NXP i.MXRT1060 Arm® Cortex®-M7
CrossOver Tutorial using the MIXMRT1060-EVK
using ARM® Keil® MDK™ toolkit *featuring Serial Wire Viewer*

Version 1.5   Summer 2020   Robert Boys

The latest version of this document is here: www.keil.com/appnotes/docs/apnt 332.asp

Introduction:

The purpose of this lab is to introduce you to the NXP Cortex™-M7 processor family using the ARM Keil MDK toolkit featuring the µVision® IDE. At the end of this tutorial, you will work confidently with NXP processors and Keil MDK™.

Keil MDK has an evaluation version that limits code and data size to 32 Kbytes. Most Keil examples are under this 32K. The addition of a license number will turn it into a full, unrestricted version. Contact Keil sales for a temporary full version license to evaluate MDK with larger programs or Keil Middleware. MDK includes a full version of Keil RTX™ RTOS with all source code included.   See www.keil.com/NXP for information concerning Keil support of NXP products.

Why Use Keil MDK ?

MDK provides these features particularly suited for NXP Cortex-M0, M0+, M3, M4, M7, M23 and Cortex-M33 processor users:

1. µVision IDE with Integrated Debugger, Flash programmer and the ARM® Compiler toolchain. MDK is turn-key "out-of-the-box".

2. NXP MCUXpresso SDK and MCUXpresso Config support MDK.

3. ARM Compiler 5 and Arm Compiler 6 (LLVM) are included. GCC is also supported: https://developer.arm.com/open-source/gnu-toolchain/gnu-rm

4. Dynamic Syntax checking on C/C++ source lines.

5. **Keil Middleware:** Network, USB, Flash File and Graphics.

6. CMSIS-RTOS RTX is included. www.keil.com/RTX.

7. *NEW!* **Keil Event Recorder.** Annotate your source code.

8. MISRA C/C++ support using PC-Lint. www.gimpel.com

9. **Compiler Safety Certification Kit:** www.keil.com/safety/   *NEW !* **FuSa RTS** www.keil.com/fusa-rts

10. **TÜV** certified. ISO 26262 ASIL D, IEC62304 Class C, IEC 61508 SIL 3, EN50128 SIL 4

11. CoreSight™ Serial Wire Viewer and ETM trace capability on appropriately equipped NXP processors.

12. Choice of adapters: LPC-Link2 (CMSIS-DAP), ULINK™2, ULINK-ME, ULINK*pro,* ULINK*plus* and J-Link.

13. Keil Technical Support is included for one year and is renewable. This helps you get your project completed faster.

14. Affordable perpetual and term licensing. Contact Keil sales for pricing, options and current special offers.
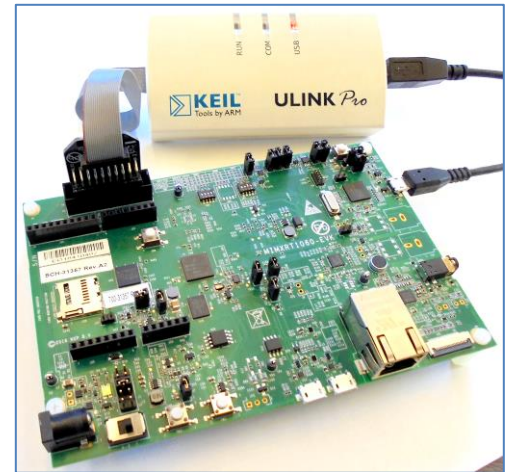
Significant features shown in this document:

1. A Blinky example showing basic operation of µVision and its debugging capabilities.

2. Serial Wire Viewer (SWV) with the Keil ULINK2, ULINK-ME, ULINK*pro*, ULINK*plus* and J-Link.

3. Real-time Read and Write memory accesses for Watch, Memory, Peripheral and RTX Tasks windows. These are non-intrusive to your program. No CPU cycles are stolen. No instrumentation code is added to your source files.

4. Keil Event Recorder. Functions with all debug adapters.

5. Power Measurement with ULINK*plus*

Serial Wire Viewer (SWV): *Use a ULINK2, ULINKpro, ULINKplus or a J-Link for this feature.*

Serial Wire Viewer (SWV) displays PC Samples, Exceptions (including interrupts), data reads and writes, printf (ITM), CPU counters and a timestamp. SWV is non-intrusive and no code stubs are needed in your source code.

Keil Event Recorder (EVR):

Event Recorder (EVR) is a system to annotate your source code and display messages and data in various windows. If used with a Keil ULINK*plus*, Power Measurement and CPU speed are added. www.keil.com/mdk5/debug/eventrecorder

---

1

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com

1) NXP Arm Cortex Processor Evaluation Boards:

This tutorial is for the NXP MIMXRT1060 evaluation board. You can adapt this for other members of the i.MX Cortex-M7 family. Keil supports and makes boards with other NXP processors. See www.keil.com/NXP for more information.

All NXP Arm processors implement Serial Wire Viewer (SWV) except Cortex-M0 and M0+, which have neither SWV nor ETM. SWV is data trace. ETM is instruction trace and is not covered in this document.

The Cortex-M0+ (LPC800, LPC81x, LPC82x and LPC83x) has MTB Micro Trace Buffer. All have non-intrusive read/write to memory locations (for Watch, Memory and Peripheral windows), hardware breakpoints and Watchpoints. Many (but not all) NXP Cortex-M3, M4, M7 and M33 have ETM instruction trace which also provides program flow debugging, Code Coverage and Performance Analysis. i.MXRT1060 does have ETM trace.

NXP MIMXRT1060-EVK Evaluation Board:
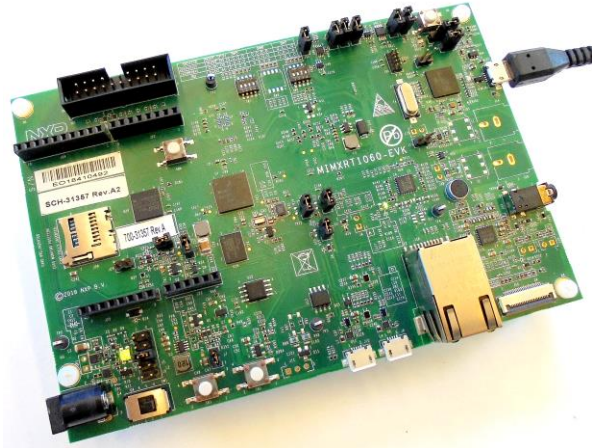
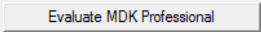# 2) *Summary:* Four Steps to Get Connected and Configured:

1.   Obtain and install Keil MDK Core (evaluation version) on your PC. Use the default directory C:\ Keil_v5\.
2.   Download the Software Packs: i.MX RT1062 DFP and i.MX1060 BSP and examples for your board.
3.   Power the board with USB to J41 USB connector. The on-board LPC-Link2 (U38) will be the debug adapter.
4.   Build the project in μVision and load it into the i.MX RT1060 board Flash memory and RUN it.

These steps are detailed in the next few pages.

3) Keil MDK Core Software Download and Installation: This tutorial used MDK 5.31.

1.   Download MDK Core from the Keil website. www.keil.com/mdk5/install   ± Download MDK-Core
2.   Install MDK into the default folder. You can install into any folder or drive, but this lab uses the default C:\Keil_v5\
3.   We recommend you use the default folders for this tutorial. We will use C:\00MDK\iMXRT1060\ for the examples.
4.   If you install MDK or the examples into different folders, you will have to adjust for the folder location differences.

Licensing:

1.   No license is needed for MDK for code size less than 32K including the examples in this tutorial using MDK 5.31.
2.   You can obtain a one-time free 7 day MDK Pro license in File/License Management. If you are eligible, this button is visible: Evaluate MDK Professional
3.   Or go to: www.keil.com/support/man/docs/license/license_eval.htm
4.   This gives you unlimited code size compilation and access to Keil Middleware.
5.   Contact Keil Sales to extend this license for evaluation purposes. Inside-Sales@arm.com

CMSIS-Pack documentation: www.keil.com/pack/doc/CMSIS/Pack/html/

Complete CMSIS documentation: www.keil.com/CMSIS/

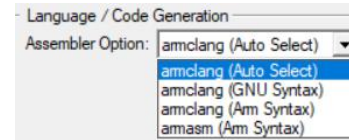CMSIS 5 on GitHub: https://github.com/ARM-software/CMSIS_5

---

4) Notes on Using the MIMXRT1060-EVK Evaluation Board:

These steps are done in appropriate places in this tutorial. You do not need to complete them at this time.

1) Software Pack NXP::MIMXRT1062_DFP and NXP::MIMXRT1060_BSP: both version 12.1.0

Build errors: (735): too many positional arguments. An assembler option must be set.

1. Select Options for Target 🪄 or ALT-F7 twice. Click on the Asm tab.
2. In the Misc Controls box add: **-x assembler-with-cpp**
3. **IF MDK 5.30:** Select armclang (Auto Select):
4. GNU Syntax seems to work too.
5. This will be fixed in the next Pack release from NXP.

Language / Code Generation
Assembler Option: armclang (Auto Select) ▾
armclang (Auto Select)
armclang (GNU Syntax)
armclang (Arm Syntax)
armasm (Arm Syntax)

2) Select Load Application and Run to main():

1. Select Options for Target 🪄 or ALT-F7 twice.
2. Select the Debug tab: ☑ Load Application at Startup   ☑ Run to main()
3. Select Load Application at Startup:

3) Periodic Window Update:

This µVision attribute must be enabled to update various windows while the processor is running. If windows update only when you stop the processor (other than most Trace Data windows), this attribute is turned off.

1. When in Debug Mode and processor running or not, select View from main µVision menu.
2. Select Periodic Window Update.   ☑ Periodic Window Update
3. Select File/Save All or click:

This is not set by default in any of the examples provided in the Packs.

4) Enable Trace Clock for Serial Wire Viewer (SWV):

Serial Wire Viewer Data trace will not operate. Two registers must be set.

Configure CCM_CCGR0 Clock Gate Register 0:

In led_blinky.c, a call to BOARD_InitBootClocks(); near line 61 turns off the trace_clk_enable in Clock Gate Register 0. This must be enabled in order for SWV to operate. There are two ways to accomplish this: Do one of these:

Comment out the call to the function CLOCK_DisableClock: This is easiest.

1. In clock_config.c, comment out CLOCK_DisableClock(kCLOCK_Trace); which is found at line 285.

Modify led_blinky.c to configure the trace_clk_enable register.

1. In led_blinky.c near line 19, add: **#define trace_clock_enable (*((volatile unsigned int *)(0x400FC068)))**
2. Near line 59 right after BOARD_InitBootClocks();, add **trace_clock_enable=trace_clock_enable|0x00C00000;**

The SWO signal must be directed out the JTAG TDO pin.

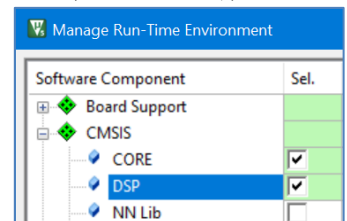### Configure ARM_TRACE_SWO: IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_10:

1. This directs the SWO signal to the JTAG TDO pin 13. This is on the large 20 pin JTAG connector J21.
2. Select Options for Target 🪄 or ALT-F7 twice. Click on the Debug tab.
3. In the Initialization box will be a *.ini file listed. Click Edit… to show it in the source windows.
4. In this ini file, just before the LOAD command near line 86, add: **_WDWORD(0x401F80E4, 0x00000009);**

5) DSP Libraries are selected in MRTE: iled_blinky project.

The DSP libraries are not used in this project. DSP can be unselected removing many files from the project.

Manage Run-Time Environment
Software Component	Sel.
⊞ ◆ Board Support	
⊟ ◆ CMSIS	
● CORE	☑
● DSP	☑
● NN Lib	☐

6) Use optimization -O0 for best debug visibility.

See https://developer.arm.com/documentation/100748/0614/Using-Common-Compiler-Options/Selecting-optimization-options?lang=en
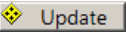

4

5) Software Pack Installation:

A Software Pack is a zip file with a .pack file extension. It contains header files, Flash programming algorithms, examples, documentation and more. The contents of a Pack are described by a .pdsc file contained inside the Pack. A DFP (Device Family Pack) contains contents for device(s) and optionally a BSP. A BSP (Board Support Package) contains information related to a specific board. A BSP can be inside a DFP or available separately. A Pack is installed using the μVision Pack Installer utility or installed manually. You can create your own Pack and distribute them for confidential projects.

1) Start μVision and Open Pack Installer (PI):

When the first MDK install is complete and if you are connected to the Internet, μVision and Software Packs will automatically start. Otherwise, follow Steps 1 and 2 below.

1. Connect your computer to the Internet. This is normally needed to download the Software Packs unless you have a standalone Pack. In this case, import it with Pack Installer or double-click on the .pack file it to manually install it.

2. Start μVision by clicking on its desktop icon. The Pack descriptions will download on the initial μVision run.

3. Open the Pack Installer by clicking on its icon: A Pack Installer Welcome screen will open. Read and close it.

4. If there are any Updates available, you can download them now if they are applicable.

5. The window below opens up: Select the Devices tab. Scroll down and expand NXP. Select MIMXRT1062 as shown below. You could select a specific NXP processor but in this case one Pack supports all of them at this time.

6. Alternatively, you can click on the Boards tab and select MIMXRT1060 to filter the Packs or Examples tabs.
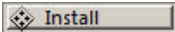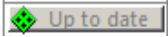
TIP: The Devices and Boards tabs are used to filter the items displayed on the right side in the Packs and Examples tabs.

7. Note: "ONLINE" is displayed at the bottom right. If "OFFLINE" is displayed, connect to the Internet.
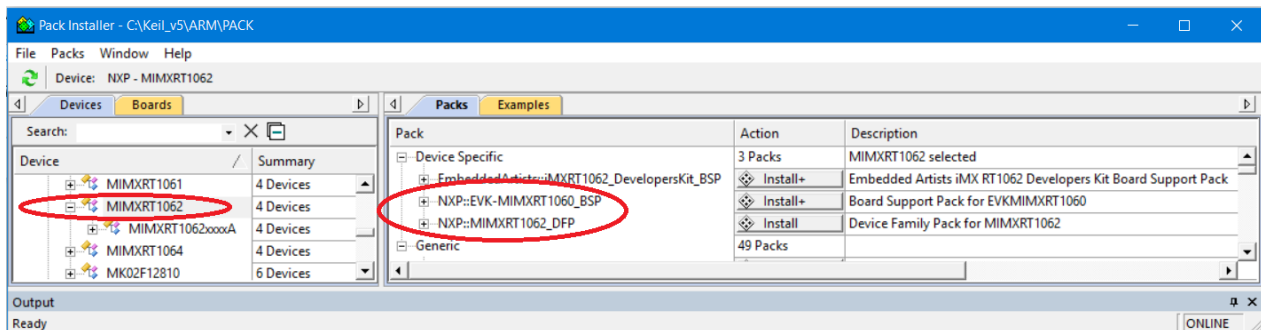
TIP: If there are no entries shown because you were not connected to the Internet when Pack Installer opened, select Packs/Check for Updates or to refresh once you have connected to the Internet.

