

The latest version of this document is here: www.keil.com/appnotes/docs/apnt_321.asp

Introduction:

The purpose of this tutorial is to introduce you to Keil® µVision® Event Recorder (EVR). Event Recorder is used to annotate your source code to emit messages to be used and displayed in µVision debugger. EVR uses the Arm® CoreSight™ Debug Access Port (DAP). ST-LINK/V2, any Keil ULINK, CMSIS-DAP and J-Link supports this link. Serial Wire Viewer (SWV) is not available with Cortex-M0/M0+ processors. SWV is another method to send valuable debugging information to µVision. Event Recorder works on any Arm Cortex-M device with or without SWV.

This tutorial uses the Nucleo-G071RB evaluation board. It is easily adaptable to other ST evaluation or your custom boards.

Keil MDK is provided free for ST Cortex-M0 and Cortex-M0+ processors. MDK supports all ST Cortex-M processors.

STM32CubeMX provides projects in MDK µVision format. These projects are readily imported into µVision.

Keil MDK-Lite™ is a free evaluation version that limits code size to 32 Kbytes for Cortex-M processors. The addition of a valid license number will turn MDK into a full commercial version. MDK is free for STM32 Cortex®-M0 and Cortex-M0+ processors. See www.keil.com/ST or contact Keil Sales for more information regarding obtaining evaluation licenses.

This tutorial describes practical methods of using µVision Event Recorder. It is in a hands-on format.

Complete Event Recorder documentation is located here: www.keil.com/pack/doc/compiler/EventRecorder/html/

For certified Arm tools and software see: www.keil.com/safety Arm FuSa RTS Run Time System: www.keil.com/fusa-rtts

STM32G0 Online Course:

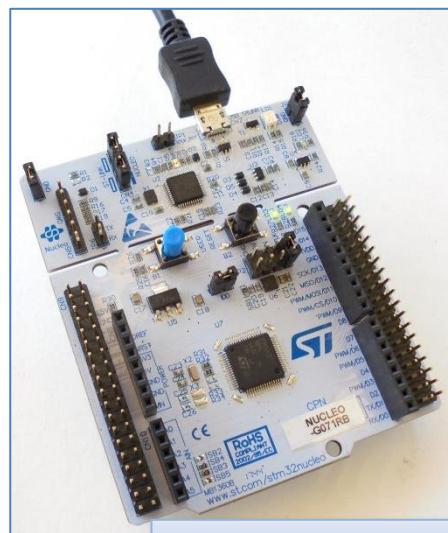
www.st.com/content/st_com/en/support/learning/stm32-education/stm32-moocs/STM32G0_workshop_MOOC.html

Hardware and Software Requirements:

1. Any ST evaluation board with an on-board ST-LINK/V2. Includes Discovery, Nucleo and EVAL boards. This tutorial uses a Nucleo-G071RB. Other debug adapters can also be used including Keil ULINK™ series, ST-LINK or a J-Link. It is easy to modify this tutorial for use with any STM32 board including custom boards.
2. Keil MDK V 5.28a or later. The free ST Cortex-M0/M0+ version (MDK-ST) was used for this document. No license is needed: you can also use the free evaluation (MDK-Lite) or any licensed copy of MDK.
3. The provided Blinky exercise works with any version of MDK including the free evaluation MDK-Lite.
4. The correct Software Pack for your ST processor is installed.
5. A valid MDK 5 project. One is provided with this appnote. You can also use any example provided by STM32Cube, a Software Pack example or your own project. Event Recorder is easy to use.

This document details these features:

1. Real-time Read and Write to memory locations for Watch, Memory and Peripherals windows. These are non-intrusive to your program. No CPU cycles are stolen. No instrumentation code is added to your source. These windows update while your program runs.
2. Four Hardware Breakpoints (they can be set/unset on-the-fly).
3. A *printf* utility using Event Recorder. A UART is not used.
4. Annotation for your code using Keil Event Viewer.
5. RTOS Viewers: Kernel awareness windows for the Keil RTX RTOS update while your program is running. RTX5 is open source. FreeRTOS is now supported: www.keil.com/pr/article/1280.htm



Nucleo-G071RB board with the on-board ST-LINK/V2

TIP: Keil has an adapter to connect a ULINK to the board SWD header. See page 17.

1. Keil Software Overview: MDK 5:	3
2. Keil MDK Core Software Download and Installation:	3
3. Getting Started Guide MDK 5:	3
4. STMicroelectronics Evaluation Boards:	3
5. Methods to Obtain Debug Information from Cortex-M Processors:	3
i. DAP Debug Access Port:	3
ii. Serial Wire Viewer (SWV):	3
iii. Embedded Trace Macrocell (ETM) and Micro-Trace Buffer (MTB):	3
6. CoreSight Definitions:	4
7. μ Vision Software Pack and Examples Download and Install Process:	5
8. Install the ST-LINK V2 USB Drivers and Update the ST-LINK Firmware:	6
9. Testing the ST-LINK/V2 Connection:	6
10. <i>STM32G0</i> example: NUCLEO STM32G0 board:	7
11. STM32G0 Hardware Breakpoints:	7
12. Handy μ Vision Hints and Tips:	7
13. Watch window:	8
14. Memory window:	8
15. System Viewer (SV): Peripheral Views:	9
16. printf with STM32G0 Using Event Recorder:	10
17. Using Event Recorder to Instrument Your code:	11
18. Determining Relative Timing Values:	12
19. Filtering the Event Recorder Window:	12
20. Event Statistics: Statistical information about your program:	13
21. System Analyzer Window: Displaying various program:	15
22. Time Stamps Configuration as used in Event Recorder:	16
23. Keil ULINK <i>plus</i> Operation:	17
i. Connecting ULINK <i>plus</i> to STM32G0 and other ST boards:	17
ii. Select ULINK <i>plus</i> as New Debug Adapter:	18
iii. Final Configuration and testing ULINK <i>plus</i> :	18
24. Power Measurement using ULINK <i>plus</i> :	19
25. System Analyzer Displaying Power Measurement Data:	20
26. Event Statistics Displaying Power Measurement Data:	21
27. Energy Measurement without CoreSight DEBUG:	21
28. Using Non-Initialized RAM:	22
29. Document Resources:	23
30. Keil Products and contact information:	24

Notes on using this document:

1. The latest version of this document is located here: www.keil.com/appnotes/docs/apnt_321.asp
2. ARM MDK 5.30 and Software Pack Keil::STM32G0_DFP 1.2.0 were used in the exercises in this document.
3. Later versions can be used. It is possible to use MDK 5.28a or 5.29.
4. The onboard ST-LINK/V2 debug adapter interfaces very well with Keil μ Vision including SWV.
5. If your STM32 becomes locked up see Read Protection: www.st.com/en/development-tools/stm32cubeprog.html


1) Keil Software Overview: MDK 5

MDK 5 uses Software Packs to distribute processor specific software, examples and middleware. MDK 5 Core is first installed and you then download the Software Packs you require from the web. They can also be imported manually. A Software pack contains headers, Flash programming, CMSIS files, documents and more in a regular .zip file. The extension is then renamed .pack. You need not wait for the next version of MDK or install patches to get the latest processor specific files.

STM32CubeMX provides a project in MDK 5 format consistent with Software Packs. Keil Middleware supports STM32.

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

2) Keil MDK Core Software Download and Installation:

1. Obtain a license for MDK-ST: from the Keil website. www2.keil.com/stmicroelectronics-stm32/mdk
2. Save the Product Serial Number (PSN) displayed. You will need it after downloading MDK Core.
3. Click on Download MDK-Core by clicking on this icon: 
4. Install MDK into the default folder. You can install into any folder or drive, but this lab uses the default C:\Keil_v5.
5. Return to the URL above for installation and license activation instructions under the heading “Guides”.
6. You do not need any debug adapters: just the Nucleo or Discovery board, a USB cable and MDK 5 installed.
7. For the power display exercises, you need a Keil ULINK_{plus} debug adapter. www.keil.com/mdk5/ulink/ulinkplus
8. You can also use the evaluation version (MDK-Lite) for this lab. No license is then needed.

3) Getting Started Guide MDK 5: Obtain this useful Getting Started book here: www.keil.com/gsg/.

4) STMicroelectronics Evaluation boards:

This tutorial can support any ST evaluation boards or a custom board: Nucleo, Discovery or Evaluation (EVAL). A board needs an on-board ST-LINK/V2 or an external debug adapter such as any Keil ULINK.

This tutorial uses a Nucleo-G071RB evaluation board. For other boards, you need to select the appropriate Software Pack and example. Serial Wire Viewer (SWV) is not supported with a Cortex-M0/M0+ or Cortex-M23. Only the Arm Corex-M3, M4, M7 and Cortex-M33 processors have SWV.

To use any ST-LINK/V2 equipped board, download the appropriate Pack. Use the STM32G0 example and modify it.

5) There are three main methods to transfer debugging data to and from Arm STM32 Cortex processors: *These generally do not require CPU cycles to operate.*

DAP Debug Access Port:

DAP allows reads and writes to your target while the program is running through the JTAG or SWD ports. It is nearly always non-intrusive. No code stubs are needed in your program. Watch, Memory, System Viewer (peripherals) and RTX Threads and System windows use this feature as well as Event Recorder. DAP reads are periodic and set by μ Vision. Do not confuse DAP with CMSIS-DAP which is a debug adapter standard from ARM. All Cortex-M processors have DAP.

Serial Wire Viewer (SWV): (SWV is not on Cortex-M0 or Cortex-M0+)

Serial Wire Viewer (SWV) displays PC Samples, Exceptions (includes interrupts), data reads and writes, ITM (printf), CPU counters and timestamps. This information comes from the ARM CoreSight™ debug module integrated into the STM32. SWV does not steal any CPU cycles and is completely non-intrusive. (except for the ITM Debug printf Viewer). SWV data is output on the 1 bit SWO (Serial Wire Output) pin on the debug connector. It is also possible to output SWV on the 4-bit ETM trace port if your processor is so equipped. SWV is not available on the Cortex-M0/M0+ or Cortex-M23. It is available only on Cortex-M3, M4, M7 and M33.

Up to four variables can be displayed graphically in the μ Vision Logic Analyzer.

Embedded Trace Macrocell (ETM): (includes Code Coverage and Performance Analysis)

ETM records and displays all instructions that were executed. This is useful for debugging program flow problems such as “going into the weeds” and “how did I get here?”. ETM requires a Keil ULINK™_{pro} and a 20 pin ETM connector such as on STM32756G_EVAL. Code Coverage and Performance Analysis is provided by ETM. ETM is not explored in this tutorial.

