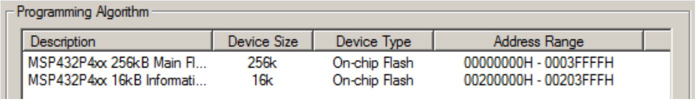


● Use: TI XDS Debugger

5. Connect your LaunchPad to your PC USB port. If using an external adapter, connect it now. Set the J101 jumpers.

TIP: You can select the debug adapter you are using at this point such as a ULINK2, ULINK*pro* or a J-Link. If Flash programming fails, you can try reducing Max Clock: to 1 MHz or so.

6. Select Settings: box on the right side as shown above.
7. Set SW as shown here: Port SW If your LaunchPad board is connected to your PC, you should now see a valid IDCODE and Device Name in the SW Device box. If you do not, you **must** correct this before continuing.
8. Click on the **Flash Download** tab. Confirm the correct Flash algorithms are present: Shown here are the correct ones for the MSP432 board:
9. Click on OK twice to return to the main menu.
10. Click on File/Save All or
11. Build the files. There will be no errors or warnings if all was entered correctly. If there are, please fix them !



Programming Algorithm

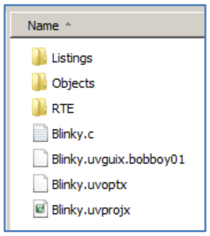
Description	Device Size	Device Type	Address Range
MSP432P4xx 256kB Main Fl...	256k	On-chip Flash	00000000H - 0003FFFFH
MSP432P4xx 16kB Informati...	16k	On-chip Flash	00200000H - 00203FFFH

The Next Step ? First we will do a summary of what we have done so far and then ….

Let us run your program and see what happens ! Please turn the page….

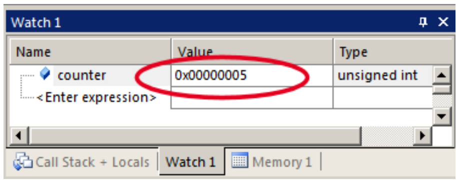
What we have so far ?

1. An MDK 5 project (Blinky) has been created in C:\00MDK\Boards\TI\MSP432\BlinkyNEW\
2. The folders have been created similar as shown here:
3. RTE contains the CMSIS-Core startup and system files.
4. The Software Pack has pre-configured many items in this new project for your convenience.



Running Your Program:

1. Enter Debug mode. The Flash will be programmed. A progress bar is displayed in the Command window.
2. Click on the RUN icon. Note: you stop the program with the STOP icon.
3. Right click on counter in Blinky.c and select Add 'counter' to … and select Watch 1.
4. counter should be updating as shown here:
5. You can also set a breakpoint in Blinky.c and the program should stop at this point if it is running properly. If you do this, remove the breakpoint.
6. You should now be able to add your own source code to create a meaningful project.



Name	Value	Type
counter	0x00000005	unsigned int
<Enter expression>		

TIP: Watch 1 is updated periodically, not when a variable value changes. Since Blinky is running very fast without any time delays inserted, the values in Watch 1 will appear skip some sequential values that you know must exist.

Configuring the CPU Clock:

The file system_msp432p401r.c contains CPU clock configuration code to change the clock from the default of 3.0 MHz to 48 MHz. We will run two functions in system_msp432p401r.c to initialize the system and configure the clock.

1. Stop the CPU. and exit Debug mode.
2. In the Project window, double click on system_msp432p401r.c to open it.
3. In system_msp432p401r .c near line 71 enter the new clock frequency: `#define __SYSTEM_CLOCK 48000000`
4. Near line 77, set variable __REGULATOR to 1.
7. In Blinky.c, near line 7, just after int main(void) {, add these lines:

```
7   SystemCoreClockUpdate();
```

8. Click on File/Save All or
9. Build the files. There will be no errors or warnings. Enter Debug mode. The Flash will be programmed.
10. In Watch 1, double click on <Enter expression> and enter SystemCoreClock and press Enter.
11. Right click on SystemCoreClock in Watch1 and unselect Hexadecimal Display.
12. The CPU speed is now 48 MHz as shown in the global variable SystemCoreClock in Watch 1.
13. Click on the RUN icon. It is difficult to notice that the processor is running faster with this simple program.
14. Stop the CPU. and exit Debug mode.