2) Install MIMXRT1062 Device Family Pack NXP::EVK-IMXRT1062_DFP and EVK-IMXRT1060_BSP:

1. Click on the Packs tab.

2. Select the Install icon beside **NXP::EVK-IMXRT1062_DFP** as shown above.

3. The latest Pack will download and install to C:\Keil_v5\ARM\PACK\NXP\ by default. This download can take two to four minutes depending on your Internet connection speed.

4. Click Install on **NXP::EVK-IMXRT1060_BSP** board support package as shown above.

5. The two Packs status will then be indicated by the "Up to date" icon:

6. Leave Pack Installer open to copy the examples on the next page.

TIP: You can also install a Pack manually. A Pack has a file extension of .pack. It is an ordinary zip file with the extension changed so it is recognized by μVision. You can download the Pack from the web or transfer the file in any other way. Double click on this file and it will automatically be recognized (.pack) and installed by Pack Installer.

TIP: You can create your own Pack to distribute a DFP, BSP and an SDK. This is a way to distribute confidential material.

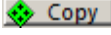CMSIS-Pack documentation: www.keil.com/pack/doc/CMSIS/Pack/html/

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com

## 6) Copy the iled_blinky Example to your PC using Pack Installer:

This example can be obtained from the Pack repository as described below. An updated copy is also provided at the website where you obtained this document. You can use either as the updates are incorporated in the instructions in this tutorial.

### 1) Copy the iled_Blinky example:

1. Select the Examples tab. There are many examples available. We will use one.

TIP: You could also select the Boards tab and search for RT1060 and then select EVK-IMXRT1060.

2. Beside iled_blinky (EVK-MIMRTX1060) as shown below: Select Copy: [Copy]
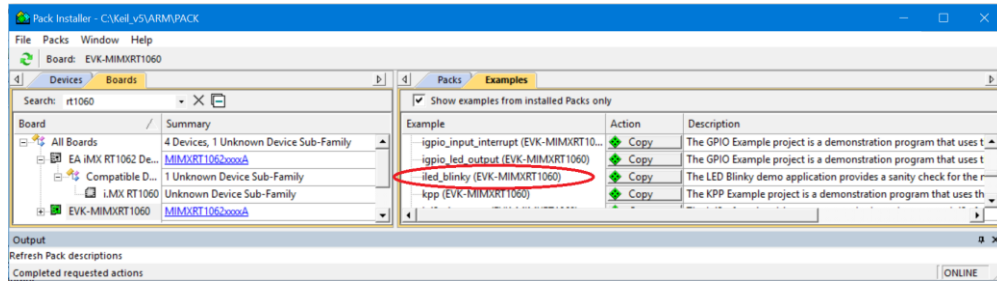
3. The Copy Example window below opens: Select Use Pack Folder Structure. Unselect Launch µVision:

4. Type in **C:\00MDK\iMXRT1060** as shown here:

5. Click OK to copy this example file into this folder:

6. Pack Installer creates the appropriate subfolders.

7. Close Pack Installer. Open it any time by clicking:

8. If a dialog box opens stating the Software Packs folder has been modified, select Yes to "Reload Packs ?"
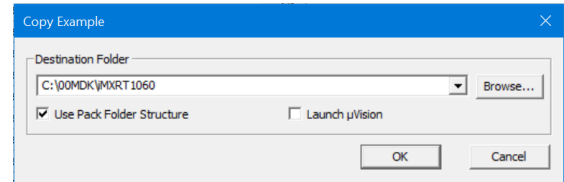
TIP: The default directory for copied examples the first time you install MDK is C:\Users\<user>\Documents. For simplicity, we will use the default folder of C:\00MDK\iMXRT1060\ in this tutorial. You can use any folder of your choosing.

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

At this point, you have everything loaded and installed to run some examples and start experimenting with various µVision debug features.

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com

7) Software Pack Version Selection and Manage Run-Time Environment:

These three sections are provided only for reference. These three utilities are useful for managing and using Software Packs. This page is for the LPC55S69. The i.MxRT1060 series options and components are similarly configured and maintained.

These windows allow you to manage Packs and software components in your µVision projects.

You probably will not see the exact examples as shown because you have different Packs installed. This section contains three parts on this page and the next one:

        A) Select Software Pack Version:

        B) Manage Run-Time Environment:

        C) Updating Source Files:

A) Select Software Pack Version: This section is provided for reference:

These screens show the LPC55S69 processor but the MIMXRT1060 is similar.

This µVision utility provides the ability to choose various Software Pack versions installed in your computer. You can select the versions you want to use. You must have µVision running and any project open for the following exercises:

1. Select Project/Open Project and navigate to C:\00MDK\LPC55S69\demo_apps\led_blinky\cm33_core0\mdk\

2. Select led_blinky.uvprojx and it will load into µVision. You can load any project for this demonstration.

3. Open Select Software Packs by clicking on its icon:

4. The window below opens: Note **Use latest versions …** is selected.

5. Unselect this setting and the window changes as shown similar to the one below right:
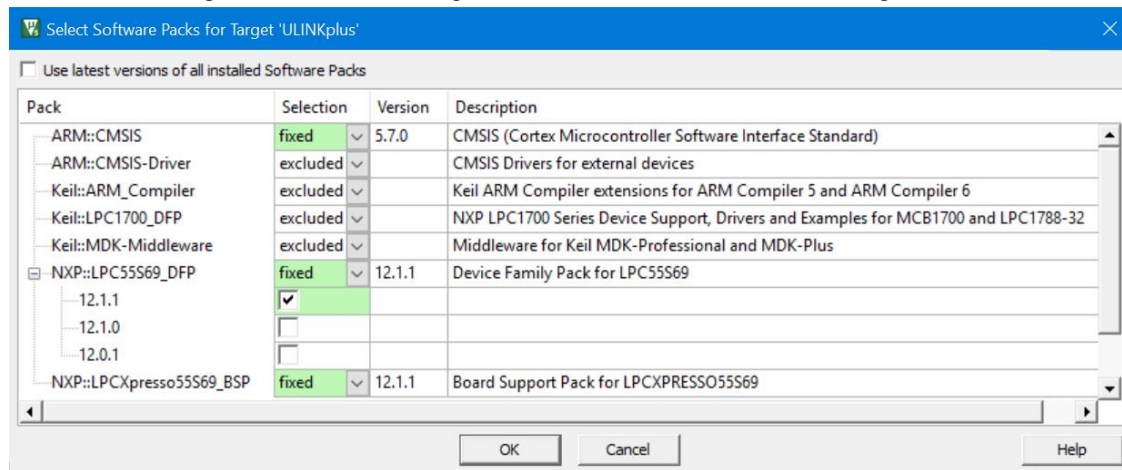

6. Expand the header NXP::LPC55S69_DFP as shown above:

7. You will see probably only one version – the one you installed. If you had installed others, you would see them as listed above and available to choose from.

8. Select the fixed pull-down menu and see the three options as shown here:

9. If you wanted to use a different version, you would select fixed and then select the check box opposite the version you wanted to use in your project. Make sure you check only one box.
   These settings are stored with your project. This is a good way to freeze software components for your designs.
   These settings are saved with the project selected.

10. Re-select Use latest versions… **Do not make any changes at this time.**

11. Click OK or Cancel to close this window.

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com

B) Manage Run-Time Environment: This section is provided for reference:

The Manage Run-Time Environment utility selects components from various Software Packs and inserts them into your Project list. From here, you can access these files in the usual manner usually by calling functions.

1. Click on the Manage RTE icon: The window below opens: This includes Keil Middleware, selected open source software, RTX and CMSIS drivers for various peripherals. Not all Packs offer all options but more are being added.

2. Expand various headers and note the selections you can make. A selection made here will automatically insert the appropriate source files into your project.

3. Note CMSIS/Core (system.c), Keil RTX and Device/Startup (startup.s) files are selected. You can see the appropriate files listed in the µVision Project window.

4. **Do not make any changes.**
    Click Cancel to close this window.

TIP: Selections made in MRTE are stored in .\RTE*Options*\RTE_Components.h
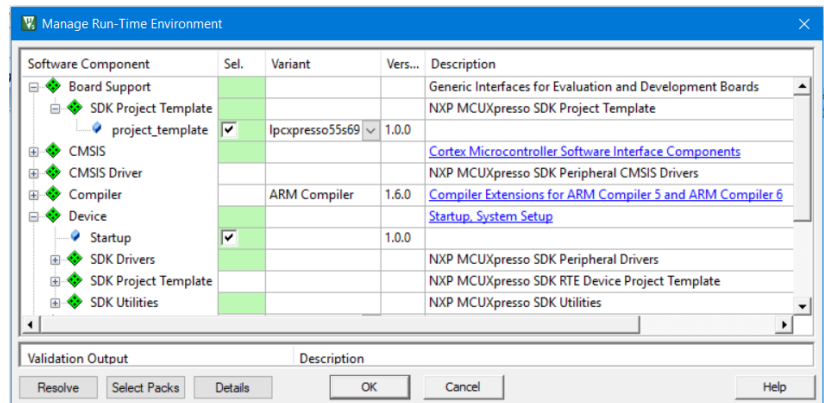
Select column colour meanings:

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.

C) Updating Source Files: *this section is provided for reference:*

Some of the source files provided by a Software Pack are stored in your project. Examples are startup.s and System.c.
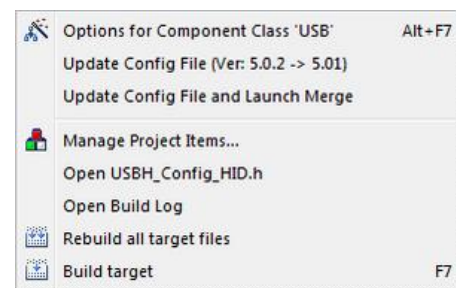
If you update a Software Pack, some of these files might need to be updated in µVision. These files will show new icons as shown below and described here: www.keil.com/support/man/docs/uv4/uv4_ca_updswcmpfiles.htm

Updating Source Files: **Note:** Any changes you made to a file will be lost when it is replaced with the new one.

1. Right click on a file you want to update. A window similar to the one below right opens:

2. Select the appropriate Update selection. Remember, any changes you made to the original file are lost.

3. This procedure is described here: www.keil.com/support/man/docs/uv4/uv4_ca_filegrp_att.htm

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com

8) CoreSight Definitions: It is useful to have a basic understanding of these terms:

Cortex-M0 and Cortex-M0+ may have only features 2) and 4) plus 11), 12) and 13) implemented. Cortex-M3, Cortex-M4 and Cortex-M7 can have all features listed implemented. MTB is normally found on Cortex-M0+. It is possible some processors have all features except ETM Instruction trace and the trace port. Consult your specific datasheet.

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 TDO shares the same pin as SWO. The SWV data normally comes out the SWO pin or 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 accesses which provide on-the-fly memory accesses without the need for processor core intervention. µVision uses the DAP to update Memory, Watch, Peripheral and RTOS kernel awareness windows while the processor is running. You can also modify variable values on the fly. No CPU cycles are used, the program can be running and no code stubs are needed. 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 trace capability providing display of reads, writes, exceptions, PC Samples and printf.

6. **SWO:** Serial Wire Output: SWV frames usually come out this one pin output. It shares the JTAG signal TDO.

7. **Trace Port:** A 4 bit port that ULINK*pro* uses to collect ETM frames and optionally SWV (rather than SWO pin).

8. **ITM:** Instrumentation Trace Macrocell: As used by µVision, ITM is 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.

9. **ETM:** Embedded Trace Macrocell: Displays all the executed instructions. The ULINK*pro* provides ETM. ETM requires a special 20 pin CoreSight connector. ETM also provides Code Coverage and Performance Analysis. ETM is output on the Trace Port or accessible in the ETB (ETB has no Code Coverage or Performance Analysis).

10. **ETB:** Embedded Trace Buffer: A small amount of internal RAM used as an ETM trace buffer. This trace does not need a specialized debug adapter such as a ULINK*pro*. ETB runs as fast as the processor and is especially useful for very fast Cortex-A processors. Not all processors have ETB. See your specific datasheet.

11. **MTB:** Micro Trace Buffer. A portion of the device internal user RAM is used for an instruction trace buffer. Only on Cortex-M0+ processors. Cortex-M3/M4 and Cortex-M7 processors provide ETM trace instead.

12. **Hardware Breakpoints:** The Cortex-M0+ has 2 breakpoints. The Cortex-M3, M4 and M7 usually 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 CPU is halted before the instruction is executed.

13. **Watchpoints:** Both the Cortex-M0, M0+, Cortex-M3, Cortex-M4 and Cortex-M7 can 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. There also referred to as Access Breaks in Keil documentation.

Read-Only Source Files:

Some source files in the Project window will have a yellow key on them: This means they are read-only. This is to help unintentional changes to these files. This can cause difficult to solve problems. These files normally need no modification.

If you need to modify one, you can use Windows Explorer to modify its permission.
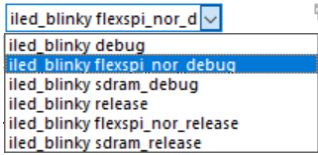
1. In the Projects window, double click on the file to open it in the Sources window.

2. Right click on its source tab and select Open Containing folder.

3. Explorer will open with the file selected.

4. Right click on the file and select Properties.

5. Unselect Read-only and click OK. You are now able to change the file in the µVision editor.

6. It is a good idea to make the file read-only when you are finished modifications.

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

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com
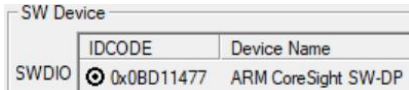
1) iled_blinky example using MIMXRT1060-EVK Evaluation Board:

Now we will run the Blinky example on the Keil MDK development system using the MIMXRT1060-EVK evaluation board. You can adapt these instructions for use on other NXP boards including custom boards.