A Cortex-M0+ can have MTB Micro Trace Buffer. Trace frames are stored in a selected area of the internal RAM. Check your data sheet to see if your Arm processor is equipped with MTB. STM32G0 does not have MTB implemented.

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

Cortex-M0 and Cortex-M0+ have only features 2) through 4) plus 11, 12 and 13 implemented. Cortex-M3/M4/M7/M33 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 ST 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 μ Vision Cortex-M Target Driver Setup. See page 4, 2nd picture. The SWJ box must be selected in ULINK2/ME or ULINK pro . Serial Wire Viewer (SWV) must use SWD because the JTAG signal TDIO shares the same pin as SWO. The SWV data normally comes out the SWO pin.
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. μ Vision uses the DAP to update memory, watch and RTOS kernel awareness windows in real-time while the processor is running. You can also modify variable values on the fly. No CPU cycles are used, the program can be running and no source code stubs are needed. You do not need to configure or activate DAP. μ Vision configures DAP when you select a function that uses it. Do not confuse this with CMSIS-DAP which is an on-board debug adapter standard.
5. **SWV:** Serial Wire Viewer: A trace capability providing display of reads, writes, exceptions, PC Samples and printf. SWV and SWO are only on Cortex-M3, M4 and M7 processors. They are not on Cortex-M0 or M0+.
6. **SWO:** Serial Wire Output: SWV frames usually come out this one pin output. It shares the JTAG signal TDIO.
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 μ Vision, 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. μ Vision 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.
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 RAM is used for an instruction trace buffer. Only on Cortex-M0+ processors. STM32 Cortex-M4 and Cortex-M7 processors provide ETM trace instead.
12. **Hardware Breakpoints:** The Cortex-M0+ has 2 breakpoints. The Cortex-M3, M4 and M7 have 6. These can be set/unset on-the-fly without stopping the processor. They are no skid: they do not execute the instruction they are set on when a match occurs.
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. Keil documents also refer to these as Access Breakpoints.
14. **ELF/DWARF:** The ARM compiler produces an .axf file which is ELF/DWARF compliant. μ Vision can load similar compiler output such as from GCC with all debug information visible. You can also use GCC as your compiler of choice in μ Vision. Search for Arm GCC on developer.arm.com

7) µVision Software Pack and Examples Download and Install Process:



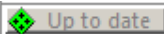

Software Packs contain files needed for your project such as header, flash programming and example files. For complete information see www.keil.com/pack/doc/CMSIS/Pack/html/

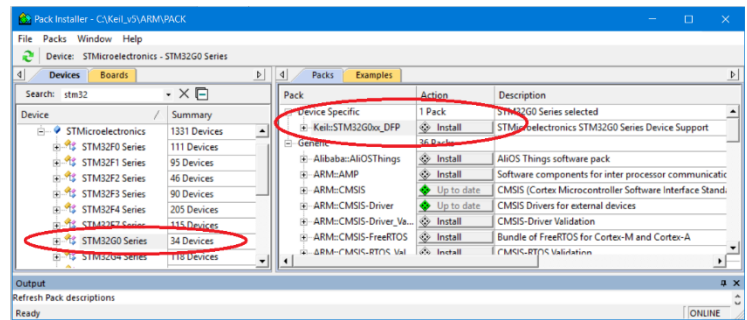
These instructions reference a Nucleo-G071RB board. If you are using a different board, install the appropriate Pack. If you already have the Pack and an example installed for your board or processor, you can skip this section.

Packs can be installed using the µVision pack Installer from the internet or with local .pack files.


The provided example is very simple and needs minimum effort to port to other processors and/or boards.

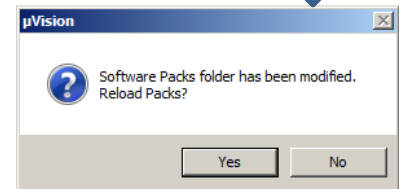
1) Start µVision and open Pack Installer (PI): After the first MDK install is complete and if you are connected to the Internet, µVision and Software Packs will automatically startup. Otherwise, follow Steps 1 and 2 below.

1. Connect your computer to the Internet. This is normally needed to download the Software Packs.
2. Start µVision by clicking on its desktop icon. 
3. Open the Pack Installer by clicking on its icon:  A Pack Installer Welcome screen might open. Read and close it.
4. Pack Installer opens: Select the Devices tab. Type **stm32g** in the Search box to filter the listings: You can also select your processor under the Boards tab.
5. Select STM32G0 Series as shown here:
6. Select the Packs tab.
7. Click Install opposite Keil::STM32G0xx_DFP:
8. The Pack will now install to your PC.
9. **Note:** “ONLINE” is displayed at the bottom right. If “OFFLINE” is displayed, connect to the Internet to continue.
10. The Pack's status will then be indicated by the “Up to date” icon: 
11. Close the Pack Installer. If a dialog box opens stating a Pack folder has been updated, click YES to reload.
12. You can open PI any time by clicking on its icon. 



TIP: What you select in the left side of the Pack Installer in the Devices or Boards tabs, determines what is displayed on the right side in the Packs and Examples tabs.

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 the Provided Example:

Currently, an example program for the STM32G0 is not available from the Pack. Obtain the example from where you got this document. This is a very simple example to help understand and modify how the program works.

TIP: Other examples, templates and working code can be obtained from STM32CubeMX.

1. Create a folder C:\00MDK\ST\EVRexample\
2. Copy the example into this folder using Microsoft Explorer. Unzip it as necessary.
3. The STM32G0 example is now located in C:\00MDK\ST\EVRexample\
4. The project name is stm32go.uvprojx.

Packs:

DFP: Device Family Pack: Contains header files, programming algorithms, examples, documents and much more.

BSP: Board Support Package. Provides board specific code for hardware items. A BSP can be inside a DFP.

8) Install the ST-LINK/V2 USB Drivers and Update the ST-LINK Firmware:

ST-LINK/V2 USB drivers initially must be installed manually. Windows may attempt to install them but this might not work.




1) Install the ST-LINK USB Drivers: *This step normally needs to be done just once !*

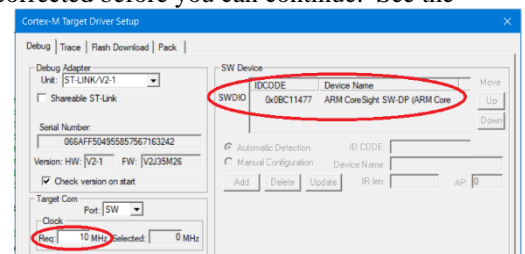
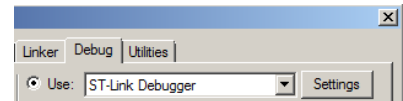
1. Do **not** have a Nucleo or Discovery board USB port connected to your PC at this time.
2. The USB drivers must be installed manually by executing dpinst_amd64.exe or dpinst_x86. This file is found in C:\Keil_v5\ARM\STLink\USBDriver\. Double-click on this file. The drivers will install normally.
3. Plug in the Nucleo or Discovery board to USB_STLINK CN2. The USB drivers will now finish installing normally.

2) Upgrading the ST-LINK/V2 Firmware: The ST-LINK/V2 firmware updater utility ST-LinkUpgrade.exe is located here: C:\Keil_v5\ARM\STLink\. It is a good idea to upgrade the ST-LINK firmware to match the version of μ Vision you are using. Navigate to this file on your computer and double click it to execute it. It will check and report the current firmware version. If you experience trouble, especially with Serial Wire Viewer, update to the latest ST firmware and USB drivers.

TIP: If your board has an old version of ST-LINK, μ Vision might ask you to upgrade it. This is normally a good idea. This way the firmware in your board matches the ST DLL inside μ Vision for best performance.

9) Testing the ST-LINK/V2 Connection: (Optional but this is a worthy test)

1. Start μ Vision  if it is not already running.
2. Connect the Nucleo or Discovery board USB ST-LINK to your PC.
3. If the ST-LINK USB drivers are installed correctly, you should hear the usual USB connected dual-tone. If not, you might have to install the drivers manually. See the directions above.
4. The red LED LD1 (for ST-LINK/V2 standby) and green LD3 (5 volt indicator) will light:
5. Select Project/Open Project.
6. Select the project C:\00MDK\ST\EVRexample\stm32go.uvprojx. (or any valid project)
7. Select Target Options  or ALT-F7 and select the Debug tab:
8. Select ST-Link Debugger: 
9. Click on Settings: and the window below opens: If an IDCODE and Device name is displayed, ST-LINK is working. If you do not see this, you must rectify the issue before you can continue.
10. Set the Clock REQ: box to 10 MHz. This is the maximum for ST-LINK. This affects Event Recorder performance.
11. A number in the SN: box means μ Vision is connected to the ST-LINK adapter. SW Device means μ Vision is connected to the processor via ST-LINK/V2.
12. If nothing or an error is displayed in this SW Device box, this **must** be corrected before you can continue. See the instructions above: Install the ST-LINK USB Drivers:
13. Once you see a proper display, your ST-LINK USB drivers are installed properly. Click OK twice to exit the Target Driver Setup windows. Continue to the next page.



TIP: To refresh the SW Device box, in the Port: box select JTAG and then select SW again. You can also exit then re-enter this window.

TIP: Cortex-M0 or Cortex-M0+ processors do not support the JTAG mode.

LED LD1 indication: (might not be applicable for all boards)

LED is blinking RED: the start of USB enumeration with the PC is taking place but not yet completed.

LED is RED: USB communication between the PC and ST-LINK/V2 is established. μ Vision is not connected.

LED is GREEN: μ Vision is now connected in Debug mode and the last communication was successful.






LED is blinking GREEN/RED: data is actively being exchanged between the target and μ Vision.

LED is off, except for a brief RED flash while entering Debug mode and a brief flash when clicking on RUN happens when the SWV trace is enabled in μ Vision.

No Led: ST-LINK/V2 communication with the target or μ Vision has failed. Cycle the board power to restart.

10) STM32G0 example: Nucleo STM32G0 board:

We will now connect the Keil MDK development system to the Nucleo board using the built-in ST-LINK/V2 debug adapter.

1. Start μ Vision by clicking on its desktop icon.  Connect your PC to the USB ST-LINK connector CN2.
2. Select Project/Open Project. Open C:\00MDK\ST\EVRExample\stm32g0.uvprojx
3. By default, the ST-LINK is selected. If this is the first time you have run μ Vision and the Nucleo or Discovery board, you might need to install the USB drivers. See the configuration instructions on the previous page.
4. Compile the source files by clicking on the Rebuild icon. . You can also use the Build icon beside it.
5. Enter Debug mode by clicking on the Debug icon.  Select OK if the Evaluation Mode box appears. The Flash will be programmed. Flash programming progress will be indicated in the Output Window.
6. Click on the RUN icon.  Note: you stop the program with the STOP icon. 