What else can we do ?

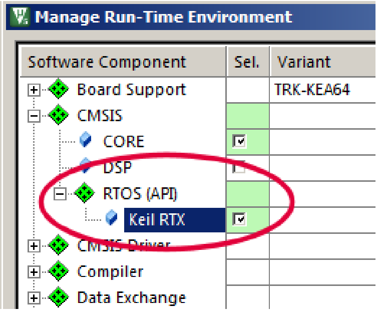
5. You can create new source files using the Add New Item window. See the top of the previous page.
6. You can add existing source files by right clicking on a Group name and selecting Add Existing Files.
7. You can easily add TI MSP432 example files to your project.
8. If you use RTX or Keil Middleware, source and template files are provided in the Add New window.

24) Adding RTX to your MDK 5 project:

The MDK Software Packs contain the code needed to add RTX to your project. RTX is CMSIS-RTOS compliant.

Configuring RTX is easy in MDK 5. These steps use the preceding Blinky example you constructed.

1. Using the same example from the preceding pages, Stop the program and Exit Debug mode.
2. Open the Manage Run-Time Environment window:
3. Expand all the elements as shown here:
4. Select Keil RTX as shown and click OK.
5. Appropriate RTX files will be added to your project. See the Project window.
6. In Blinky.c, on the first line, right click and select Insert '# include file'. Select "cmsis_os.h". This will be added as the first line in Blinky.c.

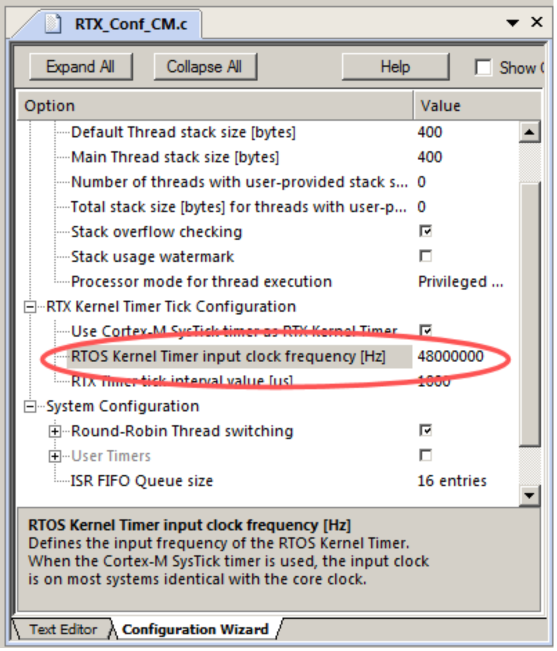


Manage Run-Time Environment

Software Component	Sel.	Variant
⊞ Board Support		TRK-KEA64
⊟ CMSIS		
CORE	☑	
DSP	□	
⊟ RTOS (API)		
Keil RTX	☑	
⊞ CMSIS Driver		
⊞ Compiler		
⊞ Data Exchange		

Configure RTX:

1. In the Project window, expand the CMSIS group.
2. Double click on RTX_Conf_CM.c to open it.
3. Select the Configuration Wizard tab at the bottom of this window: Select Expand All.
4. The window is displayed here:
5. Select Use Cortex-M SysTick Timer as RTX Kernel Timer.
6. Set Timer clock value: to 48000000 as shown: (48 MHz)
7. Use the defaults for the other settings.



Option	Value
Default Thread stack size [bytes]	400
Main Thread stack size [bytes]	400
Number of threads with user-provided stack s...	0
Total stack size [bytes] for threads with user-p...	0
Stack overflow checking	☑
Stack usage watermark	□
Processor mode for thread execution	Privileged ...
⊟ RTX Kernel Timer Tick Configuration	
Use Cortex-M SysTick timer as RTX Kernel Timer	☑
RTOS Kernel Timer input clock frequency [Hz]	48000000
RTX Timer tick interval value [us]	1000
⊟ System Configuration	
⊞ Round-Robin Thread switching	☑
⊞ User Timers	□
ISR FIFO Queue size	16 entries

