Freescale Kinetis: Cortex™-M4 Training Lab

ARM® Keil™ MDK Toolkit *featuring Serial Wire Viewer and ETM Trace*

Winter 2010 Version 0.52 by Robert Boys, bob.boys@arm.com ***Prelininary***

Introduction:

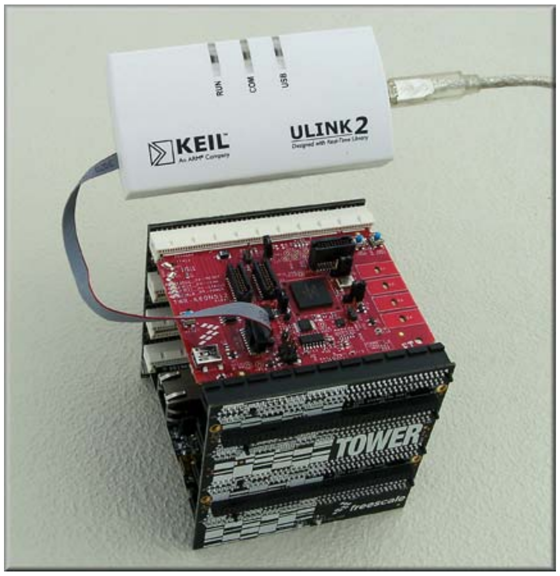
The purpose of this lab is to introduce you to the Freescale Cortex™-M4 processor using the ARM® Keil™ MDK toolkit featuring the IDE µVision®. We will use the Serial Wire Viewer (SWV) and ETM trace on the Kinetis K40 and K60 processor. At the end of this tutorial, you will be able to confidently work with these processors and Keil MDK.

Keil MDK comes in an evaluation version that limits code and data size to 32 Kbytes. Nearly all Keil examples will compile within this 32K limit. The addition of a license number will turn it into the full, unrestricted version. Contact Keil sales for a temporary full version license if you need to evaluate MDK with programs greater than 32K. MDK includes a full version of Keil RTX. No royalty payments are required. If you need the RTX source code this is included in Keil RL-ARM.

Why Use Keil MDK ?

MDK provides these features particularly suited for Cortex-M4 users:

1. µVision IDE with Integrated Debugger, Flash programmer and the ARM® Compiler toolchain. MDK is a turn-key product and includes examples.
2. A full feature Keil RTOS called RTX is included with MDK.
3. Serial Wire Viewer and ETM trace capability is included.
4. RTX Kernel Awareness window is updated in real-time.
5. Choice of adapters: ULINK2, ULINK-ME, ULINK*pro*, Segger J-Link (version 6 or later) and Signum Systems JtagJetTrace. P&E is supported by µVision.
6. Kernel Awareness for Keil RTX, CMX, Quadros and Micrium. All RTOSs will compile with MDK.
7. Keil Technical Support is included for one year and is renewable. This helps you get your project completed faster.



This document details these features:

1. Serial Wire Viewer (SWV).
2. ETM Trace.
3. Real-time Read and Write to memory locations for Watch, Memory 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. Six Hardware Breakpoints (can be set/unset on-the-fly) and four Watchpoints (also called Access Breaks).
5. RTX Viewer: a kernel awareness program for the Keil RTX RTOS.

Note: This document is a work in progress. Please contact the author or www.keil.com for the latest version.

Serial Wire Viewer (SWV):

Serial Wire Viewer (SWV) displays PC Samples, Exceptions (including interrupts), data reads and writes, ITM (printf), CPU counters and a timestamp. This information comes from the ARM CoreSight™ debug module integrated into the Cortex-M4. SWV is output on the Serial Wire Output (SWO) pin found on the JTAG adapter.

SWV does not steal any CPU cycles and is completely non-intrusive.(except for ITM Debug printf Viewer). SWV is provided by the Keil ULINK family, Segger J-Link and the Signum Systems JtagJetTrace. Best results are with a ULINK.

Enhanced Trace Macrocell (ETM):

ETM adds all the program counter values to the features provided by SWV. This allows advanced debugging features including timing of areas of code, Code Coverage, Performance Analysis and program flow analysis. ETM requires a special debugger adapter such as the ULINK*pro*. Segger J-Trace or the Signum Systems JtagJetTrace. This document uses ULINK*pro* for ETM. A ULINK2 or ULINK-ME is used for the Serial Wire Viewer exercises.

Freescale Evaluation Boards:

This document uses the Freescale Kinetis TWR-K60N512. The K40 board can also be used.

Please note LEDs E1 through E4 are on Port C on the K40 board and on Port A on the K60 board.

Software Installation:

This document was written for Keil MDK 4.13a or later which contains µVision 4. The evaluation copy of MDK is available free on the Keil website. Do not confuse µVision4 with MDK 4.0. The number "4" is a coincidence.

To obtain a copy of MDK go to www.keil.com/arm and select "Evaluation Software" from the left column.

You can use the evaluation version of MDK and a ULINK2, ULINK-ME, ULINK*pro*, J-Link or JtagJetTrace for this lab. You must make certain adjustments for non-ULINK adapters and not all features shown here will be available.

The ULINK*pro* adds Cortex-M4 ETM trace support. It also adds faster programming time and better trace display.

Example Programs:

Currently two example programs plus two for programming the Freescale FlexMem are included in either C:\Keil\ARM\Boards\Freescale\TWR-K40N256 or C:\Keil\ARM\Boards\Freescale\TWR-K60N512.

- **Blinky:** blinks 3 LEDs. Is a great starting point and can be adapted to your own project.
- **RTX_Blinky:** A motor controller with RTX (the Keil RTOS) implemented. An easy intro to RTX.
- **FlexMem_Cfg** and **ProgOnce_Cfg**: See Keil Appnote 220 to help you easily configure FlexMem.

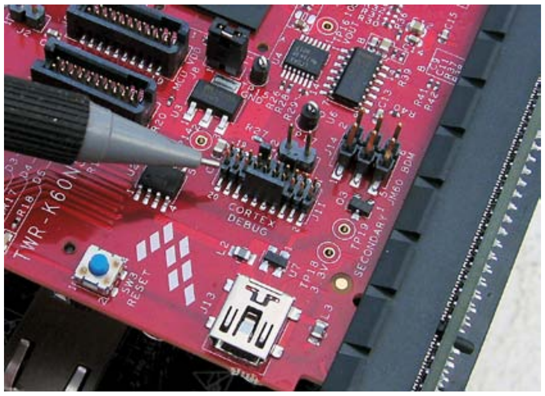
Index:

1) Connecting ULINK2, ULINK-ME or ULINK*pro* to the Freescale Tower board:

Freescale provides the ARM standard 20 pin hi-density connector for JTAG/SWD and ETM connection as shown here:

Pins 1 through 10 provide JTAG, SWD and SWO signals. Pins 11 through 20 provide the ETM trace signals.

ARM also provides a 10 pin standard connector that provides the first 10 pins of the 20 pin but this is not provided on the Kinetis board. ARM recommends that both the 10 and 20 pin connectors be placed on target boards.



Pin 7 on both 10 and 20 pin is a key. On the male connector, pin 7 is supposed to be absent. This is not the case on the Kinetis board.

On the female cable ends, a short plastic pin fills pin 7.

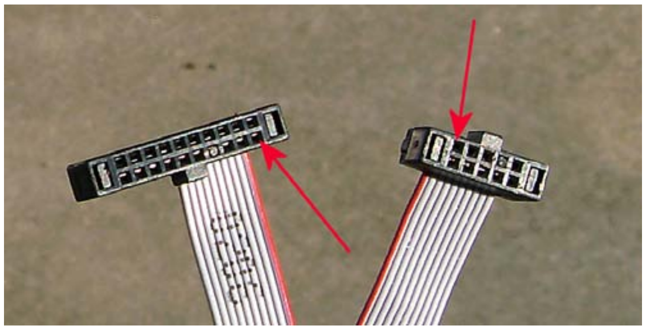
Keil cables usually have pin 7 filled and this will need to be removed before connecting to the Kinetis target. This is easily done with a sharp needle. Merely pry the pin out.

Alternatively, you can cut pin 7 off the target and ULINK2 socket. Cable orientation is provided by the socket itself and there is no chance for reverse orientation.

It is impossible to plug a 10 pin plug into the 20 pin socket without bending two pins on the socket.

Connecting ULINK2 or ULINK-ME:

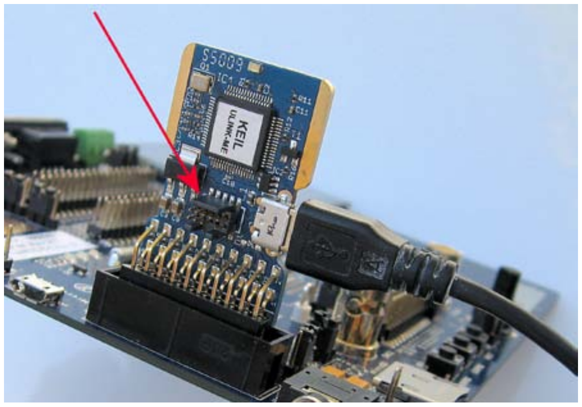
Pictured is the 10 pin to 20 pin Keil connector. The arrows point to pin 1. This cable is not currently supplied with the ULINK2 or ULINK-ME. Contact Keil technical support or sales to obtain one.