The LD4 green LED BLINKS under program control ! LD3 green Power LED is ON.
Led LD1 will blink red and green indicating traffic over ST-LINK/V2.



Now you know how to compile a program, program it into the STM32 Flash and run it !

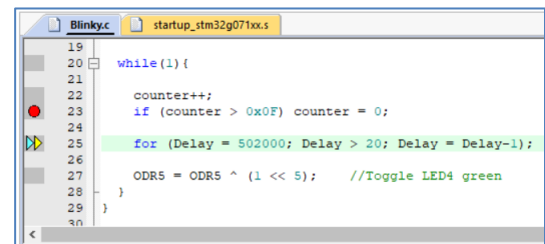
You may have to make minor modifications to use this example with a different processor and/or board.

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

11) STM32G0 Hardware Breakpoints:

The STM32G0 has four Arm CoreSight hardware breakpoints. Other Cortex-M processors have six. Some Cortex-M0 have two. Hardware breakpoints do not modify your code since CoreSight hardware comparators are used. They can be set/unset while your program runs. This is an important feature. If you set too many breakpoints, μ Vision will warn you.

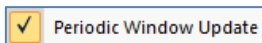
1. Click on the RUN icon if the program is halted. 
2. In the Blinky.c source file, click inside the while() loop on a grey block on the left side as shown: The gray blocks indicate that there is source or assembly code here. Breakpoints are visible in source windows as well as the Disassembly window.
3. A red circle will appear as shown here and the program will stop:
4. Unselect the breakpoint. Click on the RUN icon. 



TIP: To view breakpoints and Watchpoints, click Debug/Breakpoints or press Ctrl-B. You can temporarily unselect/select existing breakpoints, create or delete them in this window.

TIP: An Arm hardware breakpoint does not execute the instruction it is set to. Arm CoreSight breakpoints are no-skid. Your instructions in Flash are not substituted or modified. Your Flash contents are not changed. There is no intrusion to hardware or your program by Arm hardware breakpoints. These are rather important features for efficient software development.

12) Some Handy μ Vision Hints and Tips:


1. **STM32CubeMX** provides compatible μ Vision projects. There are many example projects using μ Vision available.
2. **Periodic Window Update:** Many μ Vision windows update while the program is running. If a window updates only when you stop the program: make sure Periodic Window Update is enabled. This is selected in Debug mode by selecting View in the main μ Vision menu. This window is enabled by default. It can be useful to turn this off if the DAP throughput with Event Recorder is too high and frames are lost. 
3. **Set the ST-LINK/V2 Clock to its highest value.** See page 6. In this case the highest is 10 MHz. For faster DAP operation, try higher or use a Keil ULINKpro or ULINKplus (to 15 MHz and beyond) or a suitable Segger J-Link.

13) Watch Window:

Watch windows will display variables in real-time as your program is running. You can display global and static variables and structures but not local variables since they are not always in focus. Memory and peripheral locations can also be displayed.

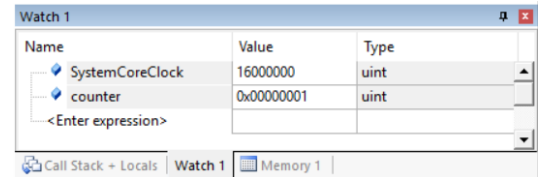
The CMSIS global variable `SystemCoreClock` contains the speed of the CPU. It is declared and set in `system_stm32g0xx.c`. Note this is a calculated value and not measured. A *ULINKplus* can measure the actual frequency on a Cortex-M3 or higher.

SystemCoreClock:

1. Click on the RUN icon.  You can configure the Watch and Memory windows while your program is running.
2. Select View/Watch Windows/Watch 1 if Watch 1 is not already open.
3. Double click on <Enter expression> and enter `SystemCoreClock` and press the Enter key.
4. A hexadecimal value will appear. Right click on the variable name and unselect Hexadecimal Display.
5. The CPU frequency will now display in Hertz. 16 MHz in this case.

Note: counter: A global variable was created in `Blinky.c`:

6. In `Blinky.c`, right click on the variable `counter`.
7. Select Add counter to ... and select Watch 1.
8. `counter` will display and will increment from 0 through 0x0F:



TIP: You can add variables to Watch and Memory windows when the program is running or halted. Stop the program to remove a Watch variable. You can modify a variable in the Memory window when stopped or running. See below.

Modify counter:

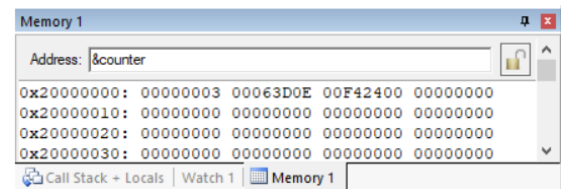
If a value in Watch is not changing too rapidly it is possible to modify a value “on-the-fly”

1. While the program is running, double-click on `counter` and enter a value and press Enter.
2. `counter` will be updated. If `counter` is changing too fast, use a Memory window instead or stop the program.

14) Memory Window:

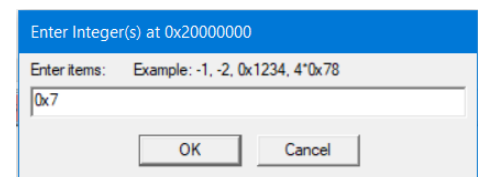
The Memory windows will display updated variable values in real-time. It is also possible to modify values into the Memory window while the program is running. This is not intrusive to your program.

1. In `Blinky.c`, right click on the variable `counter` and select Add counter to ... and select Memory 1. Memory 1 will open if needed and `counter` will be displayed as shown here:
2. Right click in the memory window and select Unsigned/Int.
3. `counter` will increment to 0x0F and will update in real-time.
4. Note the value of counter is displaying its value in Memory 1 address column 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 now.
5. Add an ampersand “&” in front of the variable name and press Enter. The physical address is shown (0x2000_0000). This address might be different depending on memory and compiler settings.



Modify counter:

1. You can modify `counter` or any changeable memory location in the Memory window with a right-click with the mouse cursor over a memory field and select Modify Memory.
2. Enter the new variable value as shown here and press Enter.



TIP: Note some values of counter might be skipped if it changes quickly because the Watch and Memory windows are updated periodically and not when values actually change.

TIP: These variables must be global, static or raw addresses such as `*((unsigned long *)0x20000000)`. They can also be a peripheral register memory or a CoreSight address as they are all memory mapped in Cortex-M processors.

15) System Viewer (SV):

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

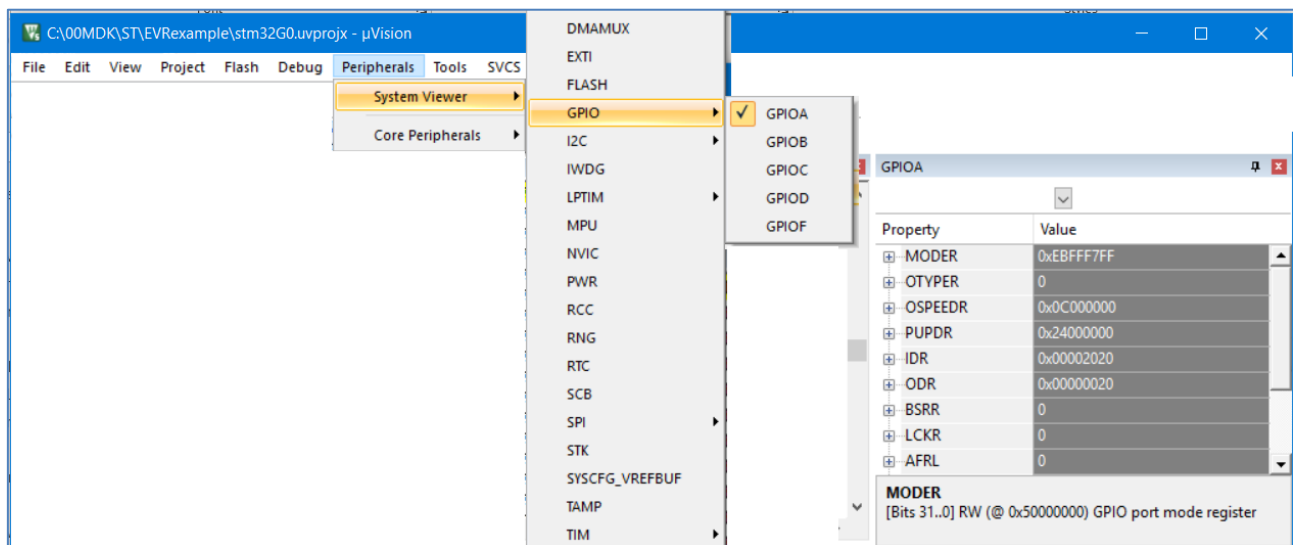
It is possible to create your own custom System Viewer for specified peripherals:



www.keil.com/support/man/docs/uv4/uv4_db_dbg_scvd_viewer.htm

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

GPIO Port A:



2. Select Peripherals/System Viewer, GPIO and then GPIOA as shown here:
3. The window GPIOA opens. Select MODER:
4. Note values changing in IDR and ODR as the LED blinks.
5. Note at the bottom of GPIOA, the physical address of MODER and a description are displayed. This is an easy method to find such physical addresses rather than searching in the device datasheet.
6. Expand one of the registers to see the individual registers or bits if applicable.




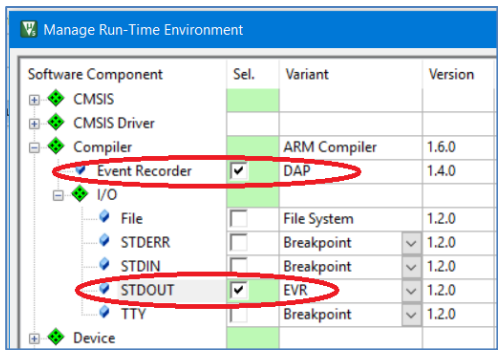
7. These values are updated periodically, not when the value changes.
8. You can modify register values while the program is running or stopped if the register parameters allows it.
9. These windows use the same CoreSight DAP technology as the Watch and Memory windows. Do not confuse this DAP (Debug Access Port) with CMSIS-DAP which is an Arm Open Source debug adapter protocol.
10. Stop the program  and exit Debug mode. 

16) *printf* for STM32G0 Cortex-M0 Using Event Recorder: (no UART is used)


Event Recorder (EVR) is a new μ Vision feature that can be used to instrument your code. Keil RTX5, CMSIS-RTOS2 FreeRTOS and Keil Middleware are already instrumented with EVR. EVR also provides a *printf* utility using the DAP read/write abilities of the Debug Access Port. A UART is not used. This can be used with any STM32 Cortex processors. DAP is the same technology used in Watch, Memory and Peripheral windows.