Open the Blinky example program:

1. Start µVision by clicking on its desktop icon.
2. Select Project/Open Project from the main menu.
3. Open the file C:\00MDK\iMXRT1060\demo_apps\led_blinky\mdk\iled_blinky.uvprojx.
4. Connect your PC USB cable to J41 USB connector. You will hear the standard USB dual tone sound.
5. Select the iled_blinky flexspi nor Debug target:

Confirm LPC-Link is connected to The Target Processor:

1. Select Options for Target or ALT-F7 twice.
2. Select the Debug tab:
3. Select Load Application at Startup:
4. Select Settings on the right where your debug adapter is displayed:
5. A valid SW Device IDCODE *must* be displayed as shown here.

Trouble ? If nothing or an error is displayed, check the LPC-Link2 firmware programming and USB connections. This *must* be solved before you can continue. Make sure the jumpers are set to their defaults.

Configure Options for Target:

1. Select the Trace tab. In Core Clock: enter 600 MHz. This is the CPU speed. We will use this later for timing.
2. Select the C/C++ tab. Set the compiler Optimization: to -O0 for best debug visibility.

 TIP: See https://developer.arm.com/documentation/100748/0614/Using-Common-Compiler-Options/Selecting-optimization-options?lang=en

3. When completed, click OK twice to close the Options for Target windows.
4. Select File/Save All or click:

Compile, Load and Run the Project:

1. Compile the source files by clicking on the Rebuild icon.   Progress is indicated in the Build Output window.

Trouble Compiling ? If compiler errors result, see page 4 step 1 for a solution.

2. Enter Debug mode by clicking on the Debug icon.   Select OK if the Evaluation Mode notice appears. Flash programming progress is indicated in bottom left corner.
3. Click on the RUN icon   to start the Blinky program. If the green RGB D18 blinks, ***Blinky is now running !***
4. The red LED D20 blinking indicates LPC-LINK2 CMSIS-DAP communication. Attempt to ignore it.
5. Stop the program with the STOP icon.

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

> *The green LED will blink !   It will be on for 1 second and off for one second.*
>
> **Now you know how to compile a program, load it into the i.MXRT1060 Flash, run it and stop it.**

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com
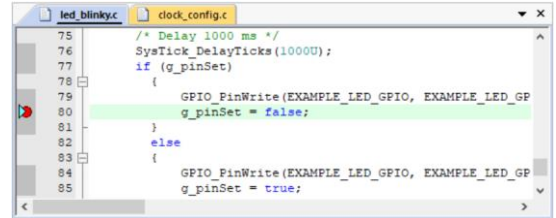
2) Hardware Breakpoints:

1. Click on the RUN icon 📑 .

2. Bring led_blinky.c in focus by clicking on its tab. If it is not visible, double-click on it in the Project window.

3. In led_blinky.c, scroll down to inside the while (1) loop found starting near line 73 as shown here: ➡️

4. Note on the left of the line numbers are darker grey blocks. This indicates there is assembly code present and you can set a hardware breakpoint on these lines. You can also see these blocks in the Disassembly window.

5. *While the program is still running*, click to the left of a suitable line in the while loop and a red circle will be created. This is a hardware breakpoint. i.MXRT1060 has 8 hardware breakpoints. μVision will warn you if you exceed this limit.

6. The program will soon stop at the line where you set the breakpoint on as shown below. The yellow arrow is the current program counter position. This will be the next instruction executed. The cyan arrow is a placeholder you can use to explore the relationship between a source window and the Disassembly window.

7. Add another breakpoint in while (1) or in a suitable place in led_blinky.c such as SysTick Handler near line 35.

8. Each time you click on RUN 📑 , the program will cycle to the next breakpoint.

TIP: You can set/unset hardware breakpoints with μVision while the program is running or stopped. This is an important feature.
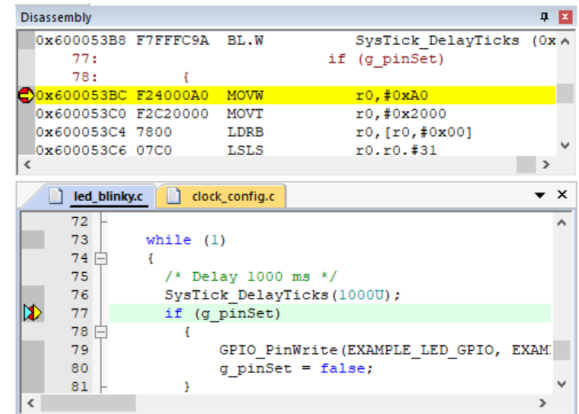
TIP: Arm hardware breakpoints are no skid. They do not execute the instruction they are set to. They do not change the instruction they are set to. These are also important features.

Compiler Optimizations: Sometimes the compiler will optimize out code. Breakpoints cannot be set on this code as evidenced by the lack of the grey blocks. Setting the compiler to the lowest optimization -O0 will usually help.

Select Target Options 🛠️ and select the C/C++ tab to set the level.

Managing Breakpoints:

1. Select Debug/Breakpoints or Ctrl-B. This window opens:

2. You can temporarily unselect, delete or create breakpoints in this window. It is easier to create a breakpoint by clicking in a source file or the Disassembly window.

3. Watchpoints are also created in this window. This is discussed in a few pages.

4. **Select Kill All to remove all breakpoints.**

5. Click Close.

6. Click RUN 📑 for the next exercise.

TIP: If you get multiple cyan arrows or can't understand the relationship between the C source and assembly, try lowering the compiler optimization to Level -O0 and rebuilding your project. Compilers can do strange things while optimizing code.

With AC6, the lowest level recommended level is -O1.

The optimization level is set in Options for Target 🛠️ under the C/C++ tab when not in Debug mode.

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com
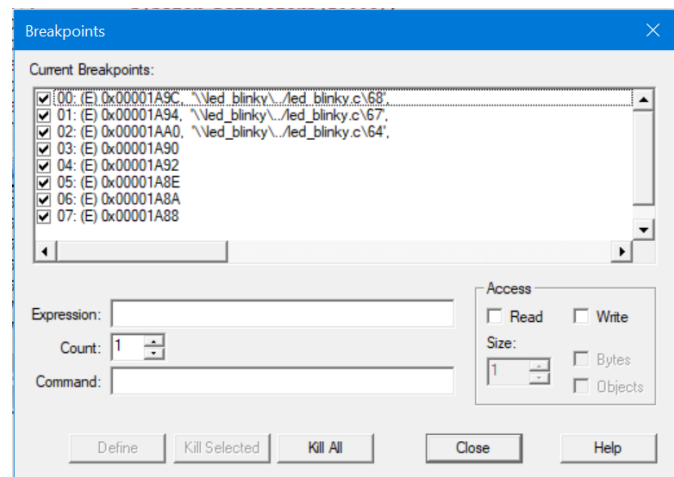
## 3) Call Stack + Locals Window:

Local Variables:

The Call Stack and Local 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 belonging to the active function.

If possible, the values of the local variables will be displayed and if not the message <not in scope> or similar will be displayed. The Call + Stack window visibility can be toggled by selecting View/Call Stack window.

1. Stop the program with the STOP icon. ⊗ The program will probably stop in the SysTick_DelayTicks function.

2. Click on the Call Stack + Locals tab in the bottom right corner of µVision.

3. Inspect the various entries in the Call Stack + Locals window as shown below right in this very simple example. Local variables are displayed only when they are in scope.

TIP: If you set a breakpoint inside the Systick Handler near line 33, the program will stop there and show SysTick_Handler in the Call Stack window.
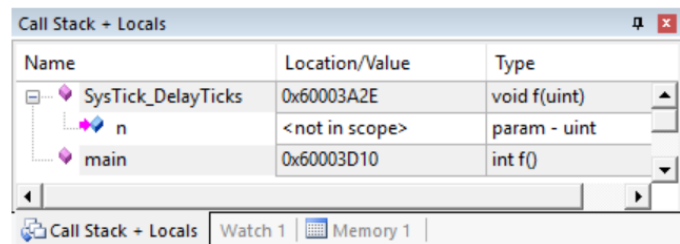
4. Set a breakpoint in the while(1) loop near line 73 on the if statement if (g_pinSet) as shown here:

5. Click RUN ⬇.

6. The program will stop here:
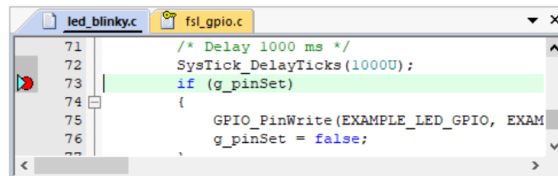
7. Click Step ⟨⟩ or F11 with led_blinky.c in focus (its name in its tab is underlined).

8. You will enter this function as shown here:

9. Note the variables displayed.

10. Note a structure base is shown. Expand its elements.

11. Click Step ⟨⟩ or F11 a few more times:

12. You will return to the SysTick_DelayTicks function.

13. **REMOVE** the Breakpoint to continue.
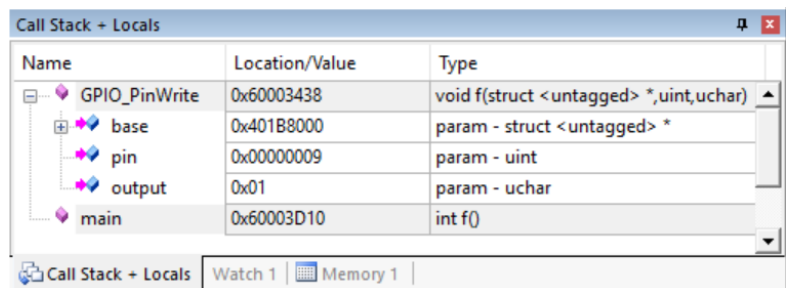
TIP: You can also use Step Out icon ⟨⟩ to exit a function. This is will be indicated in the Call Stack window.

TIP: This is standard "Stop and Go" debugging. ARM CoreSight debugging technology can do much better than this. You can display global or static variables and structures updated in real-time while the program is running. No additions or changes to your code are required. They must be converted to global or static variables so they always remain in scope.

TIP: Single Step: ⟨⟩ (F11): If you click inside a source window to bring it into focus, the program will step one source line at a time. If you click inside the Disassembly window to bring it into focus, it will then step one instruction at a time.

Call Stack Uses:

The list of stacked functions is displayed when the program is stopped. This is when you need to know which functions have been called and are stored on the stack. This is useful to find program flow problems such as crashes.

Normal functions are displayed only when they are on the Stack. They are removed and added as appropriate. When RTX is used, the threads are always displayed, even if not running. The active thread name and its number are highlighted in green.

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com

4) 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. It is also possible to "put" or insert values into these memory locations in real-time using the Memory window while the program is running. You can change memory in a Watch window if data is not changing fast.

There is a global variable g_systickCounter located in led_blinky.c near line 38 we can use in the Watch and Memory windows.

Watch window:

1. Click on the RUN icon to start the led_blinky program.

2. Right click on g_systickCounter in led_blinky.c near line 44 and select Add 'g_systickCounter to… and select Watch 1.

3. Select View in the main menu and enable Periodic Window Update:

4.   g_systickCounter will be displayed in Watch 1 as shown above:

5. The values of g_systickCounter are updated in real-time if Periodic Window Update is enabled.
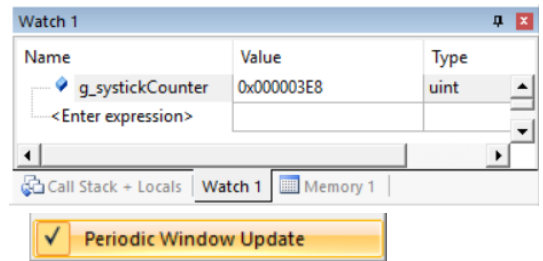
TIP: You do not need to stop the program to enter variables, raw addresses or structures in a Watch or Memory window.

6. You can modify the value in a Watch window when the program is stopped or changing slowly. You can modify a variable in a Memory window anytime. Step 5 below illustrates this technique.

SystemCoreClock: CMSIS provides this global variable which stores the CPU clock frequency.

1. Click on <Enter expression> twice and enter: SystemCoreClock

2. Right click on the Value and deselect Hexadecimal Display. 600 MHz will be displayed.

Memory window:

1. Right click on g_systickCounter and select Add 'g_systickCounter ' to… and select Memory 1.

2. Note the changing value of g_systickCounter is displaying its address 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. Right click in Memory 1 and select Unsigned Long to see the data field as 32-bit numbers.

4. Add an ampersand "&" in front of the variable name g_systickCounter and press Enter. Now the physical address is shown (0x2000_0014) in this case. This physical address could change with different compilation optimizations.

5. The data contents of g_systickCounter is displayed as shown:

6. Right click on the memory data value and select Modify Memory. Enter a value and this will be pushed into g_systickCounter. Since g_systickCounter is updated often, you will not see a new value displayed. But the update did occur.
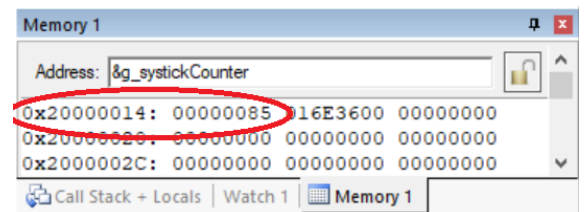
TIP: You can configure the Watch and Memory windows while the program is running in real-time without stealing any CPU cycles. You can change a Memory window value on-the-fly.

7. The global variable g_systickCounter is updated in real-time. This is ARM CoreSight technology working.

8. Stop the CPU and exit Debug mode for the next step.

TIP: View/Periodic Window Update must be selected. Otherwise variables update only when the program is stopped.

How It Works:

µVision uses ARM CoreSight technology to read or write memory locations without stealing any CPU cycles. This is nearly always non-intrusive and does not impact the program execution timings. The Cortex-M3, M4 and M7 are Harvard architectures. This means they have separate instruction and data buses. While the CPU is fetching instructions at full speed, there is plenty of time for the CoreSight debug module to read or write to memory without stealing any CPU cycles.

This can be slightly intrusive in the unlikely event the CPU and µVision reads or writes to the same memory location at exactly the same time. Then the CPU will be stalled for one clock cycle. In practice, this cycle stealing never happens.

Remember you are not able to view local variables while the program is running. They are visible only when the program is stopped in their respective functions. You must change them to a different type of variable such as global to see them update.

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com

5) System Viewer (SV):

System Viewer provides the ability to view certain registers in peripherals and the CPU core. In many cases, these windows are updated in real-time while your program is running. They are available only while in Debug mode. Select Peripherals/System Viewer to open the peripheral windows. Select CPU core registers by opening Peripherals/Core Peripherals. Both of these are illustrated here:

In our iled_blinky example, the LED D18 is output on GPIO1 bit 09.

1. Click on RUN ![run icon]. You can open SV windows running or not.

PIO Port 1:

1. Select Peripherals/System Viewer and then GPIO then GPIO1 as shown:

2. The GPIO1 window opens:

3. Expand DR and GDIR as shown in the GPIO1 window:

4. The DR value changes (slowly) as the program runs.

5. This window is updated periodically using the same CoreSight DAP technology as the Watch and Memory windows. For fast changing data, some values might be skipped.