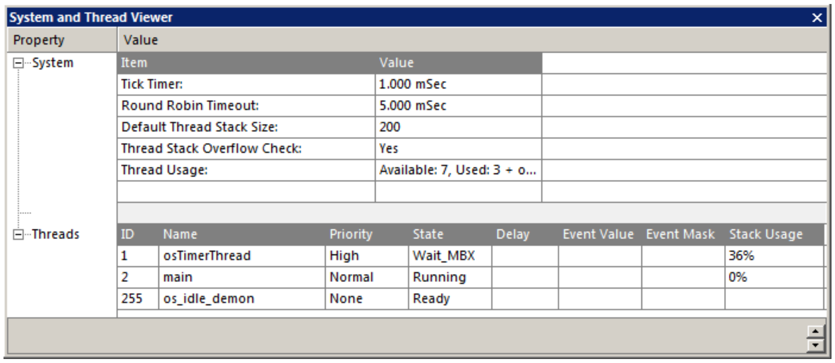
RTOS Kernel Timer input clock frequency [Hz]
Defines the input frequency of the RTOS Kernel Timer.
When the Cortex-M SysTick timer is used, the input clock
is on most systems identical with the core clock.

Build and Run Your RTX Program:

1. Click on File/Save All or
2. Build the files. There will be no errors or warnings.
3. Enter Debug mode: Click on the RUN icon.
4. Select Debug/OS Support/System and Thread Viewer. The window below opens up.
5. You can see three threads: the main thread is the only one running. As you add more threads to create a real RTX program, these will automatically be added to this window.
6. Stop the program and Exit Debug mode.

What you have to do now:

1. You must add the RTX framework into your code and create your threads to make this into a real RTX project.
2. **Getting Started MDK 5:** Obtain this useful book here: www.keil.com/gsg/. It has very useful information on implementing RTX.
3. You can use the Event Viewer to examine your threads graphically if you have a ULINK2, ULINK*pro* or a J-Link. Event Viewer uses SWV which is not yet supported by XDS110.



System and Thread Viewer

Property	Value	
⊟ System	Item	Value
	Tick Timer:	1.000 mSec
	Round Robin Timeout:	5.000 mSec
	Default Thread Stack Size:	200
	Thread Stack Overflow Check:	Yes
	Thread Usage:	Available: 7, Used: 3 + o...

ID	Name	Priority	State	Delay	Event Value	Event Mask	Stack Usage
1	osTimerThread	High	Wait_MBX				36%
2	main	Normal	Running				0%
255	os_idle_demon	None	Ready				

TIP: The Configuration Wizard is a scripting language as shown in the Text Editor as comments starting such as a </h> or <i>. See www.keil.com/support/docs/2735.htm for instructions on how to add this feature to your own source code.

25) Adding a Thread to your RTX_Blinky:

We will create and activate a thread. We will add an additional variable counter2 that will be incremented by this new thread.

1. In Blinky.c, add this line near line 4: unsigned int counter2 = 0;

```
4   unsigned int counter2 = 0;
```

Create the Thread job1:

2. Add this code before main():
   This will be the new thread named job1.

   osDelay(500) delays the program by 500 clock ticks to slow it down so we can see the values of counter and counter2 increment by 1.

```
6     void job1 (void  const *argument) {
7     for (;;) {
8               counter2++;
9                 if (counter2 > 0x0F) counter2=0;
10               osDelay(500);
11      }
12   }
```

Add osDelay to main():

3. Add this line just after the if statement near line 20:

```
20   osDelay(500);
```

Define and Create the Thread:

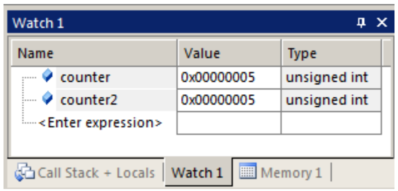
1. Add this line near line 15 just before main():

```
15   osThreadDef(job1, osPriorityNormal, 1, 0);
```

2. Create the thread job1 near line 19 just before the while(1) loop:

```
19   osThreadCreate(osThread(job1), NULL);
```

3. Click on File/Save All or
4. Build the files. There will be no errors or warnings. If there are, please fix them before continuing.
5. Enter Debug mode: Click on the RUN icon.
6. Right click on counter2 in Blinky.c and select Add counter2 to … and select Watch 1.
7. Both counter and counter2 will increment but slower than before:
   The two osDelay(500) function calls each slow the program down by 500 msec. This makes it easier to watch these two global variables increment.