You will need to take the case off the ULINK2 and install the special cable. The ULINK-ME does not have a case and the cable can be directly installed on the 10 pin connector. The ULINK-ME is pictured and the arrow points the 10 pin connector.

Connecting ULINK*pro*:

The ULINKpro connects directly to the Kinetis board with its standard 20 pin connector. You might need to remove the key pin to connect to the Kinetis target.



The pictured 10 to 20 pin cable is supplied with the ULINK*pro* but is not needed for this target. ULINK*pro* is pictured here:

2) ULINK2 or ULINK-ME and µVision Configuration:

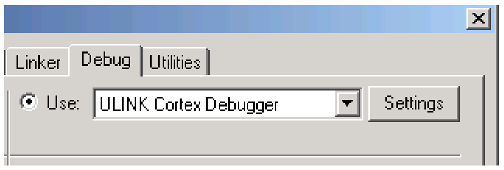
It is easy to select a USB debugging adapter in µVision. You must configure the connection to both the target and to Flash programming in two separate windows as described below. They are selected using the Debug and Utilities tabs.

This document will use a ULINK2 or ULINK-ME as described. You can substitute a ULINK*pro* with suitable adjustments.

Serial Wire Viewer is completely supported by these two adapters. They are essentially the same devices electrically and any reference to ULINK2 here includes the ME. The ULINK*pro*, which is a Cortex-Mx ETM trace adapter, can be used like a ULINK2 or ULINK-ME with the advantages of faster programming time and an enhanced instruction trace window.

1. Assume the ULINK2 is connected to a powered up Kinetis target board, µVision is running in Edit mode (as it is when first started – the alternative to Debug mode) and you have selected a valid project. The ULINK2 is shown connected to the Freescale K60 Tower board on page 1.

Select the debug connection to the target:

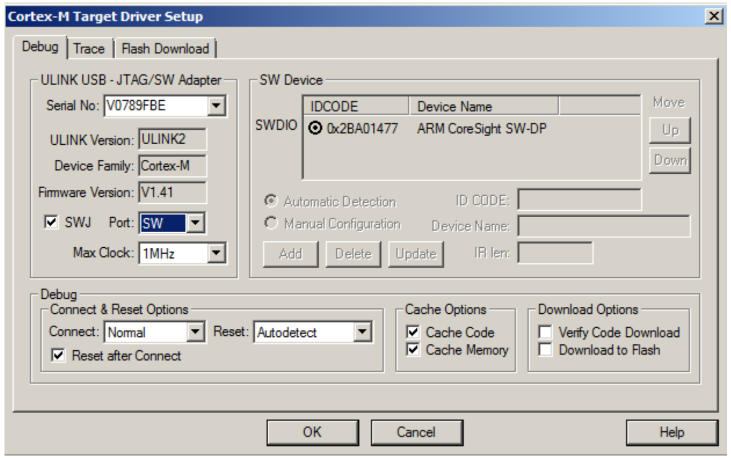


Use: ULINK Cortex Debugger

2. Select Options for Target or ALT-F7 and select the Debug tab. In the drop-down menu box select the ULINK as shown here:
3. Select Settings and the next window below opens up. This is the control panel for the ULINK 2 and ULINK-ME (they are the same).
4. In **Port:** select SWJ and SW. SWV will not work with JTAG selected.
5. In the SW Device area: ARM CoreSight SW-DP **MUST** be displayed. This confirms you are connected to the target processor. If there is an error displayed or is blank this **must** be fixed before you can continue. Check the target power supply. Cycle the power to the ULINK and the board.

TIP: To refresh this screen select Port: and change it or click OK once to leave and then click on Settings again.

TIP: You can do regular debugging using JTAG. SWD and JTAG operate at approximately the same speed. Serial Wire Viewer (SWV) will not operate in JTAG mode.



ULINK USB - JTAG/SW Adapter
Serial No: V0789FBE
ULINK Version: ULINK2
Device Family: Cortex-M
Firmware Version: V1.41
☑ SWJ Port: SW
Max Clock: 1MHz

SW Device

	IDCODE	Device Name
SWDIO	0x2BA01477	ARM CoreSight SW-DP

Automatic Detection ID CODE:
Manual Configuration Device Name:
IR len:

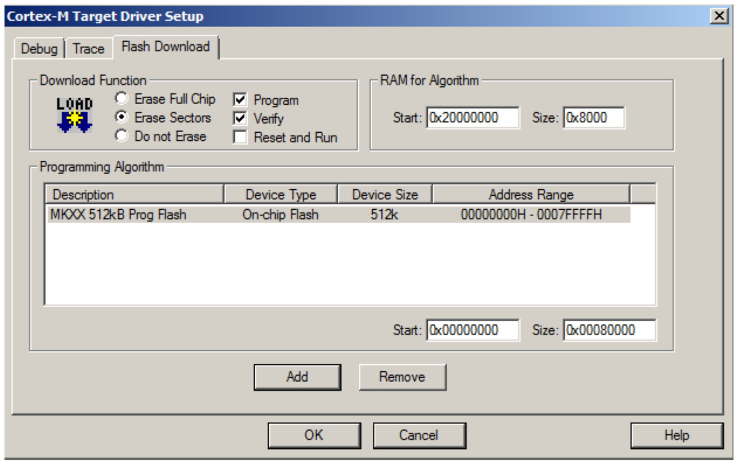
Debug
Connect & Reset Options
Connect: Normal Reset: Autodetect
☑ Reset after Connect
Cache Options
☑ Cache Code
☑ Cache Memory
Download Options
☐ Verify Code Download
☐ Download to Flash

Configure the Keil Flash Programmer:

6. Click on OK once and select the Utilities tab.
7. Select the ULINK similar to Step 2 above.
8. Click Settings to select the programming algorithm if one is not visible.
9. Select Add and select MKXX 512kB Prog Flash as shown below or the one for your processor:
10. Enter 0x2000_0000 and 0x8000 in the "RAM for Algorithm" Start: and Size: boxes.
11. Click on OK once.

TIP: To program the Flash every time you enter Debug mode, check Update target before Debugging.

12. Click on OK to return to the µVision main screen.
13. You have successfully connected to the Kinetis target processor.



Download Function
Erase Full Chip
● Erase Sectors
Do not Erase
☑ Program
☑ Verify
☐ Reset and Run

RAM for Algorithm
Start: 0x20000000 Size: 0x8000

Programming Algorithm

Description	Device Type	Device Size	Address Range
MKXX 512kB Prog Flash	On-chip Flash	512k	00000000H - 0007FFFFH

Start: 0x00000000 Size: 0x00080000

TIP: The Trace tab is where you configure the Serial Wire Viewer (SWV). You will learn to do this later.

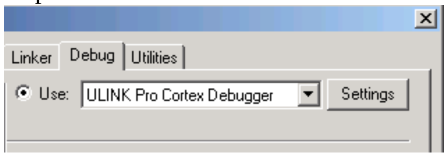
TIP: If you select ULINK or ULINK*pro*, and have the opposite ULINK physically connected to your PC; the error message will say "No ULINK device found". This message actually means that µVision found the wrong Keil adapter connected.

3) ULINK*pro* and µVision Configuration:

1. Assume the ULINK*pro* is connected to a powered up Kinetis target board, µVision is running in Edit mode (as it is when first started – the alternative to Debug mode) and you have selected a valid project. The ULINK*pro* is shown connected to the Freescale K60 Tower board below.

Select the debug connection to the target:

2. Select Options for Target or ALT-F7 and select the Debug tab. In the drop-down menu box select the ULINK as shown here:



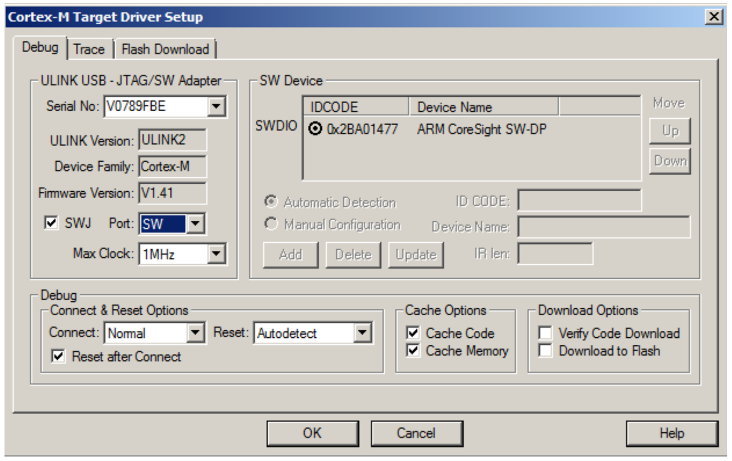
Linker | Debug | Utilities

Use: ULINK Pro Cortex Debugger

Settings

3. Select Settings and Target Driver window below opens up:
4. In **Port:** select SWJ and SW. SWV will not work with JTAG selected.
5. In the SW Device area: ARM CoreSight SW-DP **MUST** be displayed. This confirms you are connected to the target processor. If there is an error displayed or is blank this **must** be fixed before you can continue. Check the target power supply. Cycle the power to the ULINK and the board.

TIP: To refresh this screen select Port: and change it or click OK once to leave and then click on Settings again.