6. Stop the CPU ![stop icon]

7. When you change this to a "1", the LED goes out. Change it to a "0" and it comes on.

8. You can change the values in the System Viewer while the program is running or stopped. It will be difficult to see this as these values in this case are updated so often that your changes will be overwritten.

9. Some peripheral registers might not allow you to change them.
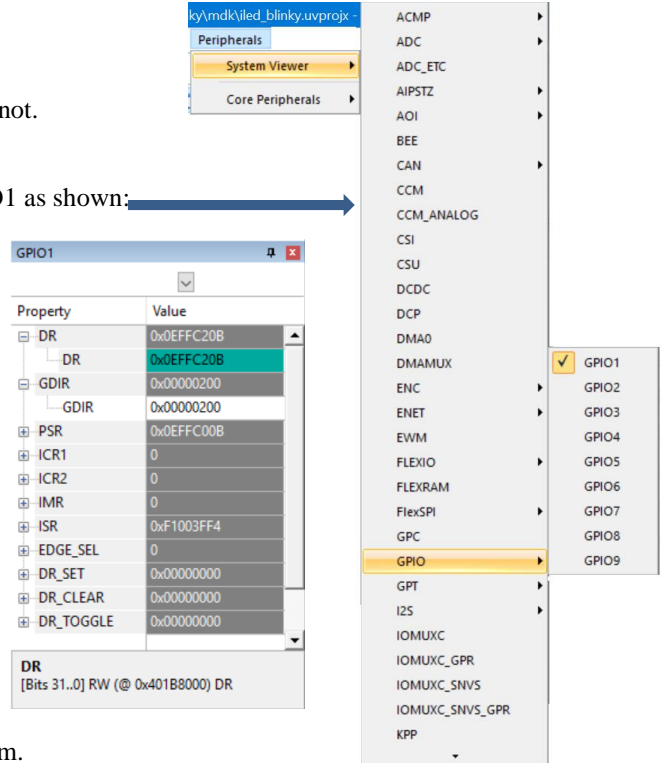
TIP: If you click on a register in the Property column, a description about this register will appear at the bottom of the window as shown here for GPIO. This is an easy method to find physical registers for various peripherals.

SysTick Timer: The iled_blinky example program uses the Cortex SysTick timer for a delay function.

1. Select Peripherals/Core Peripherals and then select SysTick Tick Timer (SysTick). Run the program.

2. The SysTick Timer window shown below opens:

3. Note it also updates in real-time while your program runs using CoreSight DAP technology.

4. Note the SysTick ->Load register. This is the reload register value. This value is set in led_blinky at line 62:
   if (SysTick_Config(SystemCoreClock / 1000U))

5. Note RELOAD is set to 0x0927BF = dec 599,999. This is created by (600 MHz/1000)-1 = 599,999. 1000 is specified as the timer tick value. A SysTick S interrupt 15 will occur every 1 msec. Changing the reload value changes how often the SysTick interrupt 15 occurs.

6. In the LOAD register in the SysTick window, *while the program is running,* type in 0x10000 and click in another register or press Enter.

7. The blinking LED will speed up. This will convince you of the power of ARM CoreSight debugging.
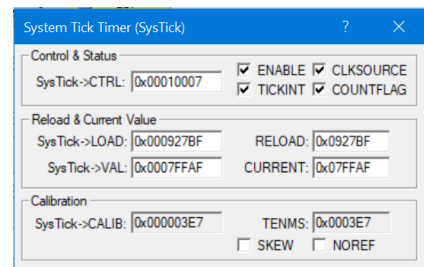
8. Replace RELOAD with 0x0000_927BF. A STOP and CPU RESET ![RST icon] will also accomplish this.

9. When you are done, Stop the program ![stop icon] and close all the System Viewer windows that are open.

TIP: It is true: you can modify values in the SV 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.

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com

6) Watchpoints: Conditional Breakpoints (Access Breakpoints)

Most NXP Cortex-M3, M4, M7 and M33 processors have four data comparators. Since each Watchpoint uses two comparators, you can configure two complete Watchpoints. Watchpoints can be thought of as conditional breakpoints. The Logic Analyzer uses the same comparators in its operations. This means in μVision you must have two variables free in the Logic Analyzer to use Watchpoints. Currently, μVision supports only one Watchpoint.

Configure Watchpoint:

1. Using the example from the previous page, RUN or STOP the program ❌ Stay in Debug mode.

2. In the main menu, select Debug/Breakpoints… or select Ctrl-B on your keyboard. The Breakpoints window opens.

3. In the Expression box, enter g_systickCounter==0x44 as shown below.

4. Select both the Read and Write Access.

5. Click on Define and it will be accepted as shown here: (the Expression: box will go blank)

6. Click on Close.

7. Click on RUN 📥.

8. When g_systickCounter = 0x44 the program will halt. This is how a Watchpoint works.

9. You will see g_systickCounter displayed with a value of 0x44 in the Watch window as shown below right:

10. **Delete the Watchpoint** by selecting Debug and select Breakpoints and select Kill All.

11. Select Close.

12. Leave Debug mode. 🔍
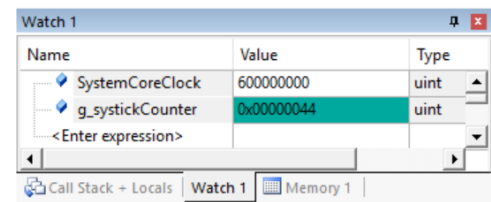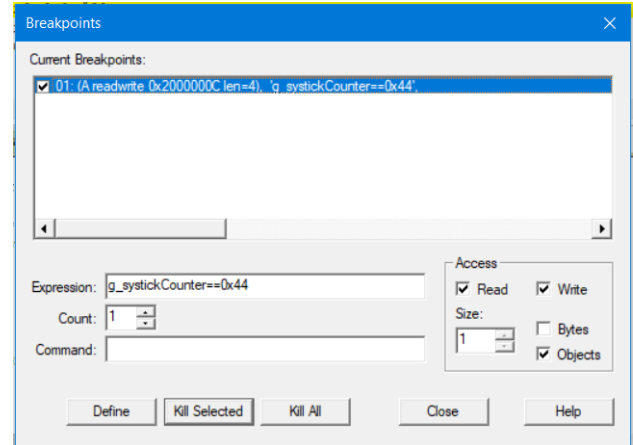
## Using the Watchpoint in the Stack Space:

If you put a RAM address as the expression with no value, the next read and/or write (as you selected) will cause the program to halt.

This can be particularly useful in the Stack. Set an address at some limit and if the program reads or writes this address, the program stops.

1. In this example a Watchpoint is created with address 0x2000_0014.

2. When the program is run the first R or W will stop the processor.

3. The Command window shows the setting of this Watchpoint and its execution including the approximate instruction location. In this case it is 0x3A2E.

TIP: SKID: The instruction noted will not be the instruction causing the break as it has not been executed yet. It will be noted in Disassembly.

The instruction causing the break is probably a few instructions before this one in the execution stream.
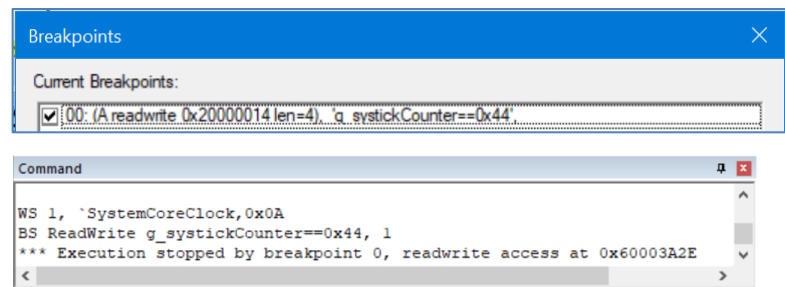
TIP: To edit a Watchpoint, double-click on it in the Breakpoints window and its information will be dropped down into the configuration area. Clicking on Define will create another Watchpoint. You should delete the old one by highlighting it and click on Kill Selected or use the next TIP:

TIP: The checkbox beside the expression in Current Breakpoints as shown above allows you to temporarily unselect or disable a Watchpoint without deleting it.

TIP: You can create a Watchpoint with a raw address and no variable value. This is useful for detecting stack overruns. Physical addresses can be entered as *((unsigned long *)0x20000000). Or simply enter the address as shown above.
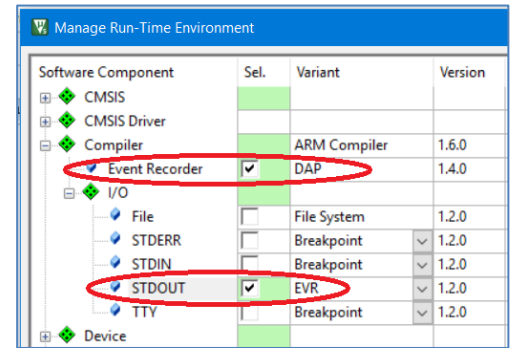
7) printf without using a UART: This printf uses Keil Event Recorder (EVR)

µVision provides a simple printf utility using Keil Event Recorder. No UART is needed and it is much faster and takes less code than a standard UART or USB COM port. Text is displayed in the Debug (printf) Viewer shown below.

Event Recorder (EVR) is a new µVision feature that can be used to instrument your code. Keil RTX5, Keil Middleware and CMSIS-FreeRTOS is already instrumented with Event Recorder. This can be used with any Cortex-M processors. DAP is the same technology used in Watch, Memory and Peripheral windows. LPC-Link2 supports Event Recorder.

Configure Event Recorder:

1. Stop the program if it is running 🛑 and exit Debug mode. 🔍

2. Open the Manage Run-Time Environment utility. 🔷 This opens:

3. Expand Compiler and I/O as shown.

4. Select Event Recorder (DAP) and STDOUT (EVR) as shown:

5. All the blocks should be green. If not, click on the Resolve button.

6. Open CMSIS header and unselect DSP. These are not needed.

7. Click OK to close this window.

8. retarget_io.c and EventRecorder.c will be added to your project under the Compiler group in the Project window.

9. Right click near the top of led_blinky.c, at line 9 and select Insert "#include" and select #include "EventRecorder.h".

10. In the main() function near line 61 just after SystemCoreClockUpdate, add this line:
    EventRecorderInitialize (EventRecordAll, 1);

11. Right after that line, add this line near line 62: EventRecorderStart ();

| Software Component	Sel.	Variant	Version
⊞ ◆ CMSIS			
⊞ ◆ CMSIS Driver			
⊟ ◆ Compiler		ARM Compiler	1.6.0
✔ Event Recorder	☑	DAP	1.4.0
⊟ ◆ I/O			
🔵 File	☐	File System	1.2.0
🔵 STDERR	☐	Breakpoint	1.2.0
🔵 STDIN	☐	Breakpoint	1.2.0
🔵 STDOUT	☑	EVR	1.2.0
🔵 TTY	☐	Breakpoint	1.2.0
⊞ ◆ Device			

Create global variable counter:

1. In led_blinky.c, near line 31, declare the global variable counter: uint32_t counter = 0;

2. In led_blinky.c add these two lines just after the SysTickDelayTicks statement starting at line 76:

    76   counter++;
    77   if(counter>0x0F) counter = 0;

Add a printf statement to led_blinky.c:

1. In led_blinky.c add #include "stdio.h" near the top of the file near line 10.

2. In led_blinky.c, near/at line 78 just after the statement if(counter>0x0F…, add: printf("Hello %d\n", counter);

3. Select File/Save All or click: 💾

Build and RUN the Blinky program and view printf:

1. Rebuild the source files 🔨.

2. Enter Debug mode 🔴. Click on RUN 📲.

3. Select View/Serial Windows and select Debug (printf) Viewer.

4. The values of counter are displayed as seen here:

5. Open the Event Recorder window:

6. Information about the printf statements are displayed.

7. Right click on a line and set Time Reference. Hover the mouse over the next frame. The printf statements occur 1 second apart.

8. Stop the program. 🛑 Exit Debug mode 🔴.

```
Debug (printf) Viewer
Hello 5
Hello 6
Hello 7
Hello 8
```
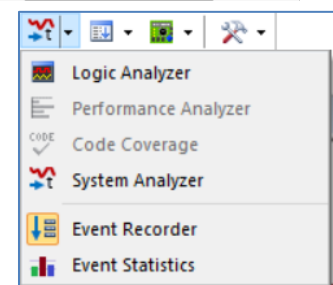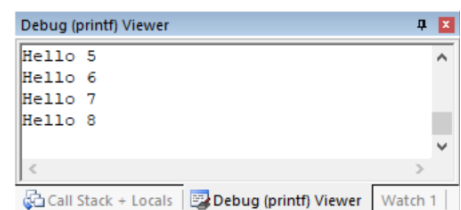
TIP: If you get a Hard Fault error, you must either select MicroLIB *or* add some heap in the correct startup.s file or the scatter file if created by your project.

Try adding a 200 byte heap. MicroLIB results in a smaller executable size and is a very useful compiler setting.

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com

1) Event Recorder (EVR):

A µVision feature where you can annotate your source code which can be displayed in various information windows. We will demonstrate this feature. It uses CoreSight DAP to output data from the target and not SWV. This means any debug adapter can be used. Keil Middleware and RTX5 is already annotated. EVR requires a certain amount of system RAM.

For more information on Event Recorder: www.keil.com/support/man/docs/uv4/uv4_db_dbg_evr.htm
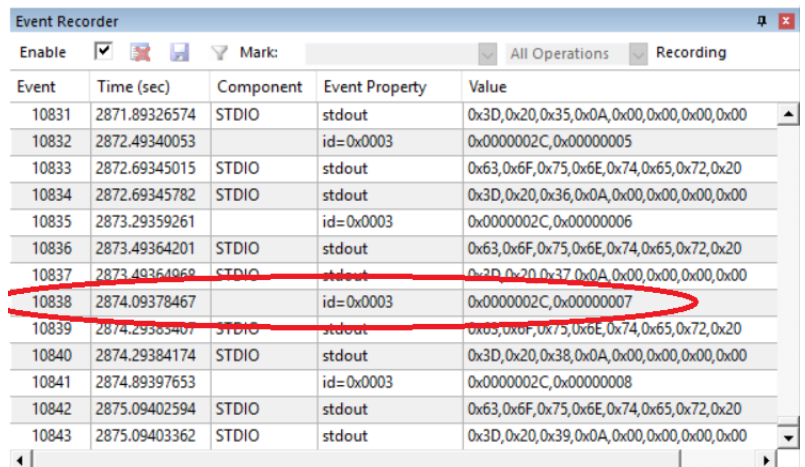
1) Configure Event Recorder:

1. Confirm the steps on the previous page under Configure Event Recorder were completed.

Add Event Record Event:

1. Add this function call: EventRecord2(3, 44, counter); just before Systick_DelayTicks(1000u); which is at line 75.

2. Select File/Save All or click:

Compile and RUN the Program:

1. Compile the source files with the Rebuild icon.

2. Enter Debug mode. Select OK if the Evaluation Mode notice appears.

3. Select the small arrow beside Analysis icon and select Event Recorder which will open: ➡️

4. Click on the RUN icon to start the Blinky program.

5. Events will start to display in the Event Recorder window as shown below:

6. You can see the results of the EventRecord2(3, 44, counter); event you entered displayed:

7. printf frames (stdout) are also displayed with the printf data in hex form.