Watch 1

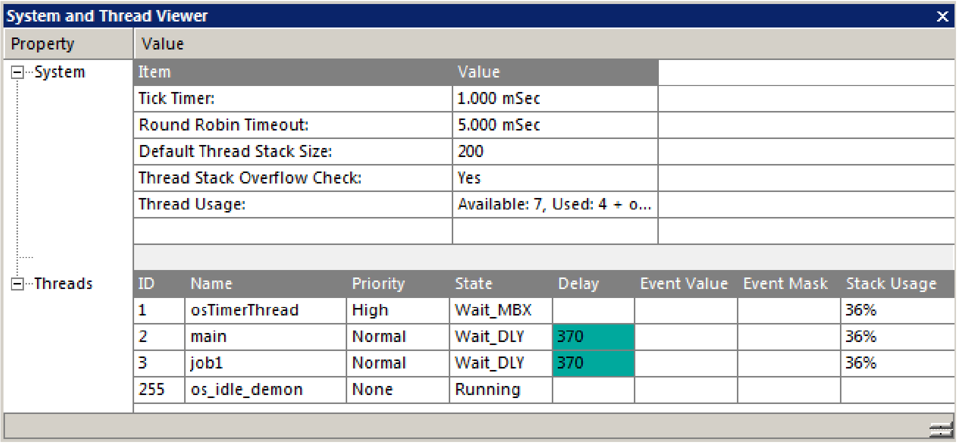
Name	Value	Type
counter	0x00000005	unsigned int
counter2	0x00000005	unsigned int
<Enter expression>		

TIP: osDelay() is a function provided by RTX and is triggered by the SysTick timer.

8. Open the System and Thread Viewer by selecting Debug/OS Support.
9. Note that job1 has now been added as a thread as shown below:
10. Note os_idle_demon is nearly always labelled as Running. This is because the program spends most of its time here.
11. Set a breakpoint in job1 and the program will stop there and job1 is displayed as "Running" in the Viewer.
12. Set another breakpoint in the while(1) loop in main() and each time you click RUN, the program will change threads.
13. There are many attributes of RTX you can add. RTX help files are located here depending on your CMSIS version: C:/Keil_v5/ARM/Pack/ARM/CMSIS/*x.x.x*/CMSIS/Documentation/RTX/html/index.html.
14. **Remove any breakpoints you have created.**

On the next page, we will demonstrate the Event Viewer. For this you must use a Keil ULINK2, ULINK*pro* or a Segger J-Link.

.



System and Thread Viewer

Item	Value
Tick Timer:	1.000 mSec
Round Robin Timeout:	5.000 mSec
Default Thread Stack Size:	200
Thread Stack Overflow Check:	Yes
Thread Usage:	Available: 7, Used: 4 + o...

ID	Name	Priority	State	Delay	Event Value	Event Mask	Stack Usage
1	osTimerThread	High	Wait_MBX				36%
2	main	Normal	Wait_DLY	370			36%
3	job1	Normal	Wait_DLY	370			36%
255	os_idle_demon	None	Running				

26) Demonstrating the Event Viewer (EV): Requires a Keil ULINK2/ME, ULINK*pro* or a J-Link.

The Event Viewer displays threads running in a graphical format. It is possible to make timing measurements.