Cortex-M Target Driver Setup

Debug | Trace | Flash Download

ULINK USB - JTAG/SW Adapter
Serial No: V0789FBE
ULINK Version: ULINK2
Device Family: Cortex-M
Firmware Version: V1.41
☑ SWJ Port: SW
Max Clock: 1MHz

SW Device

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

Move Up Down

Automatic Detection ID CODE:
Manual Configuration Device Name:
Add Delete Update IR len:

Debug
Connect & Reset Options
Connect: Normal Reset: Autodetect
☑ Reset after Connect

Cache Options
☑ Cache Code
☑ Cache Memory

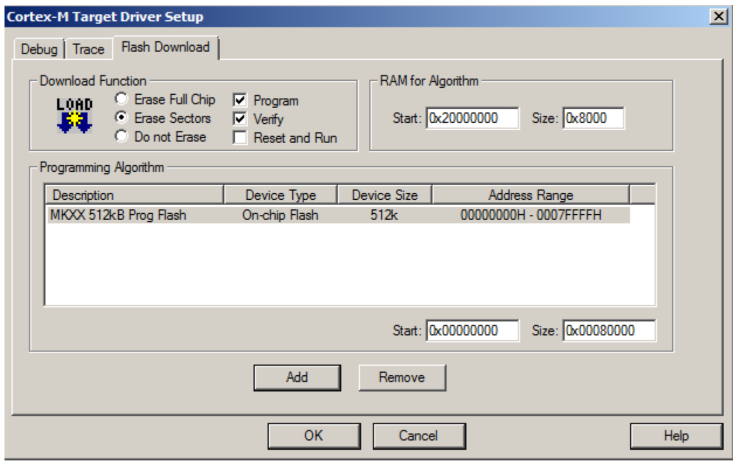
Download Options
☐ Verify Code Download
☐ Download to Flash

OK Cancel Help

TIP: You can do regular debugging using JTAG. SWD and JTAG operate at approximately the same speed. Serial Wire Viewer (SWV) will not operate in JTAG mode unless the ULINK*pro* is using the Trace Port to output the trace frames.

Configure the Keil Flash Programmer:

1. Click on OK once and select the Utilities tab.
2. Select the ULINK*pro* similar to Step 2 above.
3. Click Settings to select the programming algorithm if one is not visible.
4. Select Add and select MKXX 512kB Prog Flash as shown here or the appropriate one for your processor:
5. Enter 0x2000_0000 and 0x8000 in the "RAM for Algorithm" Start: and Size: boxes.
6. Click on OK once.



Cortex-M Target Driver Setup

Debug | Trace | Flash Download

Download Function
○ Erase Full Chip ☑ Program
● Erase Sectors ☑ Verify
○ Do not Erase ☐ Reset and Run

RAM for Algorithm
Start: 0x20000000 Size: 0x8000

Programming Algorithm

Description	Device Type	Device Size	Address Range
MKXX 512kB Prog Flash	On-chip Flash	512k	00000000H - 0007FFFFH

Start: 0x00000000 Size: 0x00080000

Add Remove

OK Cancel Help

TIP: To program the Flash every time you enter Debug mode, check Update target before Debugging.

7. Click on OK to return to the µVision main screen.
8. You have successfully connected to the Kinetis target processor.

TIP: If you select ULINK or ULINK*pro*, and have the opposite ULINK physically connected; the error message will say "No ULINK device found". This message actually means that µVision found the wrong Keil adapter connected.

TIP: A ULINK*pro* will act very similar to a ULINK2. The trace window (Instruction Trace) will be quite different from the ULINK2 Trace Records as it offers additional features.

TIP: µVision windows can be floated anywhere. You can restore them by setting Window/Reset Views to default.

4) *Blinky* example program using the Kinetis and ULINK2 or ULINK*pro*:

Now we will connect a Keil MDK development system using real target hardware and a ULINK2 or ULINK-ME. These instructions use a K60 board. If you are using a K40 board, select the appropriate files and directories.

It is possible to use the ULINKpro or the P&E

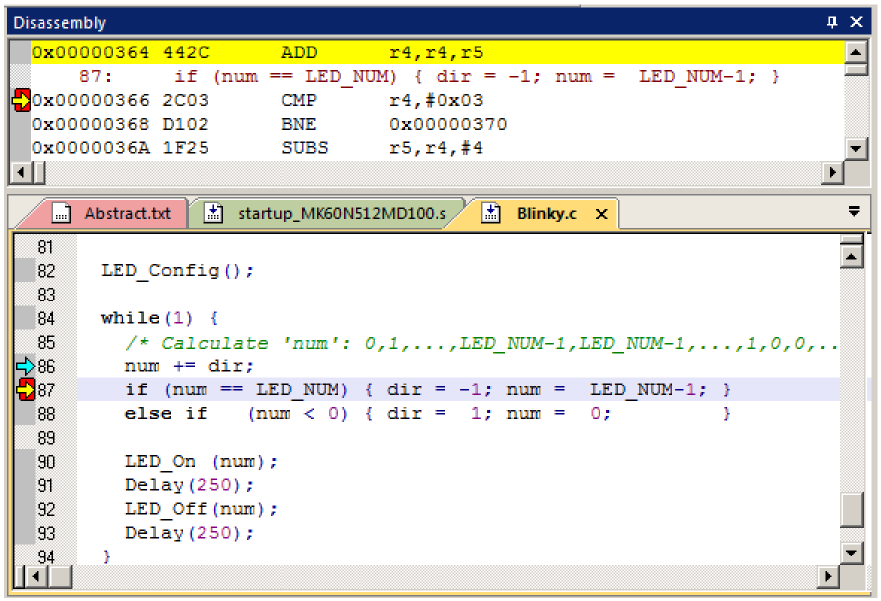
1. Connect the equipment as pictured on the first page.
2. Start µVision by clicking on its desktop icon.
3. Select Project/Open Project. Open the file C:\Keil\ARM\Boards\Freescale\TWR-K60N512\Blinky\Blinky.uvproj.
4. Make sure "K60X256 Flash" is selected. K60X256 - Flash
   This is where you create and select different target configurations such as to execute a program in RAM or Flash.
5. Configure your USB-JTAG adapter at this point. See pages 4 (ULINK2 or ME) and 5 (ULINK*pro*).
6. Compile the source files by clicking on the Rebuild icon. . You can also use the Build icon beside it.
7. Program the Kinetis flash by clicking on the Load icon: Progress will be indicated in the Output Window.
8. Enter Debug mode by clicking on the Debug icon. Select OK if the Evaluation Mode box appears.
   Note: You only need to use the Load icon to download to FLASH and not for RAM operation or the simulator.
9. Click on the RUN icon. Note: you stop the program with the STOP icon.

The LEDs E1 through E3 on the Kinetis board will now blink.

Now you know how to compile a program, load it into the Kinetis processor Flash, run it and stop it.

5) Hardware Breakpoints:

1. With Blink running, double-click in the left margin on a darker gray block somewhere in the while(1) loop as shown below:
2. A red block is created and soon the program will stop at this point.
3. The yellow arrow is where the program counter is pointing to in both the disassembly and source windows.
4. Note you can set hardware breakpoints while the program is running.
5. The Kinetis has 6 hardware breakpoints. A breakpoint does not execute the instruction it is set to.
6. Remove the breakpoint by double-clicking on the red block.



```
0x00000364 442C      ADD      r4,r4,r5
    87:      if (num == LED_NUM) { dir = -1; num =  LED_NUM-1; }
0x00000366 2C03      CMP      r4,#0x03
0x00000368 D102      BNE      0x00000370
0x0000036A 1F25      SUBS     r5,r4,#4
```

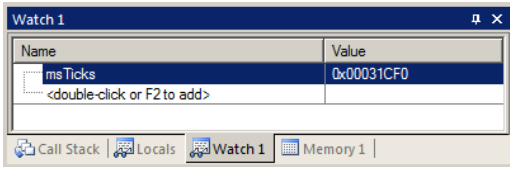
```
  LED_Config();

  while(1) {
    /* Calculate 'num': 0,1,...,LED_NUM-1,LED_NUM-1,...,1,0,0,..
    num += dir;
    if (num == LED_NUM) { dir = -1; num =  LED_NUM-1; }
    else if   (num < 0) { dir =  1; num =  0;         }

    LED_On (num);
    Delay(250);
    LED_Off(num);
    Delay(250);
  }
```

6) 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 through the ARM CoreSight debugging technology that is part of Cortex-M4 processors. It is also possible to "put" or insert values into these memory locations in real-time. It is possible to "drag and drop" variables into windows or enter them manually.

Watch window:

1. You can do the following steps while the CPU is running. Click on RUN if needed.
2. In the file Blinky.c locate the volatile variable **`msTicks`**. It will be near line 19.
3. Open the Watch 1 window by clicking on the Watch 1 tab as shown or select View/Watch Windows/Watch 1.
4. In Blinky.c, block **`msTicks`**, click and hold and drag it into Watch 1. Release it and it will be displayed updating as shown here:
5. You can also enter a variable manually by double-clicking or pressing F2 and using copy and paste or typing the variable.