8. This provides a useful time link to your source code.

9. Stop the recording by unselecting Enable in the upper left corner.

10. When you re-enable it, events were collected in the background as the program ran and will be displayed.

11. Hover your mouse over Event Property entries to gain more information:

12. You can modify the information displayed with the Funnel icon:
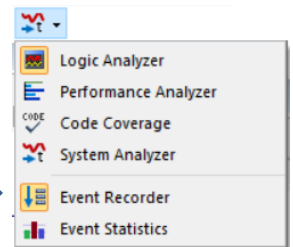
Event	Time (sec)	Component	Event Property	Value
10831	2871.89326574	STDIO	stdout	0x3D,0x20,0x35,0x0A,0x00,0x00,0x00,0x00
10832	2872.49340053		id=0x0003	0x0000002C,0x00000005
10833	2872.69345015	STDIO	stdout	0x63,0x6F,0x75,0x6E,0x74,0x65,0x72,0x20
10834	2872.69345782	STDIO	stdout	0x3D,0x20,0x36,0x0A,0x00,0x00,0x00,0x00
10835	2873.29359261		id=0x0003	0x0000002C,0x00000006
10836	2873.49364201	STDIO	stdout	0x63,0x6F,0x75,0x6E,0x74,0x65,0x72,0x20
10837	2873.49364969	STDIO	stdout	0x3D,0x20,0x37,0x0A,0x00,0x00,0x00,0x00
10838	2874.09378467		id=0x0003	0x0000002C,0x00000007
10839	2874.29385407	STDIO	stdout	0x63,0x6F,0x75,0x6E,0x74,0x65,0x72,0x20
10840	2874.29384174	STDIO	stdout	0x3D,0x20,0x38,0x0A,0x00,0x00,0x00,0x00
10841	2874.89397653		id=0x0003	0x0000002C,0x00000008
10842	2875.09402594	STDIO	stdout	0x63,0x6F,0x75,0x6E,0x74,0x65,0x72,0x20
10843	2875.09403362	STDIO	stdout	0x3D,0x20,0x39,0x0A,0x00,0x00,0x00,0x00

2) Running EVR in Non-initialized Memory:

It can be important to preserve the EVR data located in target RAM memory in the event of a crash and/or RESET. Creating and using non-initialized memory is implemented by modifying the Scatter file.
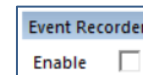If a custom scatter file is not used and one is created by µVision: see www.keil.com/support/docs/4012.htm

You will see a Warning that Event Recorder is not located in uninitialized memory.
You can ignore this warning for the purpose of this tutorial.

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com

3) Determining Relative Timing Values:

1. Unselect Enable in the Event Recorder window to stop the collection of data:

   Event Recorder
   Enable ☐

2. Right click on the first in a sequence of stdout frames and select Set Time Reference as shown below:

3. The selected frame will turn from blue to green.

4. Position your mouse pointer on the Time (sec) column on the next event frame.

5. A box will open displaying the elapsed time. It took 1.0 sec from the start of this printf to the next one.

6. **Enable the Event Recorder** so the frames continue to be captured and displayed.

7. You can use this feature to time many different events in your code.

Event Time = 2.00022165
Delta Time = +1.00000000
Reference Time = 1.00022165

TIP: Using printf with a 9600 baud UART with 8 characters uses about 80,000 CPU cycles or about 8 msec. Using Event Recorder is much faster taking only ~ 500 cycles. Event Recorder is 10 times faster than a UART running at highest speeds. Using an Event such as StartB(1) with 8 bytes is even faster: only ~250 CPU cycles.

4) Filtering the Event Recorder Window:

It is possible to filter the window contents. This is done in the Show Event Events. You can specify what elements are collected and displayed in the Event Recorder window.

1. Stop the program if necessary. Stay in Debug mode.

2. In the Event Recorder window, select the Filter icon:
   The Show Event Levels window opens up as shown below:

3. Unselect all boxes opposite STDIO as shown here:

4. Click OK to close this window.

5. Click Clear to make it easier to see what is happening.

6. Click on RUN .

7. The Event Recorder window no longer contains printf frames as shown below:
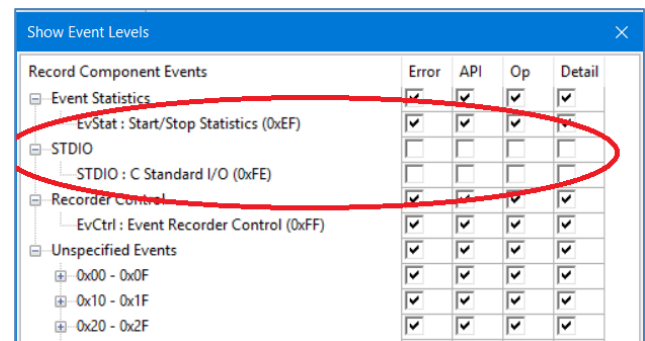
In this case, you only need to unselect the Op column. The other frames do not exist in our simple example.

Saving the Filter Settings:

You can save and recall the filter settings. See:

www.keil.com/support/man/docs/uv4/uv4_cm_er.htm

The commands are:

ER SAVE path\filename

ER LOAD path\filename

www.keil.com/support/man/docs/uv4/uv4_db_dbg_evr_setup.htm

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com
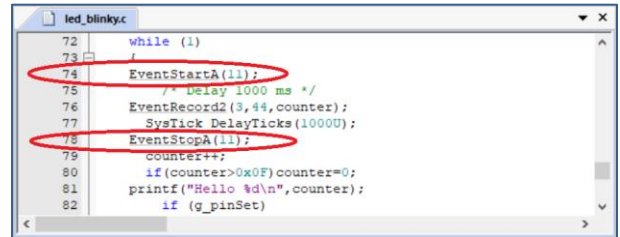
5) Event Statistics:

You can add Start and Stop events to your source code. Information collected will include execution counts and times. If you are using a Keil ULINK*plus*, information can also include voltage, current and total charge (Q) consumed. Individual and aggregate times are provided. This information will be collected between the Start and Stop Event tags including the execution of any exception handlers or program branches. Data is collected from START to the corresponding STOP event. CPU clock speed is measured and displayed graphically.

- **START:** The basic function call is EventStartG(slot) and EventStartG(slot, v1, v2)
- **STOP:** The basic function call is EventStopG(slot) and EventStopG(slot, v1, v2)
- These calls are arranged in four groups (G = A, B, C, D). **v** is for data value.
- Each group has 15 slots (0 to 15). Stop events when slot = 15 creates a global stop for all slots of a group.
- **Examples:** EventStartA(2); EventStopA(2); EventStartB(4,34, counter);

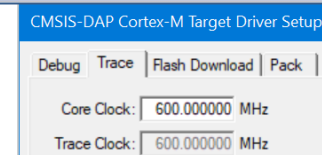### Add the EventStart and EventStop Events:

1. Stop the program if running and exit Debug mode.
2. Add this line near line 74: EventStartA(11);
3. Add this line near line 78: EventStopA(11);
4. Select File/Save All or click .

Set Core Clock: for Timing Measurements:

1. Select Target Options and select the Debug tab.
2. Select the Settings: icon to the right of this window.
3. Select the Trace tab.
4. Enter 600 MHz for Core Clock. Do not select Trace Enable which turns SWV on. EVR does not use SWV. μVision uses this Core Clock: setting to calculate timing values displayed in some windows.
5. Click OK twice to return to the main μVision menu.

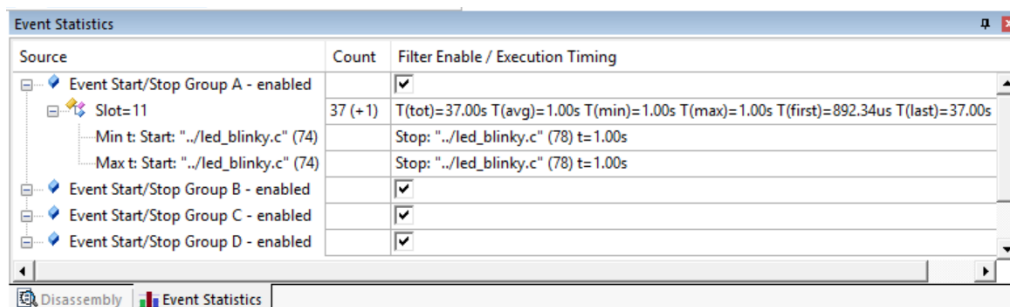### Build and RUN the program:

1. Rebuild the source files .
2. Enter Debug mode . Click on RUN .
3. Open the Event Statistics window from the toolbar:
4. Event Statistics will display timing data between the Start and Stop function calls shown below:
5. Event Group A is displayed, slot 11 as indicated. EventStartA is on source line 74. EventStopA is on line 78.
6. The execution time is 1.0 second. The total elapsed time for the execution run is 147.99 seconds.
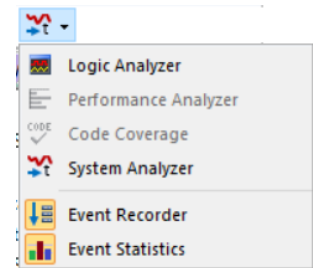
This makes it easy for you to determine these statistical values for locations in your sources.

TIP: If you are using a Keil ULINK*plus*, voltage, current and power consumption can be displayed.
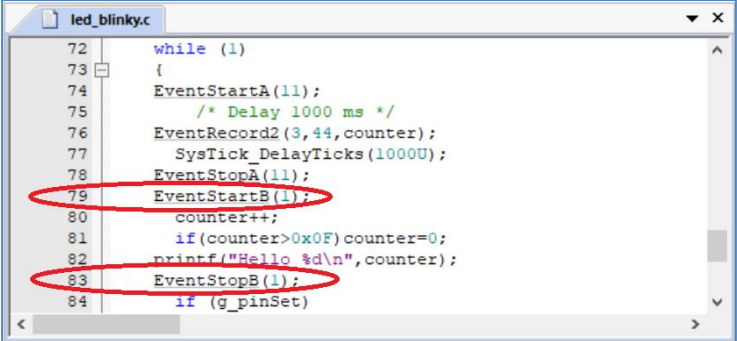
The next page shows how to add a second event.

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com

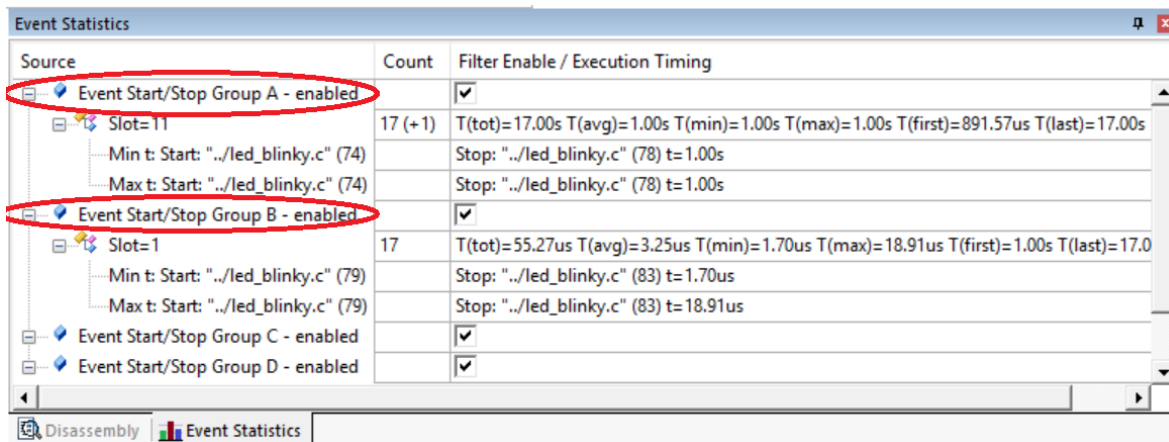## 6) Add a second pair of EventStart and EventStop Events:

1. Stop the program if it is running ⊗ and exit Debug mode. 🔍

2. In led_blinky.c:

3. Add this line near line 78: EventStartB(1);

4. Near line 82 add this line: EventStopB(1);

5. Select File/Save All or click 🗐 .

```
led_blinky.c                                          ▼ ×
72        while (1)
73  ⊟    {
74            EventStartA(11);
75                /* Delay 1000 ms */
76            EventRecord2(3,44,counter);
77              SysTick_DelayTicks(1000U);
78            EventStopA(11);
79            EventStartB(1);
80              counter++;
81              if(counter>0x0F)counter=0;
82            printf("Hello %d\n",counter);
83            EventStopB(1);
84              if (g_pinSet)
```

Build and RUN the program:

1. Rebuild the source files 🗓 .

2. Enter Debug mode 🔍 . Click on RUN 🗐 .

3. The Event Statistics will display information between the Start and Stop source lines as shown here:

4. This adds Event Group B, slot 1 as indicated. EventStartB is on Blinky.c source line 79. EventStopB is on line 83.

5. The execution time for Group A is 17.0 second.

6. The execution time for Group B is 55.27 µsec.

Source	Count	Filter Enable / Execution Timing
⊟ ◆ Event Start/Stop Group A - enabled		☑
⊟ 🕒 Slot=11	17 (+1)	T(tot)=17.00s T(avg)=1.00s T(min)=1.00s T(max)=1.00s T(first)=891.57us T(last)=17.00s
Min t: Start: "../led_blinky.c" (74)		Stop: "../led_blinky.c" (78) t=1.00s
Max t: Start: "../led_blinky.c" (74)		Stop: "../led_blinky.c" (78) t=1.00s
⊟ ◆ Event Start/Stop Group B - enabled		☑
⊟ 🕒 Slot=1	17	T(tot)=55.27us T(avg)=3.25us T(min)=1.70us T(max)=18.91us T(first)=1.00s T(last)=17.0
Min t: Start: "../led_blinky.c" (79)		Stop: "../led_blinky.c" (83) t=1.70us
Max t: Start: "../led_blinky.c" (79)		Stop: "../led_blinky.c" (83) t=18.91us
⊟ ◆ Event Start/Stop Group C - enabled		☑
⊟ ◆ Event Start/Stop Group D - enabled		☑

🔍 Disassembly | 📊 Event Statistics

The events are also displayed in the System Analyzer. Use of this window requires a ULINK*plus* or a ULINK*pro*. This provides a good method to determine timings relative to your source code. Event Statistics is easy to configure and interpret. You can create many more such Start and Stop events.