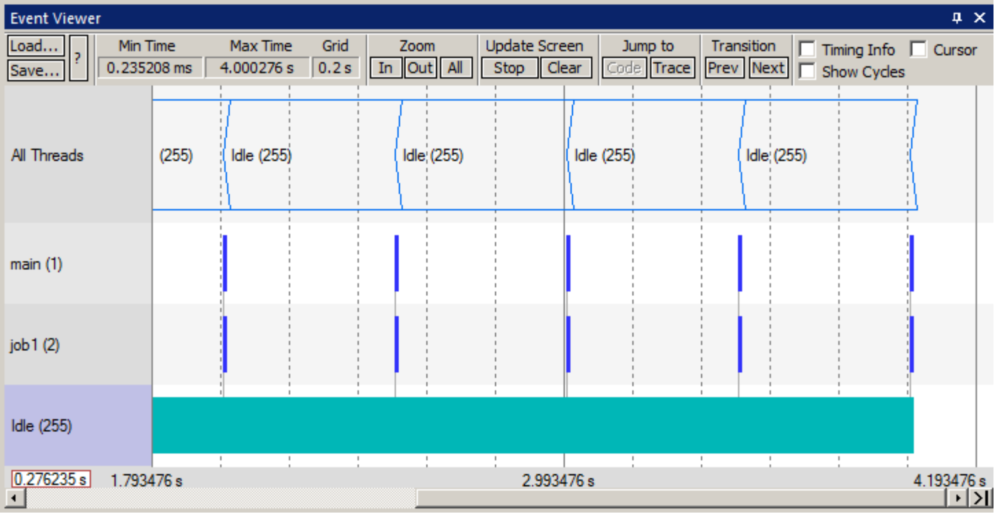
The Event Viewer uses Serial Wire Viewer (SWV). The XDS110 does not support SWV. In order to do the exercise on this page, you must use any Keil ULINK or a J-Link. A ULINK*pro* is best as it handles SWV data the fastest.

Connect and Configure SWV:

1. Connect a ULINK2 or ULINK*pro* or J-Link to J8.
2. Remove J101 jumpers RST, TMS, TCK, TDO and TDI. This action disconnects the XDS110 from the processor.
3. Connect a USB cable to the LaunchPad to supply power to the board.
4. Select the Target Options icon . Select the Debug tab. Select the debugger you are using:
5. Click the Debug tab. Select the debugger you are using in the Use: box: This is for ULINK2: ULINK2/ME Cortex Debugger
6. Click on Settings on the right side of the target Options window. The Cortex-M Target Driver Setup window opens.
7. Set SW as shown here: Port SW JTAG does not support SWV. If your LaunchPad board is connected to your PC, you should now see a valid IDCODE and Device Name in the SW Device box. If you do not, you **must** correct this before continuing.
8. Set Max. Clock: to 2 MHz. If this value is too high, Flash programming might not work reliably resulting in an error.
9. Select the Trace tab. Select Core Clock: to 48 (48 MHz). Select Trace Enable.
10. Unselect EXCTRC to minimize SWO overflows. ITM bit 31 must be selected as Event Viewer uses this port.
11. Click OK twice to close the Target configuration windows.
12. Click on File/Save All or SWV is now configured.

Run Blinky and Open Event Viewer:

1. Enter Debug mode: Click on the RUN icon.
2. The program should be running as shown in the System and Thread Viewer as shown on the previous page. Viewing this window gives a good visual indication RTX is configured and running properly. EV displays timing graphically.
3. Select Debug/OS Support and select Event Viewer.
4. This window will open and if SWV is correctly configured, the threads will be visible.
5. Use the In and/or Out buttons to select an appropriate horizontal scale.



Event Viewer

Min Time 0.235208 ms

Max Time 4.000276 s

Grid 0.2 s

□ Timing Info □ Cursor

□ Show Cycles

6. Note the program spends most of its time in Idle (os demon). This can be modified in your program.

This is the end of the examples for this tutorial. Thank you !

27) Serial Wire Viewer Summary:

Serial Wire Viewer can see:

- Global variables.
- Static variables.
- Structures.
- Peripheral registers – just read or write to them.
- Can't see local variables. (just make them global or static).
- Can't see DMA transfers – DMA bypasses CPU and SWV by definition.

Serial Wire Viewer displays in various ways:

- PC Samples.
- Data reads and writes: in trace or Logic Analyzer windows.
- Exception and interrupt events.
- CPU counters.
- Timestamps.
- RTOS Event Viewer.

TroubleShooter:

Here are some issues that arose during the creating and testing of this document.