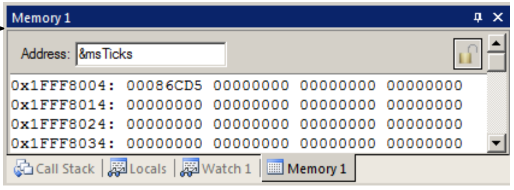
Name	Value
msTicks	0x00031CF0
<double-click or F2 to add>	

TIP: To Drag 'n Drop into a tab that is not active, pick up the variable and hold it over the tab you want to open; when it opens, move your mouse into the window and release the variable.

6. Double click on the value for **`msTicks`** in the Watch window. Enter the value 0 and press Enter. 0 will be inserted into memory in real-time. You can do this in the Memory window with a right-click and select Modify Memory.

Memory window:

1. Drag 'n Drop **`msTicks`** into the Memory 1 window or enter it manually.
2. Note the value of **`msTicks`** 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. Add an ampersand "&" in front of the variable name and press Enter. Now the physical address is shown (0x1FFF_8004).
4. Right click in the memory window and select Unsigned/Int.
5. The data contents of **`msTicks`** is displayed as shown here:
6. Both the Watch and Memory windows are updated in real-time.



Address: &msTicks

```
0x1FFF8004: 00086CD5 00000000 00000000 00000000
0x1FFF8014: 00000000 00000000 00000000 00000000
0x1FFF8024: 00000000 00000000 00000000 00000000
0x1FFF8034: 00000000 00000000 00000000 00000000
```

TIP: You are able to configure the Watch and Memory windows and change their values while the program is still running in real-time without stealing any CPU cycles.

How It Works:

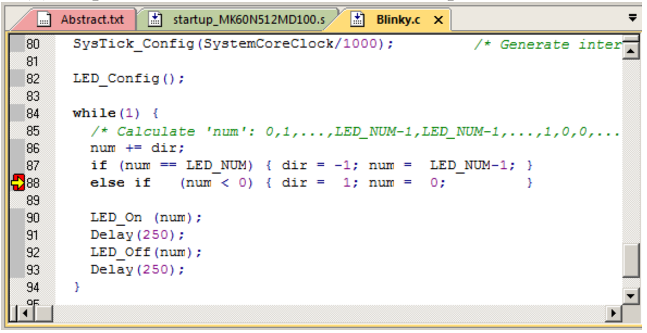
µVision uses ARM CoreSight technology to read or write memory locations without stealing any CPU cycles. This is nearly always non-intrusive and does not impact the program execution timings. Remember the Cortex-M4 is a Harvard architecture. This means it has separate instruction and data buses. While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

You are not able to view local variables while the program is running.

How to view Local Variables in the Watch or Memory windows:

1. Make sure Blinky.c is running.
2. Locate where the local variable **num** is declared in Blinky.c near line 76, at the start of the main function.
3. Drag and Drop **num** into Watch 1 window. Note you are unable to accomplish this.
4. Double-click in the left margin, somewhere past the while (1) loop in main to set a hardware breakpoint such as at line 88 shown here:

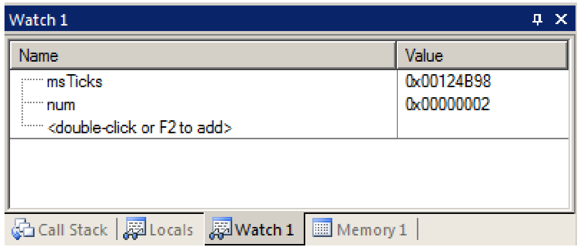


```
80    SysTick_Config(SystemCoreClock/1000);          /* Generate inter
81
82    LED_Config();
83
84    while(1) {
85      /* Calculate 'num': 0,1,...,LED_NUM-1,LED_NUM-1,...,1,0,0,...
86      num += dir;
87      if (num == LED_NUM) { dir = -1; num =  LED_NUM-1; }
88      else if   (num < 0) { dir =  1; num =  0;         }
89
90      LED_On (num);
91      Delay(250);
92      LED_Off(num);
93      Delay(250);
94    }
```

5. The program will stop and you can now enter **num** into Watch 1. Remove the breakpoint by double-clicking on it to remove the red block.

   TIP: Remember: you can set hardware breakpoints on-the-fly in the Cortex-M4 !

6. Run the program and note **num** value displays as ?????????.
7. Stop the program and it will probably stop in the Delay function. **num** now displays as <out of scope> or ????????.
8. These effects are because µVision is unable to determine the value of **num** when the program is running because it exists only when main is running. It disappears in functions and handlers outside of main.
9. Start the program by clicking on the Run icon.
10. **num** changes to ????????. Set a breakpoint by double-clicking in the margin while (1) loop as before in Step 4 above. The program will soon stop on this hardware breakpoint.
11. **num** correctly displays the value as shown here:
12. Each time you click RUN, this value is updated.



Name	Value
msTicks	0x00124B98
num	0x00000002
<double-click or F2 to add>	

How to view these variables updated in real-time:

Note: Drag and Drop does not currently work correctly for the variable num. Stop the processor and enter it manually.

All you need to do is to make **num** static !

1. In the declaration for **num** add **`static`** like this:

```
        int main (void) {
                static int num = -1;
```

2. Exit debug mode. **TIP:** You can edit files in edit or debug mode, but can compile them only in edit mode.
3. Compile the source files by clicking on the Rebuild icon. Hopefully they compile with no errors or warnings.
4. To program the Flash, click on the Load icon. . A progress bar will be displayed at the bottom left.

TIP: To program the Flash automatically when you enter Debug mode select Options For Target , select the Utilities tab and select the "Update Target before Debugging" box.

5. Enter Debug mode. You will have to re-enter **num** in the Watch 1 window because it isn't the same variable anymore – it is a static variable now instead of a local. Drag 'n Drop is the fastest way.
6. Remove the breakpoint you previously set and click on RUN. You can use Debug/Kill All Breakpoints to do this.
7. **num** is now updated in real-time. This is ARM CoreSight technology working.
8. Stop the CPU and exit debug mode for the next step. and

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

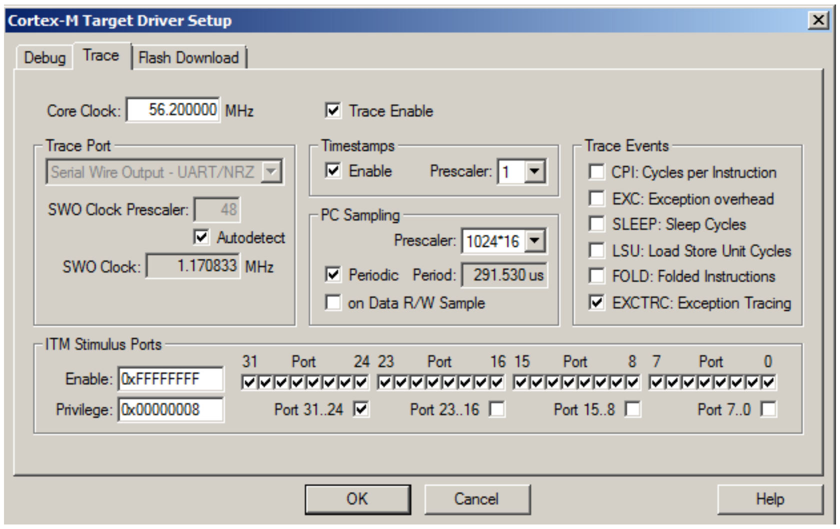
5) Getting the Serial Wire Viewer (SWV) working:

Serial Wire Viewer provides program information in real-time.

A) For ULINK2 or ULINK-ME: (ULINK*pro* instructions are on the next page)

Configure SWV:

1. µVision must be stopped and in edit mode (not debug mode).
2. Select Options for Target or ALT-F7 and select the Debug tab.
3. Click on Settings: beside the name of your adapter on the right side of the window.
4. Click on the Trace tab. The window below is displayed.
5. In Core Clock: enter 56.2 and select the Trace Enable box.
6. Select Periodic and leave everything else at default.
7. Click on OK twice to return to the main µVision menu. SWV is now configured.



Cortex-M Target Driver Setup

Debug | Trace | Flash Download

Core Clock: 56.200000 MHz   ☑ Trace Enable

Trace Port: Serial Wire Output - UART/NRZ
SWO Clock Prescaler: 48
☑ Autodetect
SWO Clock: 1.170833 MHz

Timestamps: ☑ Enable   Prescaler: 1

PC Sampling: Prescaler: 1024*16
☑ Periodic   Period: 291.530 us
☐ on Data R/W Sample

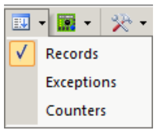
Trace Events:
☐ CPI: Cycles per Instruction
☐ EXC: Exception overhead
☐ SLEEP: Sleep Cycles
☐ LSU: Load Store Unit Cycles
☐ FOLD: Folded Instructions
☑ EXCTRC: Exception Tracing

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

OK   Cancel   Help

Display Trace Records:

8. Enter Debug mode.
9. Click on the RUN icon. .
10. Open Trace Records window by clicking on the small arrow beside the Trace icon:
11. The Race Records window will open and display Exceptions and PC Samples as shown below:



TIP: If you see the Num column contain any numbers other than 15 or 0, the trace is not configured correctly. The most probable cause is the Core Clock: frequency is wrong. Try 52.9 or 48 MHz.