A complete explanation of this window is here: www.keil.com/support/man/docs/uv4/uv4_db_dbg_evr_stat.htm

To save information in Event Statistics see: www.keil.com/support/man/docs/uv4/uv4_cm_er.htm

Keil ULINK*plus*:

If you are using a Keil ULINK*plus*, you can see values for current, voltage and total power consumed during these times.

TIP: You can use Event Recorder to determine the duration and number of times an interrupt handler has occurred by adding EventStart and EvenStop tags. Or any other part of your source code. SWV can also display exceptions graphically in the System Analyzer. For SWV you must use a Keil ULINK*plus*, ULINK*pro* or a fast J-Link.

These features are described on the next few pages. A Keil ULINK*plus* is needed.

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com

1) Serial Wire Viewer (SWV): A Keil ULINK*plus* or a Keil ULINK*pro* is needed.

Serial Wire Viewer provides program information in real-time and is extremely useful in debugging programs. Serial Wire Viewer is used to display data reads and writes and exceptions which include interrupts and more.

The on-board LPC-Link does not currently support SWV with µVision. For SWV a ULINK2, ULINK*plus*, ULINK*pro* or a fast J-Link is required. ULINK*plus* adds Power Measurement and ULINK*pro* adds ETM Instructions trace. Both provide very fast SWV support and are recommended. In this tutorial we will now add a Keil ULINK*plus*.

Prerequisites:

µVision must be running and in Edit mode (not Debug mode). Your project must be loaded. Remove power from the board.

We will use the iled_blinky example and add a Keil ULINK*plus* debug adapter.

Create a new Target Selection:

1. Select the Target Option iled_blinky flexspi nor debug to use as the template:

2. Select Project/Manage/Project Items… or select:

3. In the Project Targets area, select NEW or press your keyboard INSERT key.

4. Enter **ULINKplus debug** and press Enter. Click OK to close this window.

5. In the Target Selector menu, select the **ULINKplus debug** selection you created:

6. Select Options for Target or ALT-F7 twice. Click on the Debug tab.

7. Select ULINKplus Debugger … as shown here:

8. If you are using a different debug adapter, choose it here now.
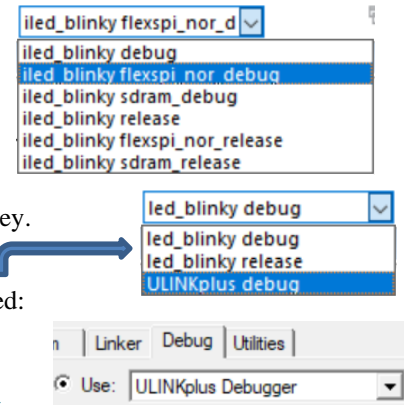
Connect and Test the ULINK*plus*:

1. Connect your debug adapter cable as shown here to J21 JTAG connector: You will need an adapter as shown and a 10 to 20 pin cable.

2. Remove jumpers J47 and J48 to disable the on-board LPC-Link2

3. Power the board with a USB cable to J41.

4. Connect a USB cable from the ULINK*plus* to your PC running µVision.

5. Click on Settings: Confirm SW is selected in the Port: box and not JTAG.

6. A valid SW Device *must* be displayed as shown below right.

7. **Trouble ?** If nothing or an error is displayed, check the SWD connections. Make sure the board is correctly powered. This *must* be solved before you can continue.

8. When completed, click OK twice to close the Options for Target windows.

9. Select File/Save All or click:

Verify the Flash Program Algorithm:

1. Select Options for Target or ALT-F7.

2. Select the Utilities tab. Select Settings:

3. Set RAM Size: to 0x00008000 as shown here:

4. This is the correct i.MXRT1060 algorithm.

5. If not selected, select Add and choose the correct algorithm as shown.

6. Click OK twice to return to the main µVision window.

7. Select File/Save All or click .

The new Debug Adapter, in this case a Keil ULINK*plus*, is now ready to use. First, we must enable the SWO data trace output.

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com

2) Configuring Serial Wire Viewer (SWV): Using a Keil ULINK*plus*:

1. µVision must be stopped and in Edit mode (not Debug mode). Hardware connected as shown on previous page.

A) Configure CCM_CCGR0 Clock Gate Register 0:

In led_blinky.c, a call to BOARD_InitBootClocks(); near line 61 turns off the trace_clk_enable in Clock Gate Register 0. This must be enabled in order for SWV to operate. There are two ways to accomplish this: Do one of these:

  1) Comment out the call to the function CLOCK_DisableClock: This is easiest.
 1. In clock_config.c, comment out CLOCK_DisableClock(kCLOCK_Trace); which is found at line 285.
  2) Modify led_blinky.c to configure the trace_clk_enable register.
  1. In led_blinky.c near line 19, add: **#define trace_clock_enable (*((volatile unsigned int *)(0x400FC068)))**
  2. Near line 59 right after BOARD_InitBootClocks();, add **trace_clock_enable=trace_clock_enable|0x00C00000;**

### B) Configure ARM_TRACE_SWO: IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_10:

1. This directs the SWO signal to the JTAG TDO pin 13. This is on the large 20 pin JTAG connector J21.

2. Select Options for Target 🪄 or ALT-F7 twice. Click on the Debug tab.

3. In the Initialization box will be a *.ini file listed. Click Edit… to display it in the source windows.

4. In this ini file, just before the LOAD command near line 86, add: **_WDWORD(0x401F80E4, 0x00000009);**

5. Select File/Save All. 🖫 This step is necessary to activate any .ini file.

C) Configure Serial Wire Viewer Target Driver Setup:

1. Select Options for Target 🪄 or ALT-F7 twice.

2. Click on the Debug tab. Click on the Settings: [Settings]

3. Select the Trace tab. The Target Driver Setup window below is displayed as shown below.

4. In Core Clock: enter 600 MHz and select Trace Enable. This value will be used for calculating timing values.

5. Unselect Use Core Clock and enter 120 MHz in Trace Clock:. The SWO clock speed will be calculated from this.

6. This value is crucial for a ULINK or J-Link as a UART is used. ULINK*pro* does not use this for the connection.

7. Confirm EXCTRC is selected. The enables exceptions (including interrupts) to be output to µVision for display.

8. SWV is now configured and ready to use.

9. Note the SWO Clock: box. µVision automatically selected 40 MHz as the SWO speed. Higher is better. 40 MHz is the highest possible in this setup with a Core Clock: of 600 MHz. A slower debug adapter will automatically divide by another ratio. SWO is the signal present at JTAG TDO pin 13 on the J21 connector.

10. Click on OK twice to return to the main µVision menu.

11. Select File/Save All or click 🖫.

12. Rebuild the source files 🛠.

TIP: For details determining Core Clock speed see: www.keil.com/appnotes/docs/apnt_297.asp

TIP: µVision uses Core Clock: value to calculate various timing displays even if SWV is not used. If the times displayed in various windows are incorrect, check these settings.
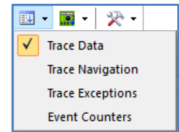
NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com
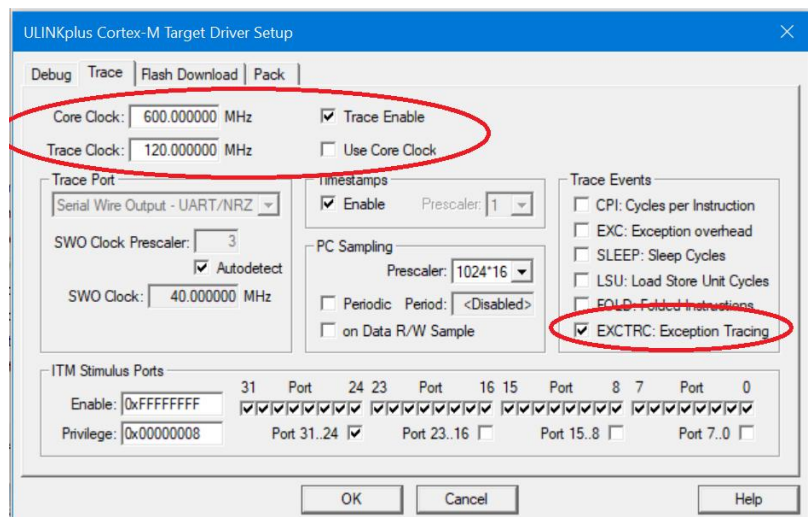
3) Displaying Exceptions and Interrupts using Serial Wire Viewer (SWV):

Trace Data Window:

The Instruction Trace window used by ULINK*plus* is shown below. The program must be stopped to update a Trace Data window unless using a Keil ULINK2. A ULINK2's SWV speed is minimal. A ULINK*plus* or ULINK*pro* are much faster.

1. Enter Debug mode. 🔍 Click on the RUN icon. 📑.
2. Open the Trace Data window by clicking on the small arrow beside the Trace icon shown here:
3. The Trace Data window opens. With a ULINK2, this is called Trace Records.
4. STOP ✖ the program and Exception 15 is shown below in Trace Data:
5. Displayed is Exception 15 (SYSTICK) with Entry, Exit and Return points.
6. Close the Instruction Trace window when you are done.

Entry: when the exception or interrupt is entered.

Exit: when the exception or interrupt exits.

Return: when all exceptions or interrupts exit. This indicates that no Cortex tail chaining is occurring.

Trouble ?: If you do not see Exceptions as shown and instead see either nothing or frames with strange data: the trace is not configured correctly. The most probable cause is the Core Clock: frequency is wrong. ITM frames 31 and 0 are the only valid ones. **Any other ITM frames displayed are bogus and usually indicate a wrong Core Clock value.** Make sure you entered the code lines shown on the previous page.

Core Clock Value: Check the global variable SystemCoreClock. See www.keil.com/appnotes/docs/apnt_297.asp Keil ULINK2, ULINK*plus* and J-Link need this value set accurately as it is the BAUD rate of a UART. ULINK*pro* needs it set accurately only for timing values. It extracts the clock from Manchester mode or the 4 bit + clock ETM trace.

This processor uses two clocks for trace: see the previous page for details. Core Clock: is used to calculate displayed timing values and Trace Clock: to calculate the SWO UART. ULINK*pro* uses Manchester mode hence speed is in the trace data.

Trace Exceptions Window:

1. Click on the RUN icon. 📑.
2. Open the Trace Exceptions window by clicking on the small arrow beside the Trace icon:
3. In this window, click on the Count column name to bring 15 SysTick into view as shown below.
4. Note the values are updating while the program runs. Unselect EXCTRC: box to stop the updates.
5. Note the various statistics displayed. Any other active exceptions will automatically be displayed.
6. *Period* = 324/323671 = 1 msec. The SysTick is generating an interrupt every 1 msec in this case.
7. See System Analyzer for graphical display of exceptions (including interrupts) using ULINK*plus* or ULINK*pro*.
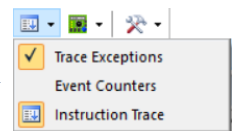
#	Name	Count	Total Time	Min Time In	Max Time In	Min Time Out	Max Time Out	First Time [s]	Last Time [s]
11	SVCall	0	0 s						
12	DebugMonitor	0	0 s						
14	PendSV	0	0 s						
15	SysTick	323671	129.778 ms	0 s	999.858 us	0 s	149.592 ms	0.00158462	324.00177242
16	ExtIRQ 0	0	0 s						
17	ExtIRQ 1	0	0 s						

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com

4) Using the Logic Analyzer (LA):

µVision has a graphical Logic Analyzer (LA) window. Up to four variables can be displayed in real-time using Serial Wire Viewer. Logic Analyzer data points occur when the events happen with each data write and not periodically like the Watch, Memory windows. If SWV overload occurs, turn off unneeded SWV features. You could sample the data writes with another variable and then display it. The idea is to reduce SWO traffic to prevent overflows and data loss.

The LA shares the comparators in CoreSight with the Watchpoints. They are mutually exclusive.

1. The project iled_blinky.uvprojx should still be open and is still in Debug mode and either running or stopped.
2. Right click on counter in led_blinky.c and select Add 'counter' to … and select Analyzer. The LA will open.

TIP: An error message saying counter cannot be added usually means SWV is not configured or counter is not in focus.

3. In the LA, click on Setup and set Max: in Display Range to 0x0F. Click on Close. The LA is now configured.
4. Run the program. ![icon] You can configure LA while the program is running.
5. counter will still be visible in Watch 1.
6. Adjust the Zoom OUT or the All icon in the LA to provide a suitable scale of about 23 seconds as shown above:

Trouble: If no waveform is displayed in the LA, exit and re-enter Debug mode ![icon] ![icon] to refresh the comparators.

7. Enable Amplitude, Signal Info and Cursor in the LA window.
8. Select Stop in the Update Screen to stop the LA from collecting data. The program keeps running.
9. Click on a spot in the LA and position the mouse to demonstrate the method used to measure timing values.
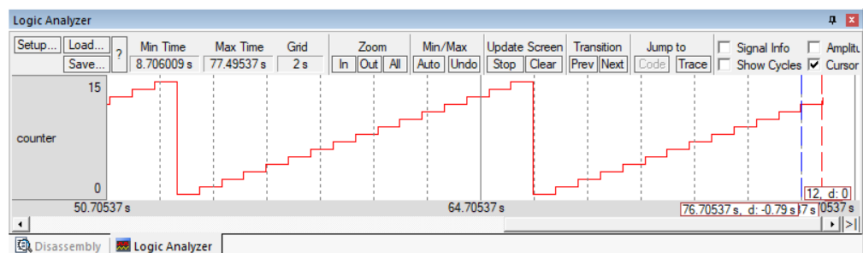
10. In this case, each step on the waveform is 1.0 seconds and the period is about 15 seconds.

TIP: When you enter a variable in the Logic Analyzer window, it is also displayed in the Trace Records (or Trace Data for ULINK*pro*). A ULINK2, ULINK*plus* or ULINK*pro* displays the Data Writes. J-Link does not.