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




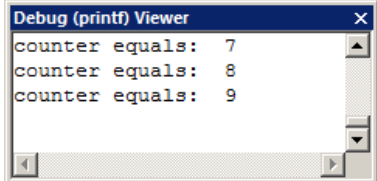

Configure Event Recorder:

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. Click OK to close this window.
7. `retarget_io.c`, `EventRecorder.c` and `EventRecorderConf.h` are added under the Compiler group in the Project window.
8. Right click near the top of `Blinky.c`, at line 3 and select Insert "#include" and select #include "EventRecorder.h".
9. At the beginning of the `main()` function near line 13, add this line: `EventRecorderInitialize (EventRecordAll, 1);`
10. Right after that line, add this line near line 14: `EventRecorderStart ();`

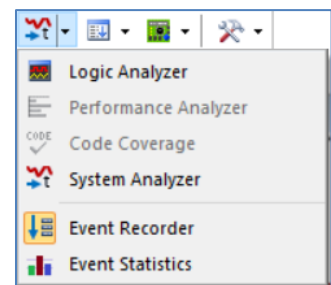
Add a *printf* statement to Blinky.c:



1. In `Blinky.c` add `#include "stdio.h"` near the top of the file near line 4.
2. In `Blinky.c`, near line 26 just after the `if (counter>....` line, add this line: `printf("counter equals: %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. Right click in this window to view some options available.
6. Open the Event Recorder window: 
7. Information about the *printf* statements are displayed as shown below:

TIP: If you get a Hard Fault error, you must either select MicroLIB *or* add some heap in `startup_stm32g071xx.s` or the correct `startup.s` file for your processor. Try adding a 200 byte heap. MicroLIB results in a smaller executable size.






Event Recorder				
Enable <input checked="" type="checkbox"/>   Mark: <input type="checkbox"/> All Operations <input type="checkbox"/> Stopped				
Event	Time (sec)	Component	Event Property	Value
0	0.00000000	EvCtrl	EventRecorderInitialize	Restart Count = 1
1	0.00002538	EvCtrl	EventRecorderStart	
2	0.00030137	STDIO	stdout	0x63,0x6F,0x75,0x6E,0x74,0x65,0x72,0x20
3	0.00036256	STDIO	stdout	0x65,0x71,0x75,0x61,0x6C,0x73,0x3A,0x20
4	0.00040394	STDIO	stdout	0x31,0x0A,0x00,0x00,0x00,0x00,0x00,0x00
5	0.45066644	STDIO	stdout	0x63,0x6F,0x75,0x6E,0x74,0x65,0x72,0x20
6	0.45072763	STDIO	stdout	0x65,0x71,0x75,0x61,0x6C,0x73,0x3A,0x20
7	0.45076900	STDIO	stdout	0x32,0x0A,0x00,0x00,0x00,0x00,0x00,0x00
8	0.90103150	STDIO	stdout	0x63,0x6F,0x75,0x6E,0x74,0x65,0x72,0x20
9	0.90109269	STDIO	stdout	0x65,0x71,0x75,0x61,0x6C,0x73,0x3A,0x20
10	0.90113406	STDIO	stdout	0x33,0x0A,0x00,0x00,0x00,0x00,0x00,0x00
11	1.35139912	STDIO	stdout	0x63,0x6F,0x75,0x6E,0x74,0x65,0x72,0x20

17) Using EventRecord to Instrument Your Code:

You can instrument your code with three basic function calls: EventRecord, EventRecord2 and EventRecord4.

They provide variable data length, fixed 2 or 4 byte values respectively. A simple example will show how this works.


Add an EventRecord Event:



1. Stop the program if it is running  and exit Debug mode. .
2. Add this line near line 23 right after the while(1) statement: EventRecord2(3, 44, counter);
3. Select File/Save All or click .

Build and RUN the program:

1. Rebuild the source files .
2. Enter Debug mode . Click on RUN .
3. Examine the Event Recorder window as shown below:

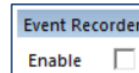
Explanation:

1. Note the result of EventRecord2 with an ID of 3 in the Event Recorder window below at Event 2,6 and 10.
2. In addition, in the Value column is displayed the value 0x44 and the value of the global variable counter.
3. A time stamp is included. In this case, this is derived from the SysTick counter. You can use other time sources such as a STM32 timer configured by STM32CubeMX if SysTick is used by other programs such as an RTOS.
4. It is possible to add additional annotations to this window using the Component Viewer in µVision. See www.keil.com/pack/doc/compiler/EventRecorder/html/cv_use.html
5. You can have many EventRecord events with many bytes of data, with two 32 bits(used here) or three 32 bit values.
6. For complete documentation, visit www.keil.com/pack/doc/compiler/EventRecorder/html/er_overview.html
7. Depending on the amount of Event Recorder data being passed through the debug adapter, you might have to increase the DAP Clock rate. You can use a higher speed adapter such as a Keil ULINKpro, ULINKplus or a fast J-Link.
8. You can save these frames with the SAVE icon. 

Event Recorder				
Enable	<input checked="" type="checkbox"/>			Mark: <input type="checkbox"/> All Operations <input type="checkbox"/> Stopped
Event	Time (sec)	Component	Event Property	Value
0	0.00000000	EvCtrl	EventRecorderInitialize	Restart Count = 1
1	0.00002538	EvCtrl	EventRecorderStart	
2	0.00005500		id=0x0003	0x00000044, 0x00000001
3	0.00032975	STDIO	stdout	0x63, 0x6F, 0x75, 0x6E, 0x74, 0x65, 0x72, 0x20
4	0.00039094	STDIO	stdout	0x65, 0x71, 0x75, 0x61, 0x6C, 0x73, 0x3A, 0x20
5	0.00043221	STDIO	stdout	0x31, 0x8A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
6	0.45045525		id=0x0003	0x00000044, 0x00000002
7	0.45072519	STDIO	stdout	0x63, 0x6F, 0x75, 0x6E, 0x74, 0x65, 0x72, 0x20
8	0.45078438	STDIO	stdout	0x65, 0x71, 0x75, 0x61, 0x6C, 0x73, 0x3A, 0x20
9	0.45082575	STDIO	stdout	0x32, 0x0A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
10	0.90084869		id=0x0003	0x00000044, 0x00000003

18) Determining Relative Timing Values:

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



Event	Time (sec)	Component	Event Property	Value
2564	145.06745037		id=0x0003	0x00000044,0x00000003
2565	145.06747906	EvStat	StartB(1)	File="Blinky.c"(22)
2566	145.06774656	STDIO	stdout	0x00000044,0x00000072,0x20
2567	145.06780775	STDIO	stdout	0x00000044,0x0000003A,0x20
2568	145.06784913	STDIO	stdout	0x00000044,0x00000000,0x00
2569	145.06789050	EvStat	StopA(11)	File="Blinky.c"(24)
2570	145.51789969	EvStat	StopB(1)	File="Blinky.c"(26)

4. Position your mouse pointer on the Time (sec) column on the next stdout frame.
5. A box will open displaying the elapsed time as shown here: One printf statement took about 61.19 μ sec to execute.
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.

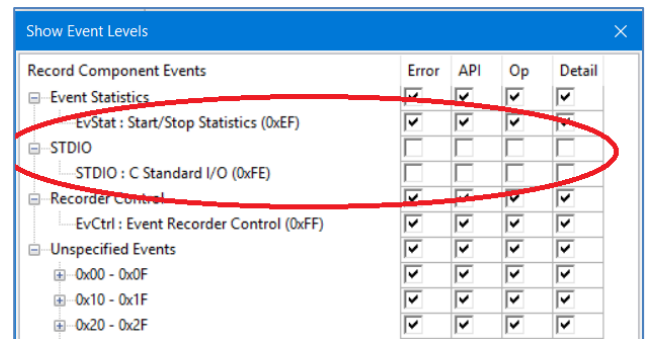
Event Time = 4.52302269
Delta Time = +0.00006119
Reference Time = 4.52296150

Note: 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.

19) 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 any of the printf frames as shown below right:



In this case, you only need to unselect the Op column. The other frames do not exist in our simple example.

Event	Time (sec)	Component	Event Property	Value
0	99.52349363		id=0x0003	0x00000044,0x0000000E
1	99.97381550		id=0x0003	0x00000044,0x0000000F
2	100.42413481		id=0x0003	0x00000044,0x00000010
3	100.87443994		id=0x0003	0x00000044,0x00000001
4	101.32474425		id=0x0003	0x00000044,0x00000002
5	101.77505113		id=0x0003	0x00000044,0x00000003
6	102.22535544		id=0x0003	0x00000044,0x00000004
7	102.67565975		id=0x0003	0x00000044,0x00000005
8	103.12596669		id=0x0003	0x00000044,0x00000006
9	103.57627100		id=0x0003	0x00000044,0x00000007

Saving the Filter Settings:

These settings are not saved during a RESET or restarting of μ Vision. To save, Event Recorder must be given a non-initialized memory area in the processor RAM. See page 22 and also:




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

20) 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 will also include voltage, current and total charge (Q) consumed. Individual and aggregated 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.