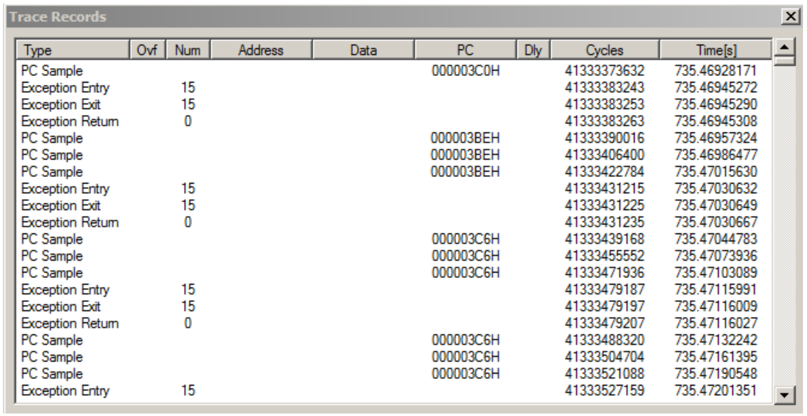
Exception 15 is the SYSTICK timer. It is a dedicated timer for use with RTOSs.

All frames have a timestamp displayed.

Exception Return means all exceptions have returned. Can detect Cortex tail-chaining.

If you open the Exceptions window you will see SYSTICK displayed in another format.

Double-click inside any of these two windows to clear them.



Trace Records

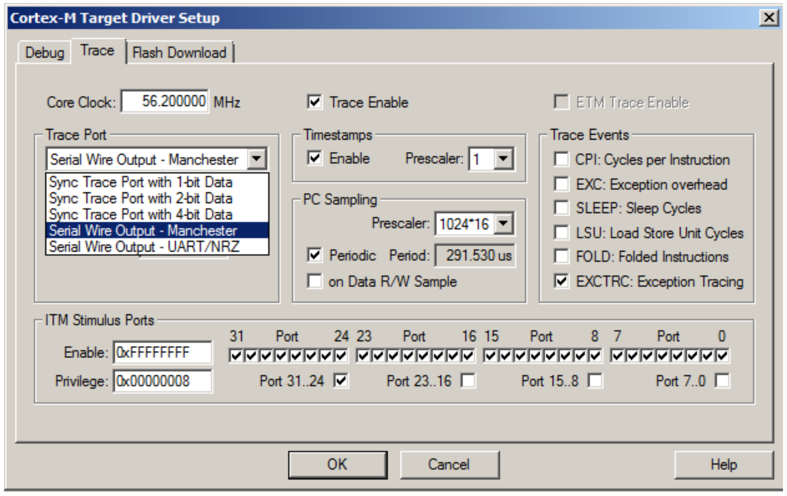
Type	Ovf	Num	Address	Data	PC	Dly	Cycles	Time[s]
PC Sample					000003C0H		41333373632	735.46928171
Exception Entry		15					41333383243	735.46945272
Exception Exit		15					41333383253	735.46945290
Exception Return		0					41333383263	735.46945308
PC Sample					000003BEH		41333390016	735.46957324
PC Sample					000003BEH		41333406400	735.46986477
PC Sample					000003BEH		41333422784	735.47015630
Exception Entry		15					41333431215	735.47030632
Exception Exit		15					41333431225	735.47030649
Exception Return		0					41333431235	735.47030667
PC Sample					000003C6H		41333439168	735.47044783
PC Sample					000003C6H		41333455552	735.47073936
PC Sample					000003C6H		41333471936	735.47103089
Exception Entry		15					41333479187	735.47115991
Exception Exit		15					41333479197	735.47116009
Exception Return		0					41333479207	735.47116027
PC Sample					000003C6H		41333488320	735.47132242
PC Sample					000003C6H		41333504704	735.47161395
PC Sample					000003C6H		41333521088	735.47190548
Exception Entry		15					41333527159	735.47201351

B) For ULINK*pro*:

Configure SWV:

1. µVision must be stopped and in edit mode (not debug mode).
2. Select Options for Target or ALT-F7 and select the Debug tab.
3. Click on Settings: beside the name of your adapter on the right side of the window.
4. Click on the Trace tab. The window below is displayed.
5. Core Clock: No need to enter anything. ULINK*pro* determines this automatically. Select the Trace Enable box.
6. In the Trace Port select Serial Wire Output – Manchester. Selecting UART/NRZ will cause an error.
7. Select Periodic and leave everything else at default. Periodic activates PC Samples.
8. Click on OK twice to return to the main µVision menu. SWV is now configured.

TIP: Sync Trace Port with 4 bit Data field sends the trace records out the 4 bit trace port rather than the single pin SWO. The Trace Port is faster and must be selected for ETM trace. It is available only with the ULINK*pro*.
We will examine this setting later.


Cortex-M Target Driver Setup

Debug | Trace | Flash Download

Core Clock: 56.200000 MHz
[✓] Trace Enable
[ ] ETM Trace Enable

Trace Port: Serial Wire Output - Manchester
- Sync Trace Port with 1-bit Data
- Sync Trace Port with 2-bit Data
- Sync Trace Port with 4-bit Data
- Serial Wire Output - Manchester
- Serial Wire Output - UART/NRZ

Timestamps: [✓] Enable Prescaler: 1

PC Sampling: Prescaler: 1024*16
[✓] Periodic Period: 291.530 us
[ ] on Data R/W Sample

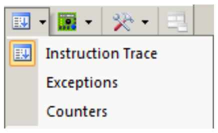
Trace Events:
[ ] CPI: Cycles per Instruction
[ ] EXC: Exception overhead
[ ] SLEEP: Sleep Cycles
[ ] LSU: Load Store Unit Cycles
[ ] FOLD: Folded Instructions
[✓] EXCTRC: Exception Tracing

ITM Stimulus Ports
Enable: 0xFFFFFFFF (Port 31 … 0 all checked)
Privilege: 0x00000008 Port 31..24 [✓] Port 23..16 [ ] Port 15..8 [ ] Port 7..0 [ ]

OK Cancel Help

Display Trace Records:

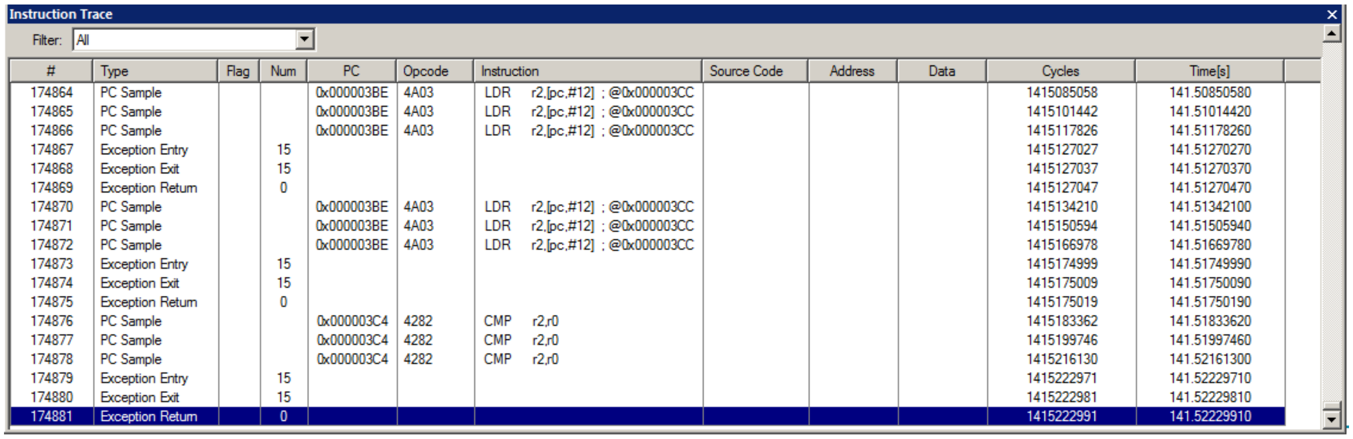
1. Enter Debug mode.
2. Click on the RUN icon.
3. Open Instruction Trace window by clicking on the small arrow beside the Trace icon:
4. The Race Records window will open and display Exceptions and PC Samples as shown below:



TIP: If you see the Num column contain any numbers other than 15 or 0, the trace is not configured correctly. The most probable cause is the Core Clock: frequency is wrong. Try 52.9 or 48 MHz.

TIP: The Instruction Trace window is different that the Trace Records window provided with the ULINK2. Note the disassembled instructions are displayed and if available, the source code is also displayed. Clicking on a PC Sample line will take you to that place in the source and disassembly windows. All the program counter values are displayed with ETM trace.

You cannot clear the Instruction Trace window by double-clicking inside it. To clear the trace, exit and re-enter debug mode.