ULINK*plus* and ULINK*pro* also displays any LA waveforms in System Analyzer window.

g_systickCounter:

1. Right click on g_systickCounter in led_blinky.c and select Add 'g_systickCounter' to … and select Analyzer.
2. In the LA, click on Setup and highlight g_systickCounter and set Max: in Display Range to 0x500. Click on Close.
3. Run the program. ![icon]
4. g_systickCounter will be visible as shown here: You can add counter2 to Watch 1 if you prefer.
5. When done, select Setup… and remove g_systickCounter. Leave counter active.
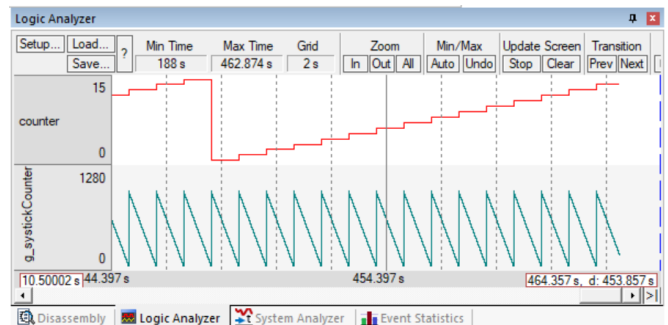
Trouble ?: If no waveform is displayed in the LA, exit and reenter Debug mode ![icon] ![icon] to refresh the comparators.

TIP: If the waveform of counter2 looks jagged, this can be from overloads in the SWO pin. Solutions are getting a faster debug adapter such as ULINK*plus* or ULINK*pro*. Turning off various SWV signals such as EXCTRC that are not needed can help. A Keil ULINK*pro* has the ability to receive SWV frames out the 4 bit + clock Trace Port instead of the 1 bit SWO pin like the other adapters which has significantly greater data throughput.

TIP: The Logic Analyzer can display static and global variables, structures and arrays. It can't see locals: make them static or global. To see peripheral registers, enter their physical addresses into the Logic Analyzer and read or write to them. Physical addresses can be entered as: *((unsigned long *)0x20000000).

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com

5) System Analyzer Window: Core Clock, Logic Analyzer and Exception Display:

System Analyzer can display much information depending on which Cortex-M processor and debug adapter you are using. Use a Keil ULINK*plus* to get the most out of System Analyzer. Data reads and writes, exception/interrupt events and more are displayed. ULINK*plus* and ULINK*pro* provide this information as well as a measured core clock value with ULINK*plus*.

In this example, Event Recorder frames, Logic Analyzer counter and SysTick interrupts are displayed.

Open the System Analyzer window:

1. Stop data collection with the Freeze Data icon:

2. Unselect the cursor icons to make the screen a bit cleaner.
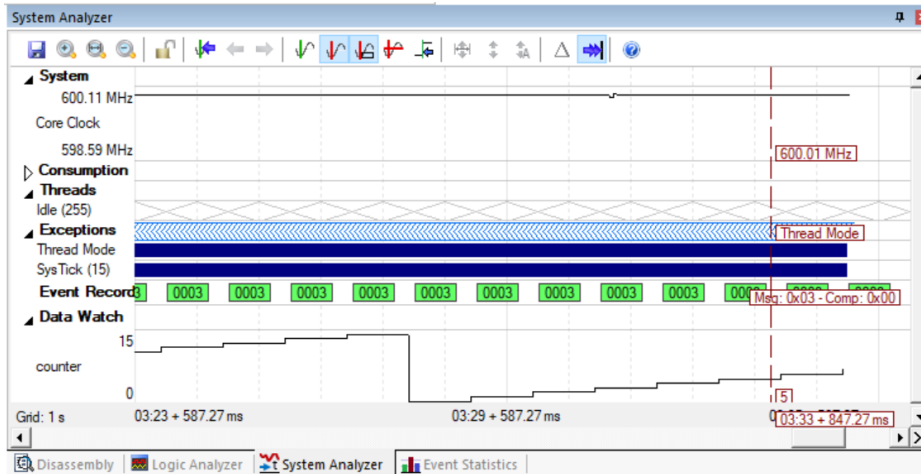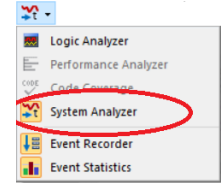
3. You will see the System Analyzer window with a series of 003 Event blocks as shown above:

4. Note the leading edge of the blocks corresponds to the increment of counter. This provides a link to your code.

5. Note SysTick interrupts are recorded and the Core Clock is measured.

6. Expand System Analyzer by clicking on a spot to create a set point and rotate your mouse wheel to expand at this point or use the expand icon:

7. You can measure the period between the interrupts and how long the handler took to execute. This is very useful to confirm interrupts are executing as designed.
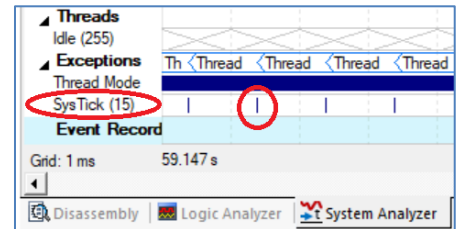
TIP: Use the column icon to align the waveforms.

Note: The Consumption rows are collapsed. This is where current and voltage waveforms will be displayed.

Where are the other Events and printf ?:

These events are there. In this simple program these events are occurring very close together. If you expand the X axis further, the events will be displayed as shown here:

You can activate the cursors and make various timing measurements.

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com
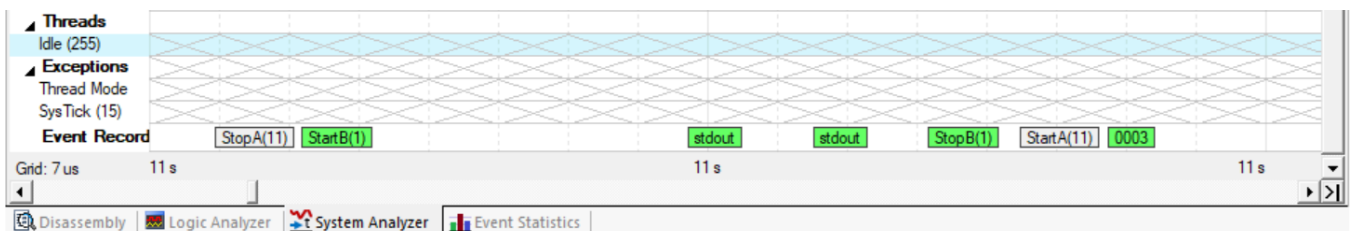
1) Power Measurement with Keil ULINK*plus*:
Connect and Configure ULINK*plus*:

An Arm Keil ULINK*plus* measures the CPU core clock and adds power measurement.

1. Stop the program if it is running ❌ and exit Debug mode. 🔴 Remove the board power.

Connecting a ULINKplus Power Connections:

1. Remove jumper J7. It is near button SW3. This will be our measurement point.
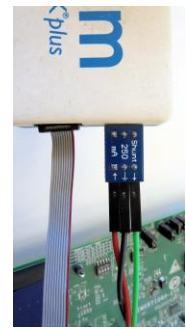
2. ULINK*plus* comes with a series of shunt resistor boards. Install the 250 mA as shown. ➡️

3. Connect the center ground pin on the shunt board to the pin 1 of J42 or any other ground.

4. Connect the two outside pins to jumper J7. The order of connection is not important at this time.
   See the photo shown on the right: ➡️

Configure ULINK*plus* with the Initialization File:

1. The UlinkPlus.ini file configures ULINK*plus*. It is executed when Debug mode is entered.

2. This file is UlinkPlus.ini and is found where you got this document on keil.com. It is also available here: C:\Keil_v5\ARM\ULINK\Templates\

3. Copy this file into C:\00MDK\iMXRT1060\demo_apps\led_blinky\mdk

4. Select Options for Target 🔷 or ALT-F7 twice.

5. Click on the Debug tab.

6. In the Initialization File box:, click Edit.. [Edit...] to open the existing .ini file.

7. In this ini file, just before the LOAD command, add this line: **include "UlinkPlus.ini"** or the actual name of the file you obtained.

8. Use File/Open and load UlinkPlus.ini into a source window.

9. Click OK to close the Target Options window and return to the main µVision window.

10. Click on the Configuration Wizard tab at the bottom of Debug_UlinkPlus.ini window.

11. Enable ULINK Power Measurement and select 1.00 Ohm as shown below:

12. Select File/Save All or click 💾. This step is necessary to activate any .ini file.
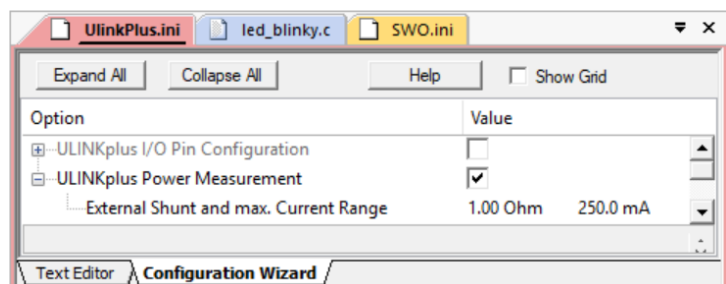
ULINK*plus* is now configured and ready to use.
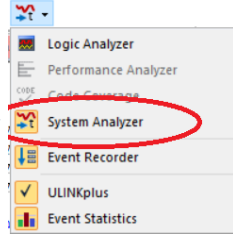
You can examine the Debug_UlinkPlus.ini file and see it contains additional features for I/O and a Toolbox window.

You can close this Toolbox window as we will not use it in this tutorial.

For complete documentation on ULINK*plus* and its features visit: www.keil.com/support/man/docs/ulinkplus/

Next Page: RUN the led_blinky program and measure its power consumption:

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com

Open System Analyzer:

1. Rebuild the source files [icon].

2. Enter Debug mode [icon]. Click on RUN [icon].

3. Open the System Analyzer window if it is not already open: ➡

4. Select to keep the waveform at the end: [icon]

5. You will have a window similar to the one below:

[menu: Logic Analyzer / Performance Analyzer / Code Coverage / System Analyzer / Event Recorder / ULINKplus / Event Statistics]

6. Stop the data collection: Click and change [unlock icon] to [lock icon]

7. Note the Current and Voltage waveforms. **TIP:** If Current has negative values, reverse the shunt wires.

8. The square wave in the current row is the LED sequencing.

TIP: The voltage and current Y axis numbers are adaptive and will change dynamically to suit the values displayed.

9. You can Zoom in or out with [icons] or your mouse wheel. First click in the waveform to set an anchor point.
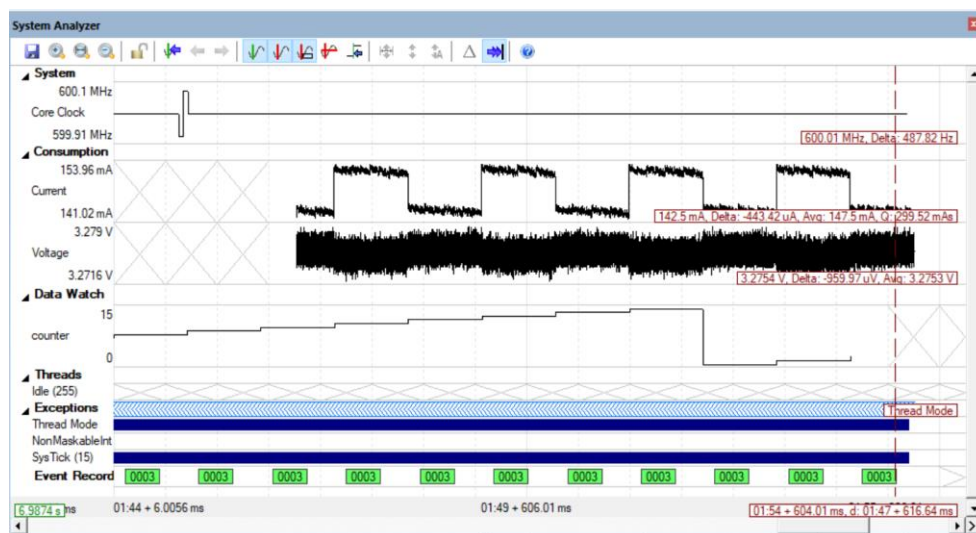
Expanded View:

You can relate the waveforms to your code using Event Recorder events as shown.

Note the SysTick timer exception events. If there are other exceptions and interrupts these will automatically added. This is an excellent method to see when your interrupts occur and how long the handlers take to execute.

System Core Clock: This is a measured value with Keil ULINK*plus* and not calculated from the system files.

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com
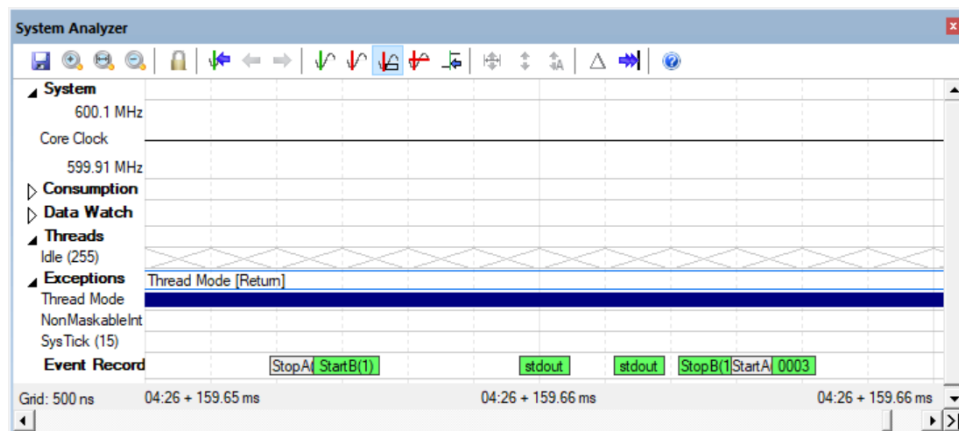
2) Event Statistics Window:

1. Open the Event Statistics window:

2. Note the various statistics displayed and updated in real-time. Timings, voltage, current and charges consumed are recorded according to Groups and Slots.