- **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 represent a global stop for all slots of a group.
- **Examples:** EventStartA(2); EventStopA(2); EventStartB(4,34, counter);

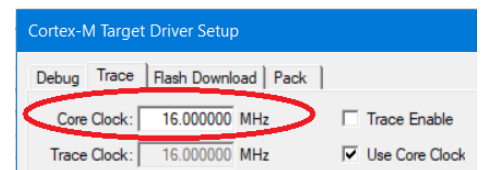
Add the EventStart and EventStop Events:

1. Stop the program if running  and exit Debug mode. 
2. Add this line near line 24: EventStartA(11);
3. Near line 28 add this line: EventStopA(11);
4. Select File/Save All or click .





```
22 while(1){
23     EventRecord2(3,44,counter);
24     EventStartA(11);
25     counter++;
26     if (counter > 0x0F) counter = 0;
27     printf("counter equals: %d\n", counter);
28     EventStopA(11);
29     for (Delay = 502000; Delay > 20; Delay = Delay-1);
30
31     ODR5 = ODR5 ^ (1 << 5); //Toggle LED4 green
32 }
```

Set Core Clock: for Timing Measurements:

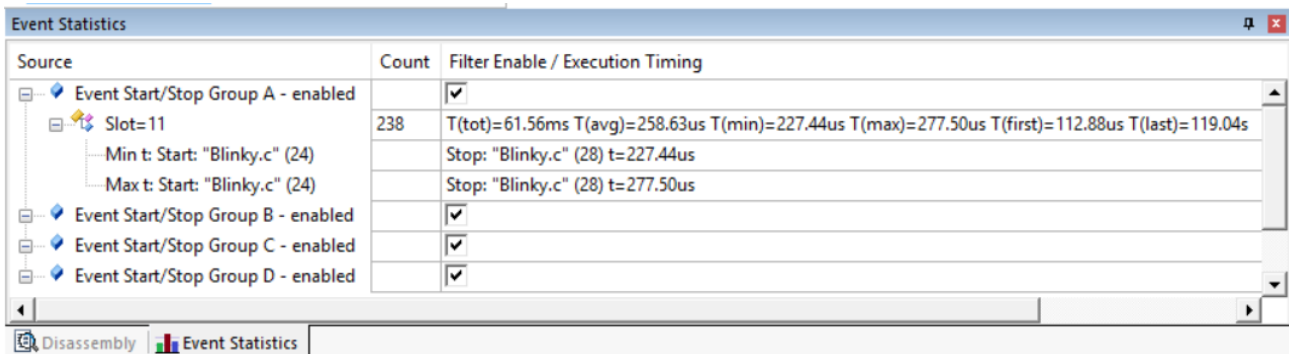
1. Select Target Options  and select the Debug tab.
2. Select the Settings: icon to the right of this window.
3. Select the Trace tab.
4. Enter 16 MHz for Core Clock. Do not select Trace Enable.
5. Click OK twice to return to the main µVision menu.
6. µVision uses this setting for many displays of timing values.



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 data between the Start and Stop function calls shown below:
5. This is for Event Group A, slot 11 as indicated. EventStartA is on Blinky.c source line 24. EventStopA is on line 28.
6. The execution elapsed is 227 µs.




This makes it easy for you to determine these statistical values to locations in your sources.






Source	Count	Filter Enable / Execution Timing
Event Start/Stop Group A - enabled		<input checked="" type="checkbox"/>
Slot=11	238	T(tot)=61.56ms T(avg)=258.63us T(min)=227.44us T(max)=277.50us T(first)=112.88us T(last)=119.04s
Min t: Start: "Blinky.c" (24)		Stop: "Blinky.c" (28) t=227.44us
Max t: Start: "Blinky.c" (24)		Stop: "Blinky.c" (28) t=277.50us
Event Start/Stop Group B - enabled		<input checked="" type="checkbox"/>
Event Start/Stop Group C - enabled		<input checked="" type="checkbox"/>
Event Start/Stop Group D - enabled		<input checked="" type="checkbox"/>

The next page shows how to add a second event. After that is discussed the additional power consumption information you get when you use a Keil ULINK_{plus} debug adapter.

Add a second pair of EventStart and EventStop Events:

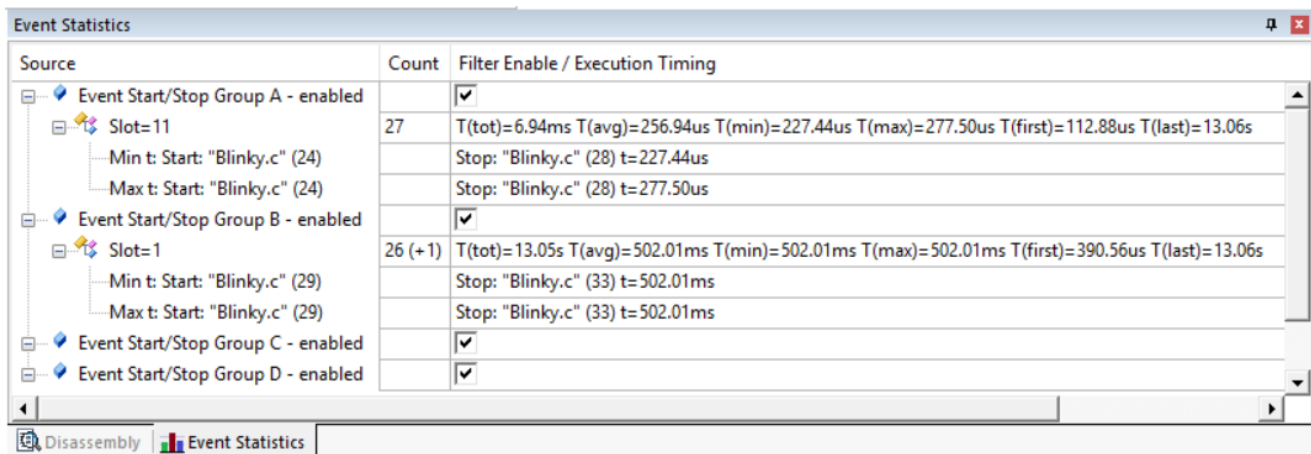
1. Stop the program if it is running  and exit Debug mode. 
2. Add this line near line 29: EventStartB(1);
3. Near line 33 add this line: EventStopB(1);
4. Select File/Save All or click .

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 29. EventStop is on line 33.
5. Group A has an elapsed time of 227 μ s as shown.
6. Group B has an elapsed time of 502 msec as shown.

```
22 while(1){
23     EventRecord2(3,44,counter);
24     EventStartA(11);
25     counter++;
26     if (counter > 0x0F) counter = 0;
27     printf("counter equals: %d\n", counter);
28     EventStopA(11);
29     EventStartB(1);
30     for (Delay = 502000; Delay > 20; Delay = Delay-1);
31
32     ODR5 = ODR5 ^ (1 << 5);    //Toggle LED4 green
33     EventStopB(1);
34 }
```

This provides a good method to determine timings in your source code Event Statistics is easy to configure and interpret.



Source	Count	Filter Enable / Execution Timing
Event Start/Stop Group A - enabled		<input checked="" type="checkbox"/>
Slot=11	27	T(tot)=6.94ms T(avg)=256.94us T(min)=227.44us T(max)=277.50us T(first)=112.88us T(last)=13.06s
Min t: Start: "Blinky.c" (24)		Stop: "Blinky.c" (28) t=227.44us
Max t: Start: "Blinky.c" (24)		Stop: "Blinky.c" (28) t=277.50us
Event Start/Stop Group B - enabled		<input checked="" type="checkbox"/>
Slot=1	26 (+1)	T(tot)=13.05s T(avg)=502.01ms T(min)=502.01ms T(max)=502.01ms T(first)=390.56us T(last)=13.06s
Min t: Start: "Blinky.c" (29)		Stop: "Blinky.c" (33) t=502.01ms
Max t: Start: "Blinky.c" (29)		Stop: "Blinky.c" (33) t=502.01ms
Event Start/Stop Group C - enabled		<input checked="" type="checkbox"/>
Event Start/Stop Group D - enabled		<input checked="" type="checkbox"/>

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 ULINKplus:

If you are using a Keil ULINKplus, you will see values for current, voltage and total power consumed during these times.

TIP: You can use Event Recorder to determine the time and number of times an interrupt handler has occurred. Or any other part of your source code.

Using Non-initialized RAM for Event Recorder:

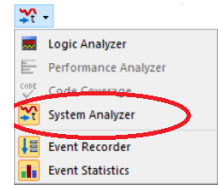
See page 22 on how to allocate noint RAM for Event Recorder. This can help with some unusual issues with Event Recorder. The EVR data will be saved through a processor reset. It is recommended to use this technique.

21) System Analyzer Window:

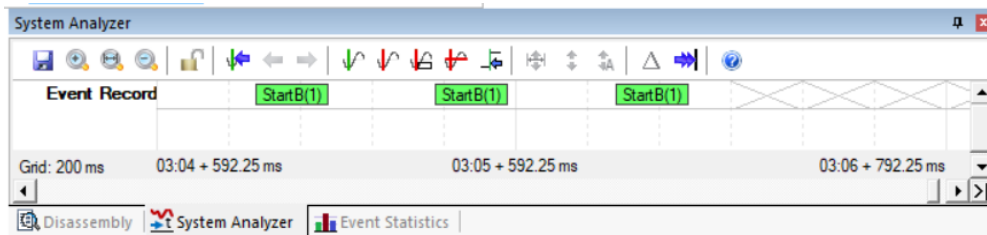
System Analyzer can display much information depending on which Cortex-M processor and debug adapter you are using. It is a bit limited in this case because a Cortex-M0/M0+ does not have Serial Wire Viewer which is used to collect data such as data reads and writes, exception/interrupt events and more. Cortex-M3 and higher processors do provide this information as well as a measured core clock value.

. In this example there is only one set of information which is the Event Recorder frames.

1. Open the System Analyzer window:



2. Stop the data collection with the Freeze Data icon:
3. Unselect the cursor icons to make the screen a bit cleaner.
4. You will see the System Analyzer window with a series of Start blocks as shown below:
5. Expand these blocks in one of two ways:

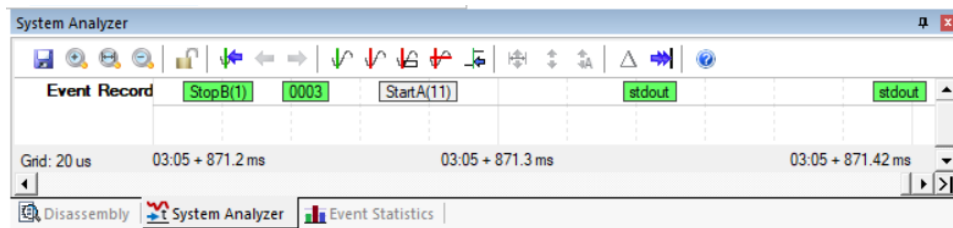


- a. Click on one of the blocks to create a set point and rotate your mouse wheel to expand at this point.

OR

- b. Click on one of the blocks to create a set point and use the expand icon:

6. When you expand it sufficiently you will be able to get something like this below: Note you can now see the individual events. In this short program these events are very close together. A real program might not have them so close together.



7. Turn on the cursors. You will now be able to measure times between the events as shown below. This can provide useful timing values in your code such as how long specified code segments take to execute.



TIP: The reason these are so close is the high speed this simple program is running. If you create another for loop (delay) and rebuild – the blocks will be more evenly spaced for demonstration purposes.