Instruction Trace

Filter: All

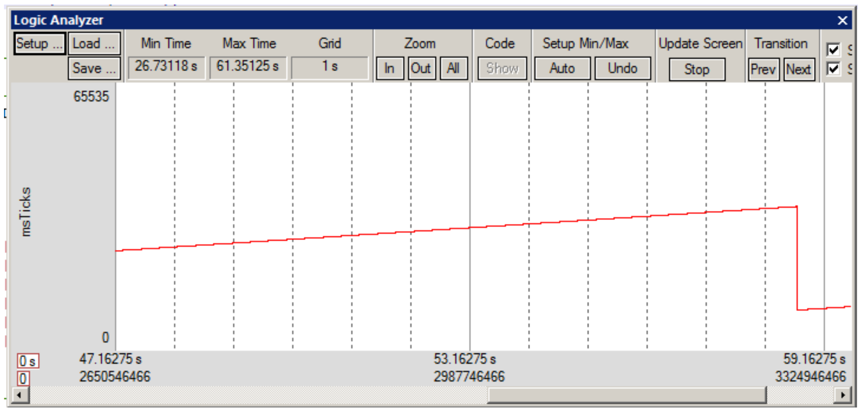
#	Type	Flag	Num	PC	Opcode	Instruction	Source Code	Address	Data	Cycles	Time[s]
174864	PC Sample			0x000003BE	4A03	LDR r2,[pc,#12] ; @0x000003CC				1415085058	141.50850580
174865	PC Sample			0x000003BE	4A03	LDR r2,[pc,#12] ; @0x000003CC				1415101442	141.51014420
174866	PC Sample			0x000003BE	4A03	LDR r2,[pc,#12] ; @0x000003CC				1415117826	141.51178260
174867	Exception Entry		15							1415127027	141.51270270
174868	Exception Exit		15							1415127037	141.51270370
174869	Exception Return		0							1415127047	141.51270470
174870	PC Sample			0x000003BE	4A03	LDR r2,[pc,#12] ; @0x000003CC				1415134210	141.51342100
174871	PC Sample			0x000003BE	4A03	LDR r2,[pc,#12] ; @0x000003CC				1415150594	141.51505940
174872	PC Sample			0x000003BE	4A03	LDR r2,[pc,#12] ; @0x000003CC				1415166978	141.51669780
174873	Exception Entry		15							1415174999	141.51749990
174874	Exception Exit		15							1415175009	141.51750090
174875	Exception Return		0							1415175019	141.51750190
174876	PC Sample			0x000003C4	4282	CMP r2,r0				1415183362	141.51833620
174877	PC Sample			0x000003C4	4282	CMP r2,r0				1415199746	141.51997460
174878	PC Sample			0x000003C4	4282	CMP r2,r0				1415216130	141.52161300
174879	Exception Entry		15							1415222971	141.52229710
174880	Exception Exit		15							1415222981	141.52229810
174881	Exception Return		0							1415222991	141.52229910

10) Using the Logic Analyzer (LA) with ULINK2 or ULINK-ME:

This example will use the ULINK2 with the Blinky example. Please connect a ULINK2 or ULKINK-ME to your Kinetis board and configure it for SWV trace. If you want to use a ULINK*pro* you will have to make appropriate modifications.

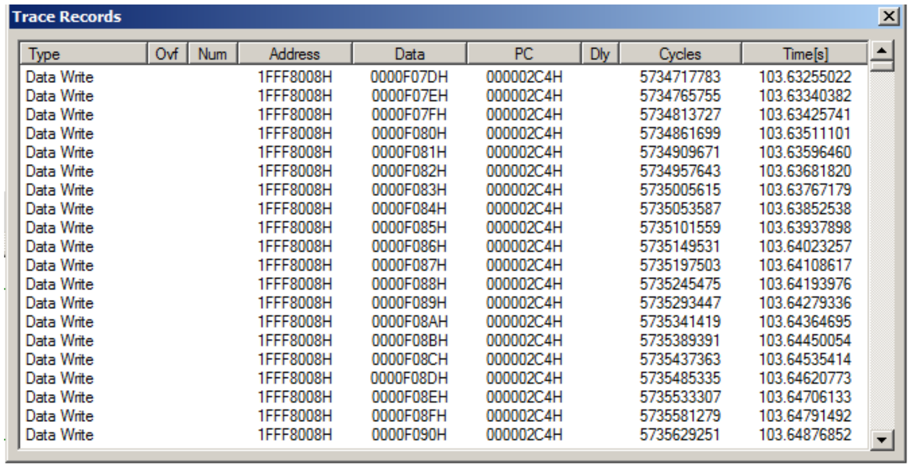
µVision has a graphical Logic Analyzer (LA) window. Up to four variables can be displayed in real-time using the Serial Wire Viewer in the Kinetis.

1. Enter debug mode .
2. Select Debug/Debug Settings and select the Trace tab.
3. Unselect Periodic and EXCTRC. This is to prevent overload on the SWO pin.
4. Run the program. . You can configure the LA while the program is running or stopped.
5. Open View/Analysis Windows and select Logic Analyzer or select the LA window on the toolbar.
6. Locate the volatile variable msTicks in Blinky.c. It is declared near line 19.
7. Block msTicks and drag it into the LA window and release it. Or click on Setup in the LA and enter it manually.
8. Click on Setup and set Max: in Display Range to 0xFFFF. Click on close. The LA is completely configured now.
9. Drag and drop msTicks into the Watch 1 window. It should be incrementing.
10. Adjust the Zoom OUT icon in the LA window to about 1 second or so to get a nice ramp as shown below.
11. In the Watch 1 window, double-click on the msTicks value and enter 0 and press Enter.
12. This value will be displayed in the LA window as shown here: You can enter any value into msTicks.



TIP: The Logic Analyzer can display static and global variables, structures and arrays. It can't see locals: just make them static. To see peripheral registers merely read or write to them and enter them into the Logic Analyzer.

1. Select Debug/Debug Settings and select the Trace tab.
2. Select On Data R/W Sample. Click OK.
3. Run the program. .
4. Open the Trace Records window.
5. The window similar below opens up:
6. The first line below says:
7. The instruction at 0x2C4 caused a write of data 0x0000_F07D to address 0x1FFF_8008 at the listed time in Cycles or Time.



Trace Records

Type	Ovf	Num	Address	Data	PC	Dly	Cycles	Time[s]
Data Write			1FFF8008H	0000F07DH	000002C4H		5734717783	103.63255022
Data Write			1FFF8008H	0000F07EH	000002C4H		5734765755	103.63340382
Data Write			1FFF8008H	0000F07FH	000002C4H		5734813727	103.63425741
Data Write			1FFF8008H	0000F080H	000002C4H		5734861699	103.63511101
Data Write			1FFF8008H	0000F081H	000002C4H		5734909671	103.63596460
Data Write			1FFF8008H	0000F082H	000002C4H		5734957643	103.63681820
Data Write			1FFF8008H	0000F083H	000002C4H		5735005615	103.63767179
Data Write			1FFF8008H	0000F084H	000002C4H		5735053587	103.63852538
Data Write			1FFF8008H	0000F085H	000002C4H		5735101559	103.63937898
Data Write			1FFF8008H	0000F086H	000002C4H		5735149531	103.64023257
Data Write			1FFF8008H	0000F087H	000002C4H		5735197503	103.64108617
Data Write			1FFF8008H	0000F088H	000002C4H		5735245475	103.64193976
Data Write			1FFF8008H	0000F089H	000002C4H		5735293447	103.64279336
Data Write			1FFF8008H	0000F08AH	000002C4H		5735341419	103.64364695
Data Write			1FFF8008H	0000F08BH	000002C4H		5735389391	103.64450054
Data Write			1FFF8008H	0000F08CH	000002C4H		5735437363	103.64535414
Data Write			1FFF8008H	0000F08DH	000002C4H		5735485335	103.64620773
Data Write			1FFF8008H	0000F08EH	000002C4H		5735533307	103.64706133
Data Write			1FFF8008H	0000F08FH	000002C4H		5735581279	103.64791492
Data Write			1FFF8008H	0000F090H	000002C4H		5735629251	103.64876852

TIP: The PC column is activated when you selected On Data R/W Sample in Step 2. You can leave this out to save bandwidth on the SWO pin.

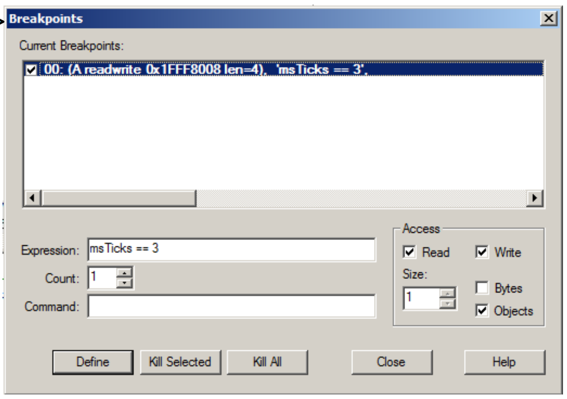
TIP: The ULINK*pro* will give a more sophisticated Instruction Trace window.

Watchpoints are described on the next page.

11) **Watchpoints:** *Conditional Breakpoints*

The Kinetis Cortex-M4 processors have four Watchpoints. Watchpoints can be thought of as conditional breakpoints. The Logic Analyzer uses watchpoints in its operations. This means in µVision you must have two variables free in the Logic Analyzer to use Watchpoints.

1. Using the example from the previous page, stop the program.
2. Click on Debug and select Breakpoint or press Ctrl-B.
3. The Trace does not need to be configured for Watchpoints. However, we will use it in this exercise.
4. In the Expression box enter: "**`msTicks`** == 3" without the quotes. Select both the Read and Write Access.
5. Click on Define and it will be accepted as shown here:
6. Click on Close.
7. Double-click in the Trace Records window to clear it.
8. Set msTicks in Watch 1 window to zero.
9. Click on RUN.
10. When msTicks equals 3, the program will stop. This is how a Watchpoint works.
11. You will see msTicks incremented in the Logic Analyzer as well as in the Watch window if you choose a watchpoint with a higher value.