1. Something doesn't work as claimed: usually a step was inadvertently missed. This is very common. Go slower.
2. No XDS110 found: Make sure µVision can see C:\ti\. This contains the XDS110 driver. Make sure the firmware in the Launchpad board is current. Try reloading it. Check to see if there is a newer version.
3. No Serial Wire Viewer: Nearly always the Core Clock: is wrong. This must be set to the exact CPU frequency with a ULINK2 or J-Link. With a ULINKpro this will result in erroneous timestamps displayed. Check the global variable SystemCoreClock in a Watch window. In this tutorial it is nearly always 48 MHz. 1.5 for the LoPower example.
4. No SWV or can't connect: Sometimes the debug adapter does not completely RESET the CoreSight components by design. Cycle the board power. This is more common when switching between debug adapters.
5. Serial Wire Viewer erratic or Logic Analyzer or Event Viewer distorted: This usually means you have selected too many attributes and SWO pin is overloaded. Unselect those you do not need or use a ULINK*pro*. If problem is in the Logic Analyzer, you might have to poll your variable in your program.
6. SWV or connection troubles: Cycle the board power. This ensures a clean restart in case something was set inside the processor. Especially when switching between debug adapters.
7. No ULINK found: If a ULINK is connected: this means you have selected the wrong ULINK in Target Options.
8. No SWD Connection: Your ULINK or J-Link is connected backwards to the 5 pin CoreSight connector or the Debug switch is in the wrong position.
9. Flash Programming failed: This means your compilation failed because of an error that will be listed. The last .axf file will be deleted and no new one created therefore there is no executable .axf file to download. It can also mean the Max Clock in the Debug tab in the Target Options is set too high. For ULINK2 try 2 MHz.
10. Program halts unexpectedly: Perhaps you failed to delete breakpoints or Watchpoints after using them ?

28) Document Resources: See www.keil.com/TI

Books:

1. ***NEW!*** **Getting Started with MDK 5:** Obtain this free book here: www.keil.com/gsg/
2. There is a good selection of books available on ARM: www.arm.com/support/resources/arm-books/index.php
3. µVision contains a window titled Books. Many documents including data sheets are located there.
4. **A list of resources is located at:** www.arm.com/products/processors/cortex-m/index.php
   Click on the Resources tab. Or select the Cortex-M processor you want in the Processor panel on the left.
5. Or search for the Cortex-M processor you want on www.arm.com.
6. **The Definitive Guide to the ARM Cortex-M0/M0+** by Joseph Yiu. Search the web for retailers.
7. **The Definitive Guide to the ARM Cortex-M3/M4** by Joseph Yiu. Search the web for retailers.
8. **Embedded Systems: Introduction to Arm Cortex-M Microcontrollers** (3 volumes) by Jonathan Valvano
9. **Introduction to the MSP432 Microcontroller** by Jonathan Valvano
10. **Real-Time Interfacing to the MSP432 Microcontroller** by Jonathan Valvano
11. **MOOC:** Massive Open Online Class: University of Texas: http://users.ece.utexas.edu/~valvano/

Application Notes:

12. ***NEW!*** ARM Compiler Qualification Kit: Compiler Safety Certification: www.keil.com/safety
13. Using Cortex-M3 and Cortex-M4 Fault Exceptions www.keil.com/appnotes/files/apnt209.pdf
14. CAN Primer: Controller Area Network: www.keil.com/appnotes/files/apnt_247.pdf
15. Segger emWin GUIBuilder with µVision™ www.keil.com/appnotes/files/apnt_234.pdf
16. Porting mbed Project to Keil MDK™ 4 www.keil.com/appnotes/docs/apnt_207.asp
17. MDK-ARM™ Compiler Optimizations www.keil.com/appnotes/docs/apnt_202.asp
18. GNU tools (GCC) for use with µVision https://launchpad.net/gcc-arm-embedded
19. RTX CMSIS-RTOS RTX Documents www.keil.com/pack/doc/CMSIS/RTX/html
20. Barrier Instructions http://infocenter.arm.com/help/topic/com.arm.doc.dai0321a/index.html
21. Lazy Stacking on the Cortex-M4: www.arm.com and search for DAI0298A
22. Cortex Debug Connectors: www.keil.com/coresight/coresight-connectors
23. Sending ITM printf to external Windows applications: http://www.keil.com/appnotes/docs/apnt_240.asp
24. ***New*** Using from Cortex-M3 and Cortex-M4 Fault Exceptions www.keil.com/appnotes/docs/apnt_209.asp
25. Sending ITM printf to external Windows applications: www.keil.com/appnotes/docs/apnt_240.asp
26. Using Cortex-M3 and Cortex-M4 Fault Exceptions www.keil.com/appnotes/docs/apnt_209.asp
27. ***New*** Migrating from Cortex-M3/M4 to Cortex-M7 processors: www.keil.com/appnotes/docs/apnt_270.asp
28. ***New*** ARMv8-M Architecture Technical Overview https://community.arm.com/docs/DOC-10896
29. ***NEW!*** Determining Cortex-M CPU Frequency using SWV www.keil.com/appnotes/docs/apnt_297.asp