22) Timestamps:

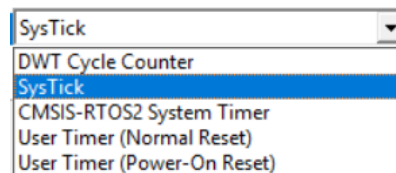
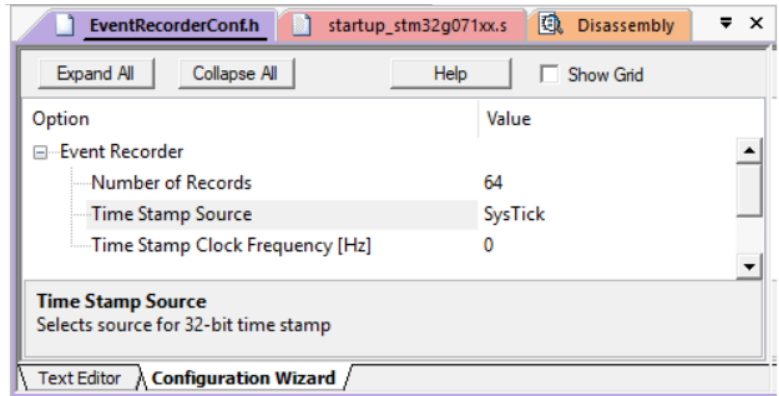
This page is provided for reference on how to select a Timer for timestamping instead of SysTick.:

Our simple example uses the SysTick timer to generate the timestamps you see in various Event Recorder windows. It is configured in EventRecorder.c. This is not normally a problem unless another component attempts to use SysTick such as an RTOS or STM32CubeMX. STM32CubeMX uses SysTick by default to provide tick timers for various functions such as a HAL_Wait() function. Most RTOSs use SysTick as their switching timer. In these cases a “multiply declaration” error results.

It is also possible to use a general purpose timer in the STM32 processor.

Selection of A Timer:

1. Open EventRecorderConf.h or click on its tab if it is already open. Select the Configuration Wizard tab at the bottom. You will see this window:
2. Click on each one and an explanation appears.
3. Open the Time Stamp Source and SysTick will be selected as shown in the next window.
4. Cortex-M0 and M0+ do not have a DWT Cycle Counter register. (DWT_CYCCNT). It therefore cannot be used as a time source and this also prevents the Keil ULINK_{plus} from providing a measured core clock value in System Analyzer.



TIP: ARM Cortex-M3/M4/M7/M33 processors do have a DWT_CYCCNT counter.

RTX and FreeRTOS: CMSIS-RTOS2 allows the RTOS to configure SysTick and Event Recorder will use that setting and not attempt to define SysTick.

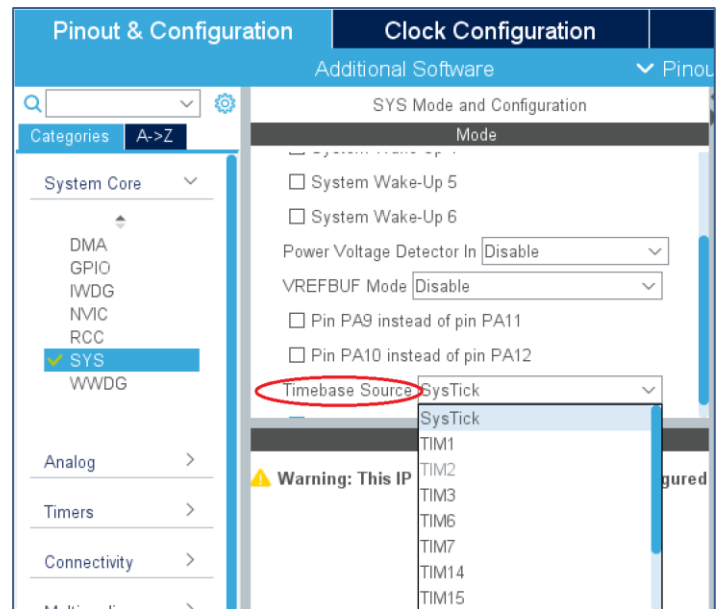
User Timers: These are the STM32 hardware timers. You will need to write some code for your processor timers.

STM32CubeMX Timer Selection:

STM32CubeMX uses the SysTick timer by default. It will use this for various features such as HAL_Delay();.

You can easily change this.

1. In the Pinout and Configuration tab expand System Core as shown below: select SYS as shown below:
2. In the SYS Mode window a Timebase Source opens as shown circled in red.
3. Select a STM32 timer that you are not using.
4. The timer you select will be used in the STM32CubeMX projects that are created.
5. SysTick will now be free to be declared and used by other projects such as Event Recorder or an RTOS.
6. In EventRecorderConf.h select SysTick in the Time Stamp Source as shown above. So EVR can use it.
7. If you are using an RTOS that will use SysTick, you must then use a User Timer else you will get a Multiply Define error at compilation time.
8. If you are using Keil RTX5 you can select CMSIS-RTOS2 option.
9. CMSIS-RTOS2 with FreeRTOS will also function. This is not the FreeRTOS that STM32CubeMX currently provides. CMSIS-RTOS2 FreeRTOS is an enhanced version. www.keil.com/pr/article/1280.htm



23) Keil ULINKplus Operation:

Connecting a Keil ULINKplus to the Nucleo-G071RB and Other Nucleo and Discovery Eval Boards:

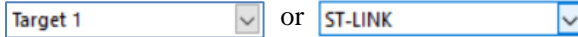
You can replace the ST-LINK/V2 with an external debug adapter to obtain faster debug speed (this helps EVR), faster SWV and more debug features such as power measurement. ULINKplus is a very useful upgrade. Any Keil ULINK can be used.


Keil has an adapter to connect a ULINK to the ST board SWD connector. One is included with a new ULINKplus. A photo is included below. See www.keil.com/support/man/docs/ulinkpro/ulinkpro_sw_d_adapter.htm

1) Create a new Target Selection:

TIP: For other boards you may need to make minor changes to these instructions.

1. Have μ Vision running and in edit mode (not in Debug mode).
2. Select the Target Option Target 1 or ST-LINK to use as a template adapter:



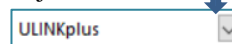
3. Select Project/Manage/Project Items... or select:  The Manage Project Items window opens as shown below right:

4. In the Project Targets area, select NEW. 

5. Enter **ULINKplus** and press Enter.

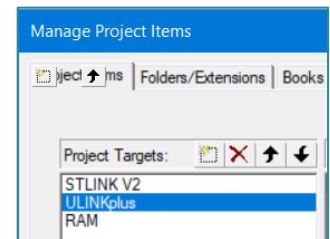
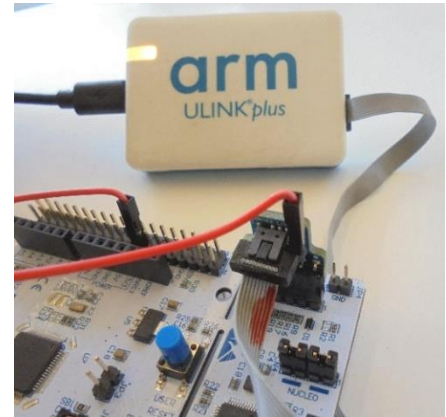
6. Click OK to close the Manage Project Items window.

7. In the Target Selector menu, select the **ULINKplus** selection you just created:



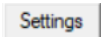


2) Connect a ULINK to Target Board:

1. Connect the Arm SWD Adapter to the SWD connector as shown.
2. Connect a jumper (red wire) from VCC to CN9 pin 16 (3.3 volts).
3. This informs the ULINK the voltage it should present to the JTAG/SWD pins.
4. Connect the 10 pin CoreSight cable to the adapter and the ULINK as shown above.
5. **Put the ST-LINK/V2 processor U2 in RESET by adding a jumper to JP1 STLK_RST. U2 is removed from the system. Not all Nucleo or Discovery boards have this jumper.**
6. **Move the power source jumper JP2 from 1-2 to 7-8 CHG. The USB connector CN2 will still power the board directly. Not all ST boards have this jumper.**



3) Select ULINKplus as the new Debug Adapter:

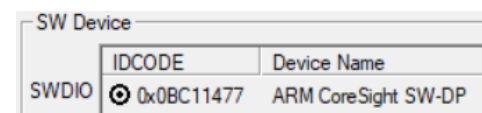
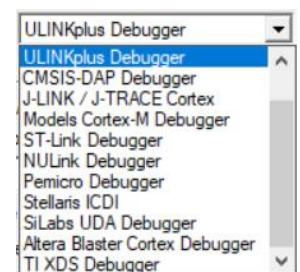
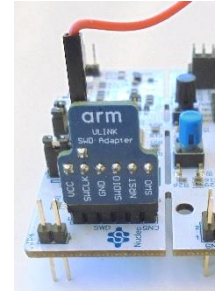
1. Select Options for Target  or ALT-F7. Click on the Debug tab.
2. Select ULINKplus Debugger (or your debugger) as shown here: 
3. Click on Settings: 
4. The Options for the Target ULINKplus window opens.
5. You must see an IDCODE as shown below right: If you do, everything is working.
6. If you see nothing or an RDDI-DAP error, the ULINKplus cannot see the CPU core. This is not a very good situation and needs to be corrected before continuing.
7. Problems include connections are wrong, the target or ULINKplus is not powered. Jumpers are not correctly set.
8. Click on the Trace tab. Set Core Clock: to 16 MHz or whatever the clock speed of the CPU is. This determines various timing displays in μ Vision.
9. Click OK once to return to the Cortex-M Target Driver window.

TIP: You may have to update settings such as NOINIT RAM (page 22).

TIP: Toggle between SW and JTAG to refresh the SW Device box.


Only when this working can you continue to the next page !

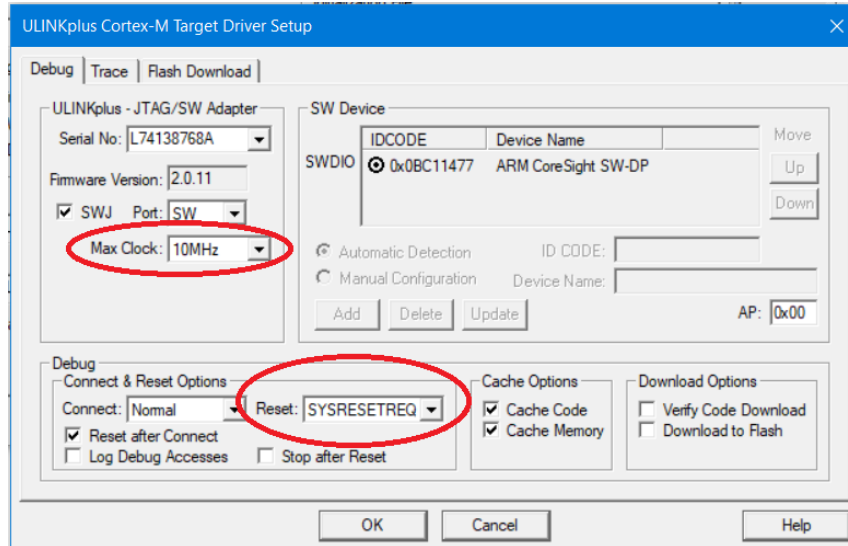
Leave the Target Options window open and continue to configure the new debug adapter:








4) Final Configuration and Testing for ULINKplus:

When you have a valid IDCODE displayed this means you can continue at this point: ULINKplus is connected to the CPU.