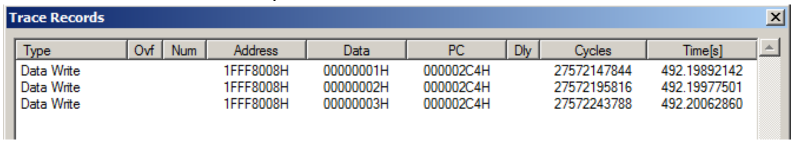


Current Breakpoints:

☑ 00: (A readwrite 0x1FFF8008 len=4), 'msTicks == 3',

Expression: msTicks == 3
Count: 1
Command:
Access: ☑ Read ☑ Write
Size: 1 ☐ Bytes ☑ Objects

12. Note the three data writes in the Trace Records window shown below. 1, 2 and 3 in the Data column. Plus the address the data written to and the PC of the write instruction. This is with a ULINK2 or ULINK-ME. The ULINK*pro* will display a different window.
13. There are other types of expressions you can enter and are detailed in the Help button in the Breakpoints window.



Type	Ovf	Num	Address	Data	PC	Dly	Cycles	Time[s]
Data Write			1FFF8008H	00000001H	000002C4H		27572147844	492.19892142
Data Write			1FFF8008H	00000002H	000002C4H		27572195816	492.19977501
Data Write			1FFF8008H	00000003H	000002C4H		27572243788	492.20062860

14. To repeat this exercise, set msTicks to a number other than 3 or disable the watchpoint.
15. When finished leave Debug mode.

TIP: You cannot set Watchpoints on-the-fly while the program is running like you can with hardware breakpoints.

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 try the next TIP:

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

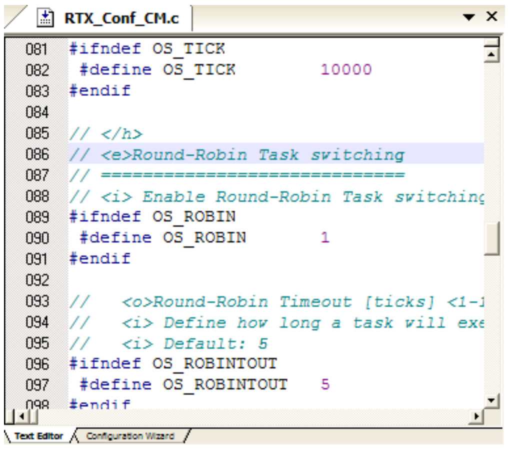
12) *RTX_Blinky* Example Program with Keil RTX RTOS: A Stepper Motor example

Keil provides RTX, a full feature RTOS. RTX is included for no charge as part of the Keil MDK full tool suite. It can have up to 255 tasks and no royalty payments are required. If source code is required, this is included in the Keil RL-ARM™ Real-Time Library which also includes USB, CAN, TCP/IP networking and a Flash File system. This example explores the RTOS project. Keil will work with any RTOS. An RTOS is just a set of C functions that gets compiled with your project.

1. Start µVision by clicking on its icon on your Desktop if it is not already running.
2. Select Project/Open Project.
3. Open the file C:\Keil\ARM\Boards\Freescale\TWR-K60N512\RTX_Blinky\Blinky.uvproj.
4. Since RTX_Blinky uses a ULINKpro as default: if you are using a ULINK2 please configure it as described on page 4 under **2) ULINK2 or ULINK-ME and µVision Configuration:** You do not need to configure the trace yet. You only have to do this once – it will be saved in the project file. You can also make a new target configuration.
5. Compile the source files by clicking on the Rebuild icon. . They will compile with no errors or warnings.
6. To program the Flash manually, click on the Load icon. . A progress bar will be at the bottom left.
7. Enter the Debug mode by clicking on the debug icon and click on the RUN icon.
8. The LEDs will blink indicating the four waveforms of a stepper motor driver changing. Click on STOP .

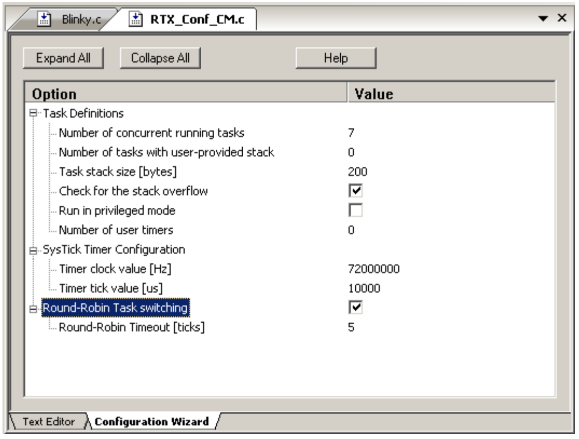
The Configuration Wizard for RTX:

1. Click on the RTX_Conf_CM.c source file tab as shown below on the left. You can open it with File/Open.
2. Click on Configuration Wizard at the bottom and your view will change to the Configuration Wizard.
3. Open up the individual directories to show the various configuration items available.
4. See how easy it is to modify these settings here as opposed to finding and changing entries in the source code.
5. This is a great feature as it is much easier changing items here than in the source code.
6. You can create Configuration Wizards in any source file with the scripting language as used in the Text Editor.
7. This scripting language is shown below in the Text Editor as comments starting such as a </h> or <i>.
8. The new µVision4 System Viewer windows are created in a similar fashion. Available Q4 2010.



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

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

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

Text Editor: Source Code

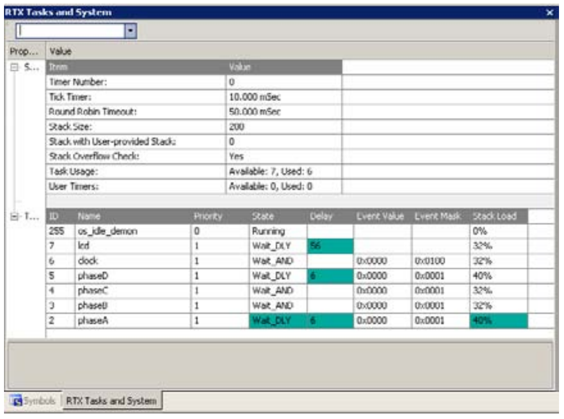


Option	Value
Task Definitions	
Number of concurrent running tasks	7
Number of tasks with user-provided stack	0
Task stack size [bytes]	200
Check for the stack overflow	☑
Run in privileged mode	☐
Number of user timers	0
SysTick Timer Configuration	
Timer clock value [Hz]	72000000
Timer tick value [us]	10000
Round-Robin Task switching	☑
Round-Robin Timeout [ticks]	5

Configuration Wizard

13) RTX Kernel Awareness using Serial Wire Viewer

Users often want to know the number of the current operating task and the status of the other tasks. This information is usually stored in a structure or memory area by the RTOS. Keil provides a Task Aware window for RTX. Other RTOS companies also provide awareness plug-ins for µVision.



RTX Tasks and System

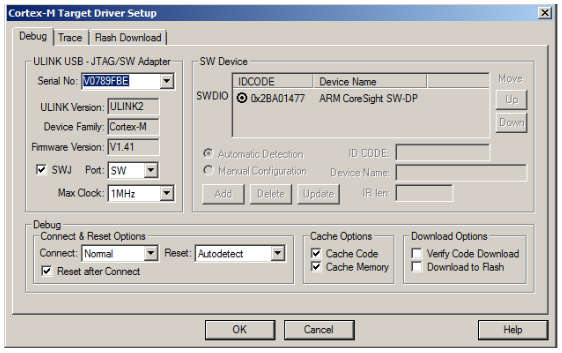
Item	Value
Timer Number:	0
Tick Timer:	10.000 mSec
Round Robin Timeout:	50.000 mSec
Stack Size:	200
Stack with User-provided Stack:	0
Stack Overflow Check:	Yes
Task Usage:	Available: 7, Used: 6
User Timers:	Available: 0, Used: 0

ID	Name	Priority	State	Delay	Event Value	Event Mask	Stack Load
255	os_idle_demon	0	Running				0%
7	lcd	1	Wait_DLY	56			32%
6	clock	1	Wait_AND		0x0000	0x0100	32%
5	phaseD	1	Wait_DLY	6	0x0000	0x0001	40%
4	phaseC	1	Wait_AND		0x0000	0x0001	32%
3	phaseB	1	Wait_AND		0x0000	0x0001	32%
2	phaseA	1	Wait_DLY	6	0x0000	0x0001	40%

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

RTX Viewer: Configuring Serial Wire Viewer (SWV):

We must activate Serial Wire Viewer to get the Event Viewer working.