3. A description of the contents of this window are here:
   www.keil.com/support/man/docs/uv4/uv4_db_dbg_evr_stat.htm

4. These are created by the START and STOP function calls you put in your code.
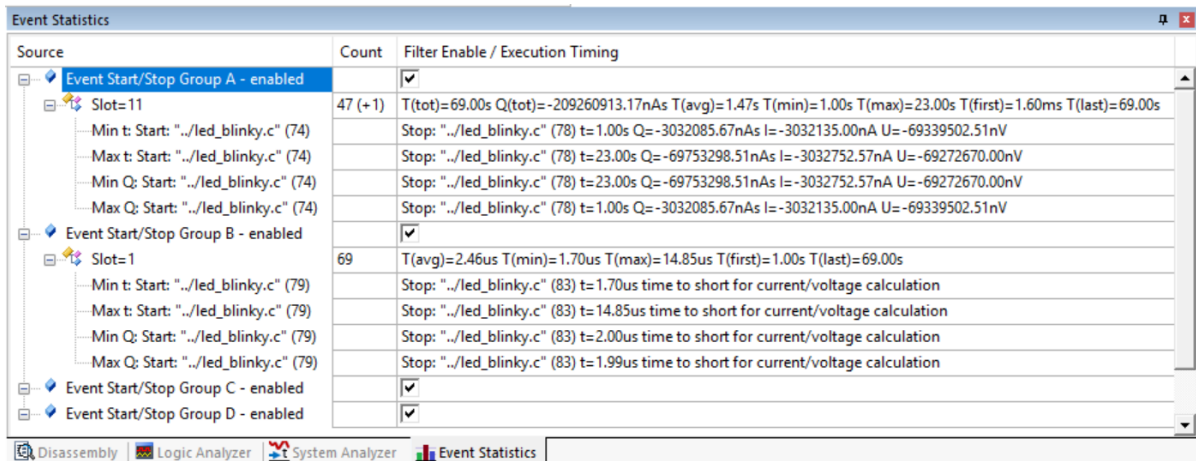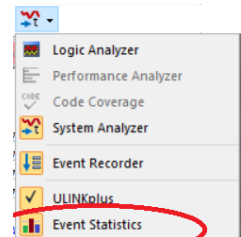
3) Energy Measurement without CoreSight DEBUG:

The CoreSight debug components consume some amount of power. This will be reflected in the System Analyzer window. Since it is not likely debug will be used in a released product, you can measure the power without Debug.

You will not be able to relate the waveforms with your code. You will need to do something else to do this. Toggling GPIO ports from within your code is one method that is useful.

The effects will be most pronounced during processor Sleep and WAIT states when the current drops to very low values.

1. When not in Debug mode: select the small arrow beside the debug icon and the Energy Measurement without debug will display as shown here:

2. Select the green Energy icon.

3. µVision will start Energy Measurement. Click RUN to start the program.

4. Statistics Analyzer will display the waveform as shown below: Any features needing CoreSight will not be available.
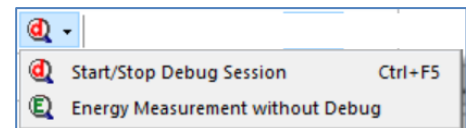
This is the end of the exercises.

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com
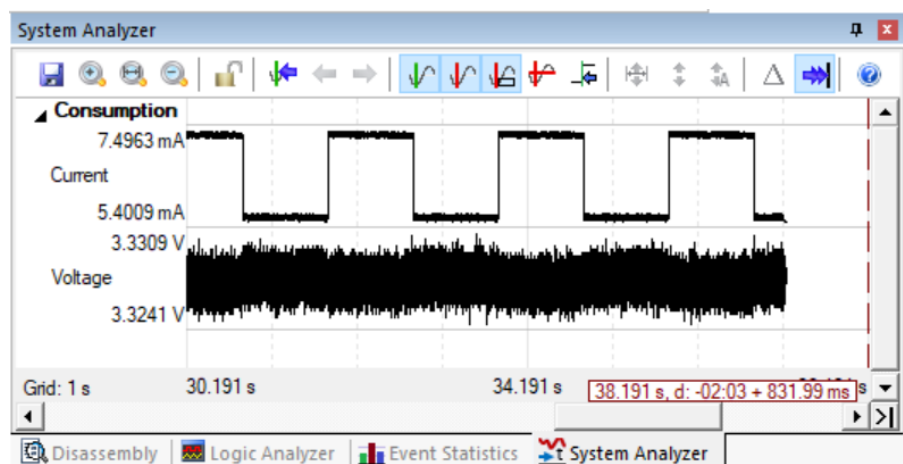
1) External Debug Adapter Types:

A) LPC-Link2 (This has both CMSIS-DAP and J-Link Lite modes):

A small processor located on the target board acts as the debug adapter. On the MIMXRT1060-EVK board, this extra processor is U382. Connect a USB cable to connector J41 to use LPC-LINK2. J47, J48 and J49 must be populated. CMSIS-DAP provides run control debugging, Flash and RAM programming and Watchpoints. Hardware breakpoints can be set/unset while the program runs. Reads and writes in the Watch, Memory and Peripheral windows are updated in real-time as well as the RTX System kernel awareness viewer. At this time, the LPC-Link2 version of CMSIS-DAP does not support Serial Wire Viewer (SWV). LPC-Link2 also has a J-Link Lite mode. Both can be installed with NXO utility LPCScrypt. The MIMXRT1060-EVK is preprogrammed with a LPC-LINK2 which is CMSIS-DAP compliant.

You can incorporate a CMSIS-DAP design on your own custom target boards. For documents and software go to https://github.com/ARM-software/CMSIS_5 Click on the CMSIS folder and then on DAP/Firmware.

This document was prepared using MDK 5.31.

A) Keil ULINK2/ME:

This is a hardware JTAG/SWD debugger. It connects to various connectors found on boards populated with ARM processors. With NXP Cortex-M3, M4 and M7 processors, ULINK2 supports Serial Wire Viewer (SWV). ULINK-ME is equivalent to a ULINK2.

ULINK2

B) Keil ULINK*pro*:

ULINK*pro* is Keil's most advanced debug adapter. ULINK*pro* provides Serial Wire Viewer (SWV) and ETM Instruction Trace support for NXP Cortex-M3, Cortex-M4 and Cortex-M7 processors. Code Coverage, Performance Analysis and Execution Profiling are then provided using ETM. ULINK*pro* programs the Flash memories very fast. ARM DS-MDK can use a ULINK*pro* or a ULINK*pro* D to debug NXP i.MX Cortex-A processors.

C) Keil ULINK*plus*:

ULINK*plus* is Keil's latest debug adapter. Power Measurement is its main feature. It also measures the clock frequency and has very fast SWV performance. It displays exceptions (interrupts) and has facilities for testing. ULINK*plus* provides very fast Serial Wire Viewer (SWV) performance. It has been tested on this board up to 48 MHz. It might go higher.

ULINK*pro*

D) Segger J-Link:

µVision supports various J-link adapters.

1. Select the Target Options icon [icon].
2. Click on the Debug tab. Select J-Link/J-Trace in the Use: box:
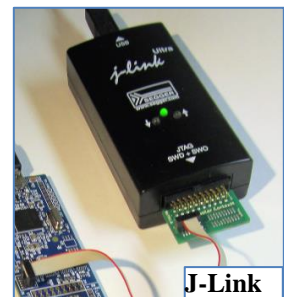
ULINK*plus*

External Debug Adapter USB Connections: *Important !*

LPC-Link2:
1. Use J41 USB connector.
2. J47, J48 and J49 must be populated.

ULINK2, ULINK*pro*, ULINK*plus* or J-Link:
1. Use JTAG connector J21 with a suitable adapter depending on your debug adapter.
2. Remove jumpers J47 and J48 to disable the on-board LPCLink2.
3. Connect J41 to supply board power.

J-Link

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com

2) External Debug Adapter Configuration:

Once you know which debug adapter you will be using, you can quickly select and test it in µVision.

It is easy to configure µVision for a variety of supported Debug Adapters.

Prerequisites:

µVision must be running and in Edit mode (not Debug mode). Your project must be loaded.

We will use the Blinky example and add LPC-Link2 CMSIS-DAP support. You can choose your own debug adapter by using these instructions.
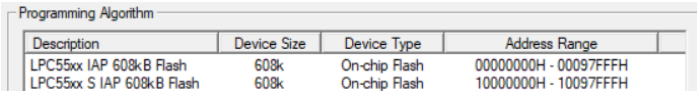
Create a new Target Selection:

1. Select a Target Option LPC54114 Flash to use as the template adapter: `LPC54114 Flash`
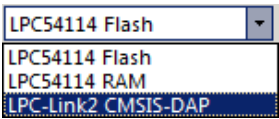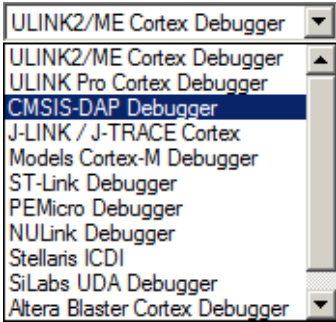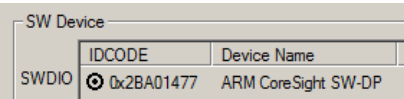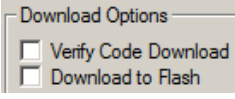
2. Select Project/Manage/Project Items… or select:

3. In the Project Targets area, select NEW or press your keyboard INSERT key.

4. Enter **LPC-Link2 CMSIS-DAP** and press Enter. Click OK to close this window.

5. In the Target Selector menu, select the **LPC-Link2 CMSIS-DAP** selection you just created:

6. Select Options for Target or ALT-F7 twice. Click on the Debug tab.

7. Select CMSIS-DAP Debugger… as shown here:

8. If you are using a different debug adapter, choose it here now.

Test the Debug Adapter:

1. Connect your debug adapter as described on the previous page.

2. Power the board appropriately.

3. In this case, for CMSIS-DAP, connect USB to the USB Link connector.

4. Click on Settings: Confirm SW is selected in the Port: box and not JTAG.

5. A valid SW Device *must* be displayed as shown here.

6. **Trouble ?** If nothing or an error is displayed, check the LPCXpresso firmware programming and USB connections. See the previous page for instructions. This *must* be solved before you can continue.

7. If using LPC-Link2, unselect Verify Code Download as shown here:

8. If this selected, an error may occur during the Flash programming process.

9. When completed, click OK twice to close the Options for Target windows.

10. Select File/Save All or click:

Verify the Flash Program Algorithm: This step is optional. It is useful if Flash programming fails.

1. Select Options for Target or ALT-F7. Select the Utilities tab.

2. Select Settings:

3. The window that opens will display the correct algorithm. This is selected automatically according to the processor selected in the Device tab.

4. This is the correct algorithms for the LPC55S69.

5. If it is not visible, select Add to add it.

6. Click OK twice to return to the main µVision window.

7. Select File/Save All or click .

The new Debug Adapter is now ready to use.

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com

3) Document Resources: See www.keil.com/NXP

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.keil.com/books/armbooks.asp

3. µVision contains a window titled Books. Many documents including data sheets are located there.

4. A list of Arm processors is located at: www.arm.com/products/processors/cortex-m/index.php

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. MOOC: Massive Open Online Class: University of Texas: http://users.ece.utexas.edu/~valvano/

Application Notes:

1. *NEW!* ARM Compiler Qualification Kit: Compiler Safety Certification: www.keil.com/safety

2. Using Cortex-M3 and Cortex-M4 Fault Exceptions   www.keil.com/appnotes/files/apnt209.pdf

3. CAN Primer: www.keil.com/appnotes/files/apnt_247.pdf

4. Segger emWin GUIBuilder with µVision™   www.keil.com/appnotes/files/apnt_234.pdf

5. Porting mbed Project to Keil MDK™   www.keil.com/appnotes/docs/apnt_207.asp

6. MDK-ARM™ Compiler Optimizations   www.keil.com/appnotes/docs/apnt_202.asp

7. GNU tools (GCC) for use with µVision   https://launchpad.net/gcc-arm-embedded

8. RTX CMSIS-RTOS Download   www.keil.com/demo/eval/rtx.htm

9. Barrier Instructions   http://infocenter.arm.com/help/topic/com.arm.doc.dai0321a/index.html

10. Lazy Stacking on the Cortex-M4:   www.arm.com and search for DAI0298A

11. Cortex Debug Connectors: www.keil.com/coresight/coresight-connectors

12. Sending ITM printf to external Windows applications: http://www.keil.com/appnotes/docs/apnt_240.asp

13. FlexMemory configuration using MDK   www.keil.com/appnotes/files/apnt220.pdf

14. Sending ITM printf to external Windows applications: www.keil.com/appnotes/docs/apnt_240.asp

15. *NEW!* Migrating Cortex-M3/M4 to Cortex-M7 processors: www.keil.com/appnotes/docs/apnt_270.asp

16. *NEW!* ARMv8-M Architecture Technical Overview   https://community.arm.com/docs/DOC-10896

17. *NEW!* Using TrustZone on ARMv8-M with Keil MDK: www.keil.com/appnotes/docs/apnt_291.asp

18. *NEW!* Determining Cortex-M CPU Frequency using SWV   www.keil.com/appnotes/docs/apnt_297.asp

Useful ARM Websites:

1. CMSIS: https://github.com/ARM-software/CMSIS_5   www.arm.com/cmsis/   www.keil.com/cmsis

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

3. ARM Developer: https://developer.arm.com/

4. ARM University Program: www.arm.com/university.   Email: university@arm.com

5. mbed™: http://mbed.org

> **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 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

4) Keil Products and Contact Information: See www.keil.com/NXP

Keil Microcontroller Development Kit (MDK-ARM™)

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

   For the latest MDK details see: www.keil.com/mdk5/selector/

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

USB-JTAG adapter (for Flash programming too)

- **ULINK2** -*(ULINK2 and ME - SWV only – no ETM)*   **ULINK-ME** – Electrically equal to ULINK2.
- *New* **ULINKplus**- Cortex-M*x* High performance SWV & power measurement.
- **ULINKpro** - Cortex-M*x* SWV & ETM trace.
- **ULINKpro D** – The same as a ULINKpro without ETM support. It can be used with µVision or DS-5.

For Serial Wire Viewer (SWV), a ULINK2 or a J-Link is needed. For ETM support, a ULINK*pro* is needed.

All ULINK products support MTB (Micro Trace Buffer) with NXP Cortex-M0+.

The Keil RTX RTOS is provided with a BSD or Apache 2.0 license. This makes it free. All source code is provided.

See https://github.com/ARM-software/CMSIS_5 and www.keil.com/gsg.
RTX documentation is free. www.keil.com/pack/doc/CMSIS/RTOS/html/
For the feature list see: www.keil.com/RTX

Keil provides free DSP libraries for the entire Arm Cortex processor family.

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 useful.

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

Call Keil Sales for special university pricing. Go to www.arm.com/university

Keil supports many other NXP processors including ARM7™and ARM9™
series processors. See the Keil Device Database® on www.keil.com/dd for the
complete list of NXP support. Also see www.keil.com/NXP

NXP i.MX Cortex-A processors are supported by ARM DS-MDK™.
www.keil.com/ds-mdk or 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

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

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

Forums: www.keil.com/forum   http://community.arm.com/groups/tools/content   https://developer.arm.com/

NXP Cortex-M7 Lab with the i.MX RT1060 EVK evaluation board   www.keil.com