1. With the Target Options still open, select Max Clock to 10 MHz as shown below: This is the maximum DAP speed.
2. Set the Reset: box to SYSRESETREQ. Leave everything else at default as shown below.
3. Click OK twice to return to the main menu.
4. Select File/Save All or click: 



5) Compile and Run the Project:

1. Compile the source files by clicking on the Rebuild icon.  Progress is indicated in the Build Output window.
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. The program will run as indicated in various windows.
4. Everything will essentially be running the same as it was with the ST-LINK/V2. There is less chance of any Event Recorder overloads as the ULINKplus is faster.
5. Stop the CPU  and exit Debug mode  for the next step.

TIP: If you cannot find an Arm SWD Adapter:

You can create your own with the information below. You will need to make a simple jig. Use the same configuration settings as shown above. Connect the debug SWD header signals from the ULINKplus to the Nucleo board.

SWD Signal from ULINKplus	Description	Pin on Nucleo	Connector on Nucleo board	ULINK 10 pin CoreSight	Comments
VTREF (VCC)	3.3 volt Target voltage to ULINK	1	CN9 pin 16	1	Tells ULINK the target voltage.
SWDCLK	clock	2	CN5 SWD	4	
GND	ground	3	CN5 SWD	3	
SWDIO	SWD I/O signal	4	CN5 SWD	2	
nRESET	Target reset	5	CN5 SWD	10	optional

24) Keil ULINKplus Operation with Power measurement:

To make power measurements we must connect a small shunt board between the STM32 target and ULINKplus. We will use JP3. This is the 3.3 volt supply to the STM32G0 processor. The led LD3 is not measured as it is ahead of JP3.

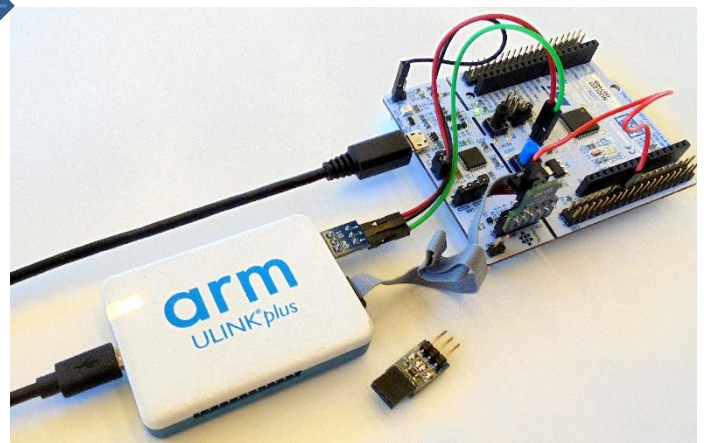
Connecting a Keil ULINKplus Shunt Board to the Nucleo-G071RB Eval Board:

ULINKplus comes with several small shunt boards of various current capabilities. This one is 10 mA:





This board draws about 4 mA at the JP3 IDD jumper. So, we will use the 10mA shunt adapter.

1. Remove the USB cables from both the Nucleo board and the ULINKplus to power them down.
2. Plug the 10 mA shunt adapter into the ULINKplus as shown below. You can connect them either way. There is a method on the next page to ensure they are on the correct polarity.
3. Connect the center pin to ground on the Nucleo board. Either JP4 or JP5 at the corner of the board will do. You can remove the jumper on JP4 or JP5.
Note: You must use this ground as all grounds on ULINKplus are well isolated from each other. Do not depend only on the USB ground.
4. Connect the two outer pins to IDD JP3. The order is not important yet.
5. Your setup will look similar to this now.
6. Connect the two USB cables to power the Nucleo and ULINKplus.





Configuring µVision:




1. Select Options for Target  or ALT-F7.
2. Click the Debug tab.
3. Using the Browse icon , select UlinkPlus.ini as shown below:

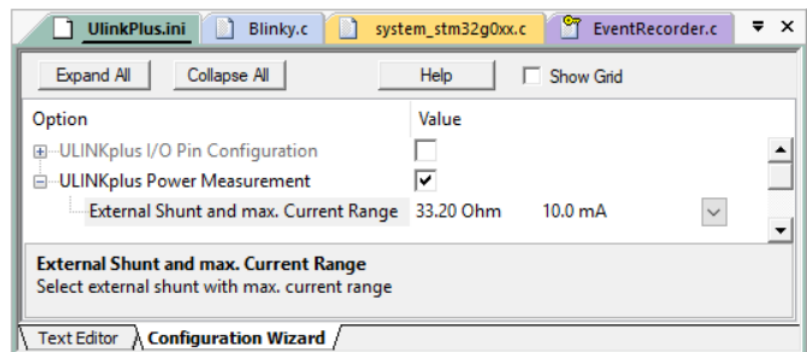


UlinkPlus.ini is used to configure ULINKplus and is executed when you enter Debug mode. It is an ASCII file.

4. Select Edit to open this .ini file. 
5. Select 33.20 Ohm 10.0 mA as shown here:
6. Enable ULINKplus Power Measurement: ☒
7. Select File/Save All or click: 

Run the Program:

1. Click Rebuild. 
2. Enter Debug mode: 
3. Click RUN:  The program will run as indicated by LED LD4 blinking.



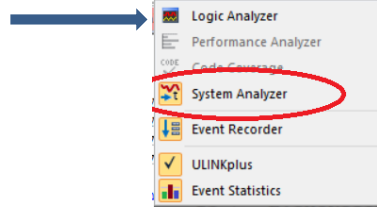





If it does not run, you must find out why before continuing. Problems are usually from missing a step in this document or the cables or jumpers are not connected properly.

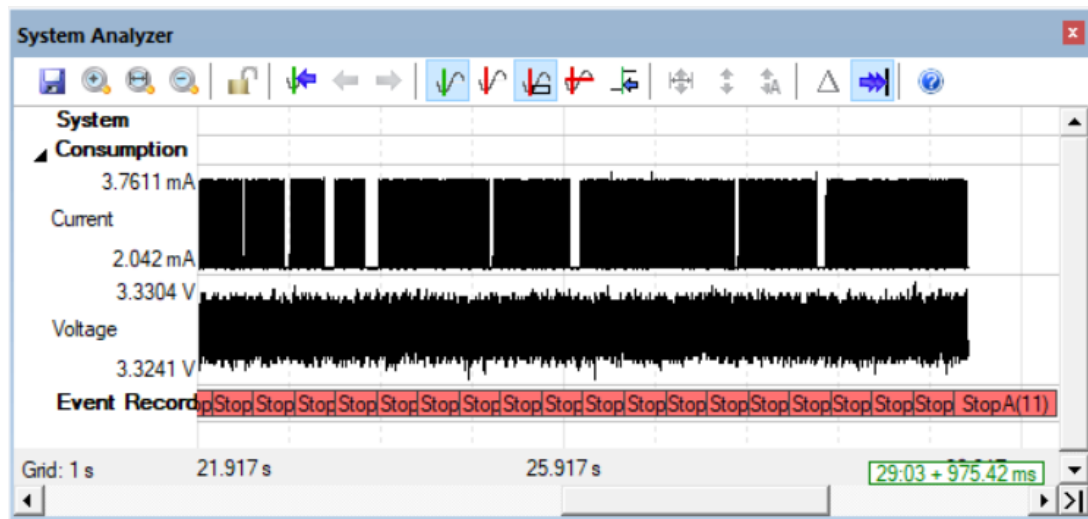
The next page will demonstrate the System Analyzer and Event Statistics windows.

25) Keil ULINKplus Operation with Power measurement:

Note: MDK 5.29 and MDK 5.30 do not accurately display the timestamp accurately. This appears to affect only certain Cortex-M0 processors. Others are not affected. Keil is working on this and any updates will be reflected immediately in this spot with the updated appnote. The voltage and current waveforms are separated from the EVR blocks at each end.

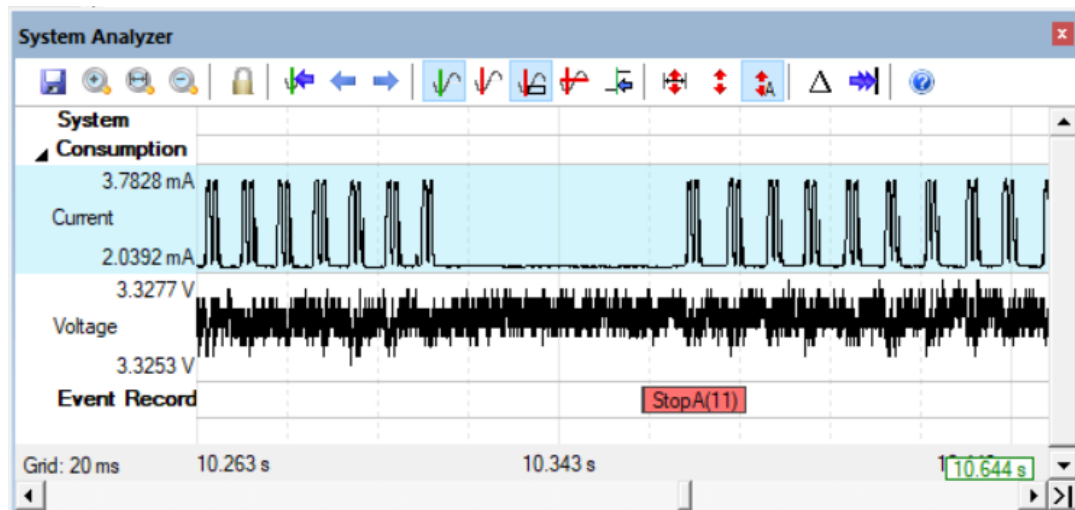
System Analyzer Window:

1. Open the System Analyzer window if it is not already open: 
 2. The window below opens:
 3. Select to keep the waveform at the end: 
 4. Stop the data collection: Click and change  to 
 5. Note the Current and Voltage waveforms. **TIP:** If Current has negative values, exit Debug mode and reverse the shunt wires. Enter Debug mode and they will now be positive.
- TIP:** the voltage and current Y axis numbers are adaptive and will change dynamically to suit their values.
6. You can Zoom in or out with   or your mouse wheel. First click in the waveform to set an anchor point.