Useful ARM Websites:

1. CMSIS Standards: https://github.com/ARM-software/CMSIS_5 and www.arm.com/cmsis/
2. CMSIS Documentation: www.keil.com/pack/doc/CMSIS/General/html
3. ARM and Keil Community Forums: www.keil.com/forum and http://community.arm.com/groups/tools/content
4. ARM University Program**:** www.arm.com/university. Email: university@arm.com
5. mbed™: http://mbed.org

29) Keil Products and Contact Information:

Keil Microcontroller Development Kit (MDK-ARM™)

- **MDK-Lite™** (Evaluation version) 32K Code and Data Limit - $0
- *New* **MDK-ARM-CM™** For all Cortex-M series processors only – unlimited code limit.
- *New* **MDK-Plus™** MiddleWare Level 1. ARM7™, ARM9™, Cortex-M, SecureCore®.
- *New* **MDK-Professional™** MiddleWare Level 2. For complete details: www.keil.com/mdk5/version520.

Middleware includes Network, USB, Graphics and File System. www.keil.com/mdk5/middleware/

USB-JTAG adapters (for Flash programming too)

- **ULINK2** - *(ULINK2 and ME - SWV only – no ETM)*
- **ULINK-ME** – sold only with a board by Keil or OEM. Electrically equivalent to a ULINK2.
- *New* **ULINK*plus***- Cortex-Mx High performance SWV & power measurement.
- **ULINK*pro*** - SWV & ETM trace. Fast Flash programming speed. Also works with DS-5.
- **ULINK*pro* D** - SWV & no ETM. Fast Flash programming speed. Also works with DS-5.

- ***For special promotional or quantity pricing and offers, please contact Keil Sales.***
- ***In USA, Canada and South America:*** **1-800-348-8051** ***sales.us@keil.com***
- ***Europe, Asia, Middle East, Africa:*** **+49 89/456040-20** ***sales.intl@keil.com***

Keil RTX RTOS is now provided as a component of CMSIS 5: ***New*** https://github.com/ARM-software/CMSIS_5

All versions, including MDK-Lite, include Keil RTX RTOS *with source code !*

Keil provides free DSP libraries for the Cortex-M0, Cortex-M3, Cortex-M4 and Cortex-M7.

KEIL Tools by ARM
Development Tools
Getting Started
Creating Applications with µVision®4
For 8-bit, 16-bit, and 32-bit Microcontrollers
ARM
Cx51
C166

Call Keil Sales for details on current pricing, specials and quantity discounts. Sales can also provide advice about the various tools options available to you. They will help you find various labs and appnotes that are quite useful.

All products include Technical Support for 1 year. This is easily renewed.

Call Keil Sales for special university pricing.

See www.keil.com/TI for more information regarding TI support.

For Linux, Android, various RTOS and no OS support (bare metal) on TI Cortex-A processors, please see DS-5: www.arm.com/ds5.

For more information:

Sales In Americas: sales.us@keil.com or 800-348-8051. **Europe/Asia:** sales.intl@keil.com +49 89/456040-20

Global Inside Sales Contact Point: Inside-Sales@arm.com **Keil World Wide Distributors:** www.keil.com/distis/

Keil Technical Support in USA: support.us@keil.com or 800-348-8051. Outside the US: support.intl@keil.com.

For comments or corrections please email bob.boys@arm.com

For the latest version of this document and for more Texas Instrument specific information: See www.keil.com/TI

CMSIS: www.arm.com/cmsis **CMSIS Version 5:** https://github.com/ARM-software/CMSIS_5