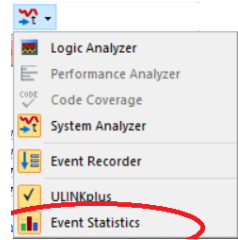
Expanded View:

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



26) 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.

A screenshot of the 'Event Statistics' window. It displays a table with columns for Source, Count, and Filter Enable / Execution Timing. The data is organized into two main groups: 'Event Start/Stop Group A - enabled' and 'Event Start/Stop Group B - enabled'. Each group contains multiple slots with detailed timing and execution statistics for 'Blinky.c'.

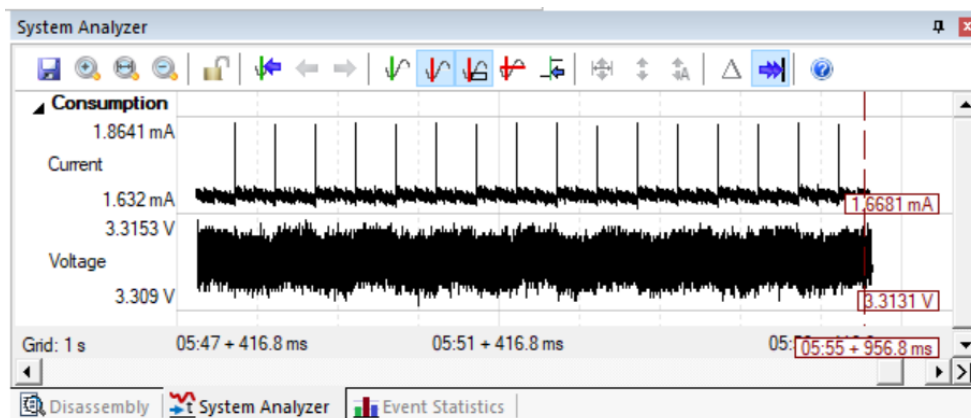
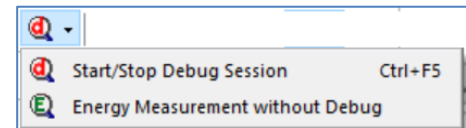
27) Energy Measurement without CoreSight DEBUG:

The CoreSight debug components consume some amount of power. This will be reflected in the System Analyzer window. Since it is not likely the 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 are useful.

The effects will be most pronounced during processor Sleep and WAIT states.

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.




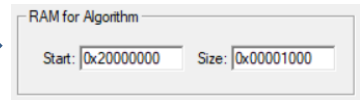
28) Using STM32 Internal Non-Initialized RAM for EVR:

Internal RAM is used to store the data for Event Recorder. When the processor is RESET, this RAM is normally zero-initialized. The settings for the Event Recorder filter (funnel) are also stored in this RAM. Therefore, at RESET all Event Recorder information is lost. To preserve this information through a RESET, you can easily locate the Event Recorder information in non-initialized internal RAM memory.

1. Stop the CPU  and exit Debug mode .

Configure new Internal RAM (IRAM) Area:


2. Select Options for Target  or ALT-F7. Click on the Utilities tab. Flash programming is configured here.
3. Click on the Settings: icon. Note the Start and Size: Do not put EVR memory here.
4. Click OK to go back one window.
5. Click on the Target tab. This is where memory is allocated. From this table μ Vision creates a scatter file. You can also directly create a scatter file. This is selected and entered in the Linker tab.
6. Divide the internal RAM (IRAM) into two segments as shown below: What values are used depends on your setup.
7. Do not use the Flash programming algorithm memory for IRAM2 as determined in Step 3 above.
8. Select NoInit for IRAM2. We will use this for the Event Recorder data.
9. Click OK to close this window.

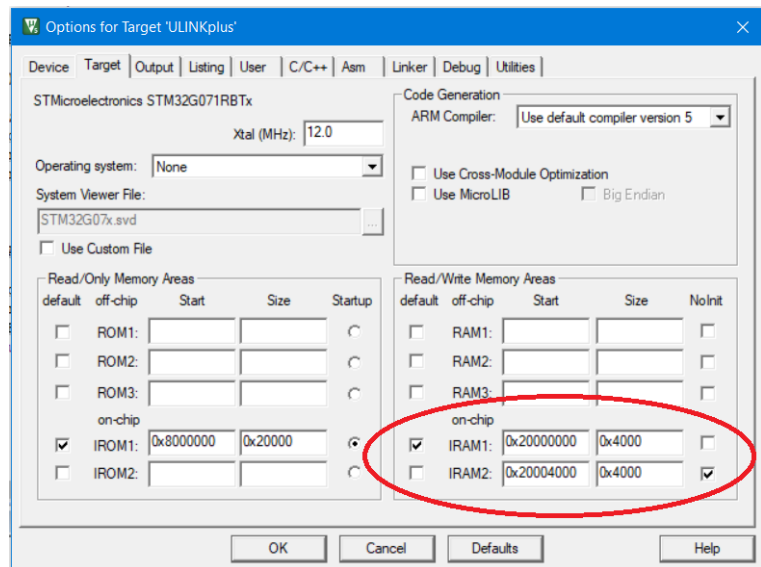


Allocate This Memory to EventRecorder.c:




1. In the Project window, right click on EventRecorder.c and select Options for Component Class 'Compiler'... as shown below middle right:
2. In the window that opens, select the Memory tab.
3. In Memory Assignment select IRAM2 as shown below right:

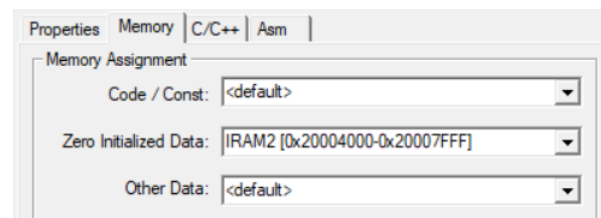
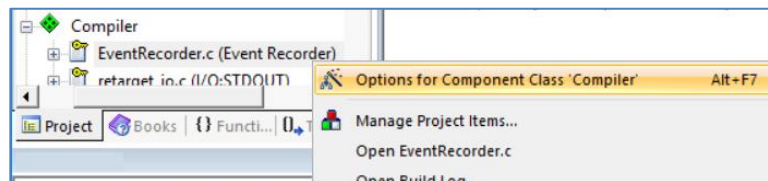
TIP: Selecting the NoInit box in the Target tab effectively makes this memory to not be zero initialized at the program start. Any RAM contents such as EVR frames will be preserved.

4. Click OK to close this window.
5. Select File/Save All or click: 



Build and Run the Program:

1. Click Rebuild. 
2. Enter Debug mode: 
3. Click RUN:  The program will run as before.
4. Details are found here: www.keil.com/pack/doc/compiler/EventRecorder/html/er_use.html



29) Document Resources:

See www.keil.com/ST

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. μ Vision 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.

Application Notes:

1. Using Cortex-M3 and Cortex-M4 Fault Exceptions www.keil.com/appnotes/files/apnt209.pdf
2. Segger emWin GUIBuilder with μ Vision™ www.keil.com/appnotes/files/apnt_234.pdf
3. Porting mbed Project to Keil MDK™ www.keil.com/appnotes/docs/apnt_207.asp
4. MDK-Arm™ Compiler Optimizations www.keil.com/appnotes/docs/apnt_202.asp
5. GNU tools (GCC) for use with μ Vision <https://launchpad.net/gcc-arm-embedded>
6. RTX CMSIS-RTOS in MDK 5 C:\Keil_v5\ARM\PACK\ARM\CMSIS\4.5.0\CMSIS\RTOS\RTX (replace 4.5.0 with the RTX version you desire)
7. RTX CMSIS-RTOS: www.keil.com/rtx and www.keil.com/cmsis
8. Barrier Instructions: <http://infocenter.arm.com/help/topic/com.arm.doc.dai0321a/index.html>
9. Lazy Stacking on the Cortex-M4: www.arm.com and search for DAI0298A
10. Cortex Debug Connectors: www.arm.com and search for cortex_debug_connectors.pdf
11. Sending ITM printf to external Windows applications: www.keil.com/appnotes/docs/apnt_240.asp
12. **NEW!** Migrating Cortex-M3/M4 to Cortex-M7 processors: www.keil.com/appnotes/docs/apnt_270.asp
13. **NEW!** Armv8-M Architecture Technical Overview: <https://community.arm.com/docs/DOC-10896>
14. **NEW!** Determining Cortex-M CPU Frequency using SWV: www.keil.com/appnotes/docs/apnt_297.asp
15. Using ST-LINK/V2 with Keil μ Vision: www.keil.com/appnotes/docs/apnt_286.asp

Training Courses:

1. Keil Training Courses, Workshops and Tradeshows: www.keil.com/events/
2. www.st.com/content/st_com/en/support/learning/stm32-education/stm32-moocs/STM32G0_workshop_MOOC.html

Useful Arm Websites:

1. **NEW!** CMSIS Standards: https://github.com/ARM-software/CMSIS_5 and www.arm.com/cmsis/
2. CMSIS Documents: www.keil.com/pack/doc/CMSIS/General/html
3. Keil Community Forums: <https://community.arm.com/developer/tools-software/tools/f/keil-forum>
4. Arm University Program: www.arm.com/university. Email: university@arm.com
5. [mbed™](http://mbed.org): <http://mbed.org>

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

30) Keil Products and Contact Information:

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.

Free MDK for STM32 F0/L0

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** is equivalent to a ULINK2.
- **New ULINKplus-** Cortex-Mx High performance SWV & power measurement. Available now !
- **ULINKpro** - Cortex-Mx SWV & ETM instruction trace. Code Coverage and Performance Analysis.
- **ULINKpro D** - Cortex-Mx SWV no ETM trace ULINKpro also works with ARM DS-5.

- **For special promotional or quantity pricing and offers, please contact Keil Sales.**
- **In USA and Canada, 1-800-348-8051.** sales.us@keil.com
- **Outside the US: +49 89/456040-20** sales.intl@keil.com
- **Global Inside Sales Contact Point:** Inside-Sales@arm.com

RTOS: Keil RTX RTOS is now provided as part of MDK: See www.keil.com/rtx

DSP: Keil provides free DSP libraries with source code for Cortex-M processors.

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 to view various programs and resources.

MDK supports all STM32 Cortex-M0/M0+, M3, M4, M7 and M33 processors.

Keil supports many other ST processors including 8051, ARM7™, ARM9™ and ST10 processors. See the Keil Device Database® on www.keil.com/dd.

For Linux, Android, other OSs and no OS support on ST Cortex-A processors such as SPEAr, see DS-5 or Arm DS: www.arm.com/ds5.



For more information:

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

Global Inside Sales Contact Point: Inside-Sales@arm.com

Keil World Wide Distributors: www.keil.com/distis/

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

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