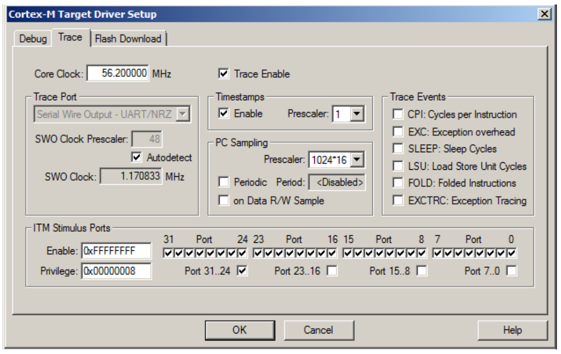
Cortex-M Target Driver Setup — Debug | Trace | Flash Download

ULINK USB - JTAG/SW Adapter: Serial No: V0789FBE; ULINK Version: ULINK2; Device Family: Cortex-M; Firmware Version: V1.41; ☑ SWJ Port: SW; Max Clock: 1MHz

SW Device: SWDIO — IDCODE 0x2BA01477, Device Name ARM CoreSight SW-DP; Automatic Detection; Manual Configuration; ID CODE:; Device Name:; IR len:

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

1. Stop the CPU and exit debug mode.
2. Click on the Options icon next to the target box.
3. Select the Debug tab and then click the Settings box next to ULINK Cortex Debugger dialog.
4. In the Debug window as shown here, make sure SWJ is checked and Port: is set to SW and not JTAG.
5. Click on the Trace tab to open the Trace window.
6. Set Core Clock: to 56.2 MHz and select Trace Enable.
7. Unselect the Periodic and EXCTRC boxes as shown here:
8. ITM Stimulus Port 31 must be checked. This is the method the RTX Viewer gets the kernel awareness information out to be displayed in the Event Viewer. It is slightly intrusive.



Cortex-M Target Driver Setup — Debug | Trace | Flash Download

Core Clock: 56.200000 MHz; ☑ Trace Enable

Trace Port: Serial Wire Output - UART/NRZ; SWO Clock Prescaler: 48; ☑ Autodetect; SWO Clock: 1.170833 MHz

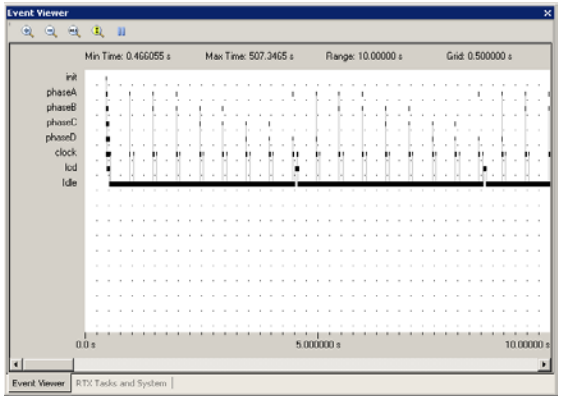
Timestamps: ☑ Enable; Prescaler: 1

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

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

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

9. Click on OK twice to return to µVision.
   The Serial Wire Viewer is now configured in µVision.
10. Enter Debug mode and click on RUN to start the program.
11. Select "Tasks and System" tab: note the display is updated.
12. Click on the Event Viewer tab.
13. This window displays task events in a graphical format as shown in the RTX Kernel window below. You probably have to change the Range to about 5 seconds by clicking on the ALL and then the + and – icons.



Event Viewer

Min Time: 0.466055 s   Max Time: 507.3465 s   Range: 10.00000 s   Grid: 0.500000 s

TIP: View/Periodic Window Update must be selected !

TIP: If Event Viewer doesn't work, open up the Trace Records and confirm there is good ITM frames present.

Cortex-M4 Alert: µVision will update all RTX information in real-time on a target board due to its read/write capabilities as already described. The Event Viewer uses ITM and is slightly intrusive.

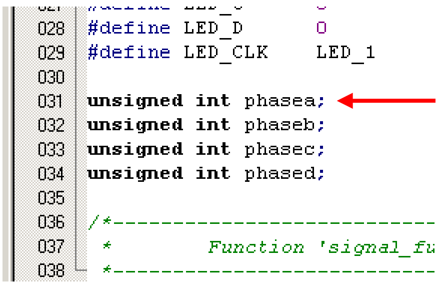
You will not have to stop the program to view this data. No CPU cycles are used. Your program runs at full speed. No instrumentation code need be inserted into your source. You will find this feature very useful !

TIP: You can use a ULINK2, ULINK-ME, ULINK*pro* or J-Link for these RTX Kernel Awareness windows.

14) Logic Analyzer Window: View variables real-time in a graphical format:

µVision has a graphical Logic Analyzer window. Up to four variables can be displayed in real-time using the Serial Wire Viewer in the Kinetis. RTX_Blinky uses four tasks to create the waveforms. We will graph these four waveforms.

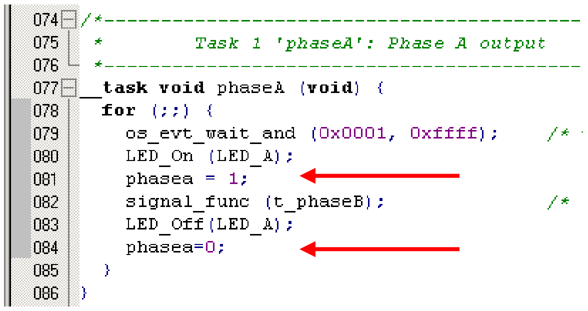
1. Close the RTX Viewer windows. Stop the program and exit debug mode.
2. Add 4 global variables **`unsigned int phasea`** through **`unsigned int phased`** to Blinky.c as shown here:



```
028 #define LED_D       0
029 #define LED_CLK     LED_1
030
031 unsigned int phasea;
032 unsigned int phaseb;
033 unsigned int phasec;
034 unsigned int phased;
035
036 /*---------------------------
037  *          Function 'signal_fu
038  *---------------------------
```

3. Add 2 lines to each of the four tasks Task1 through Task4 in Blinky.c as shown below: **phasea=1;** and **phasea=0;** :the first two lines are shown added at lines 081 and 084 (just after LED_On and LED_Off function calls). For each of the four tasks, add the corresponding variable assignment statements phasea, phaseb, phasec and phased.
4. We do this because in this simple program there are not enough suitable variables to connect to the Logic Analyzer.

TIP: The Logic Analyzer can display static and global variables, structures and arrays. It can't see locals: just make them static. To see peripheral registers merely read or write to them and enter them into the Logic Analyzer.

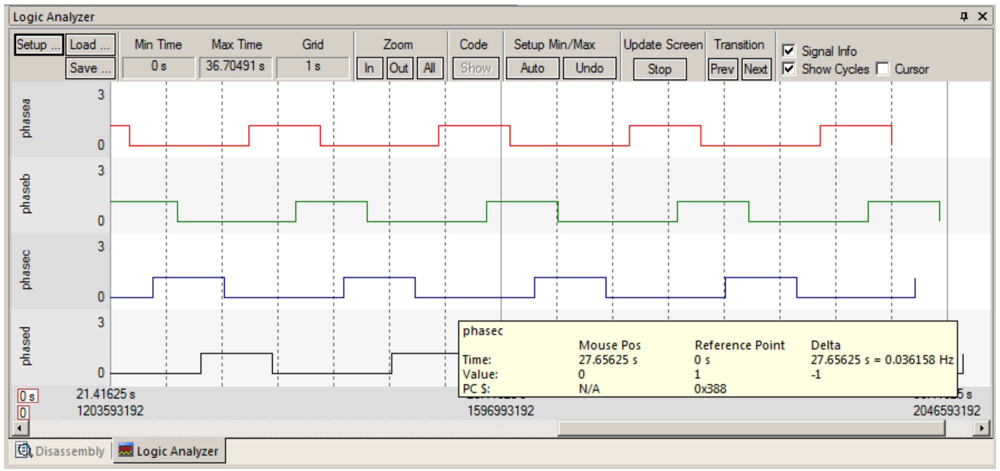


```
074 /*-----------------------------------------------
075  *          Task 1 'phaseA': Phase A output
076  *-----------------------------------------------
077 __task void phaseA (void) {
078   for (;;) {
079     os_evt_wait_and (0x0001, 0xffff);      /*
080     LED_On (LED_A);
081     phasea = 1;
082     signal_func (t_phaseB);                /*
083     LED_Off(LED_A);
084     phasea=0;
085   }
086 }
```

5. Rebuild the project. Program the Flash .
6. Enter debug mode .
7. You can run the program at this point.
8. Open View/Analysis Windows and select Logic Analyzer or select the LA window on the toolbar.

Enter the Variables into the Logic Analyzer:

9. Click on the Blinky.c tab. Block **`phasea`**, click, hold and drag up to the Logic Analyzer tab (don't let go yet!)
10. When it opens, bring the mouse down anywhere into the Logic Analyzer window and release.
11. Repeat for **phaseb, phasec and phased.** These variables will be listed on the left side of the LA window as shown. Now we have to adjust the scaling.
12. Click on the Setup icon and click on each of the four variables and set Max. in the Display Range: to 0x3.
13. Click on Close to go back to the LA window.
14. Using the OUT and In buttons set the range to 20 seconds. Move the scrolling bar to the far right if needed.
15. You will see the following waveforms appear. Click to mark a place See 252 s below. Place the cursor on one of the waveforms and get timing and other information as shown in the inserted box labeled phasec:



Min Time: 0 s
Max Time: 36.70491 s
Grid: 1 s
[✓] Signal Info
[✓] Show Cycles
[ ] Cursor

0 s 21.41625 s
0 1203593192 1596993192 2046593192

phasec

	Mouse Pos	Reference Point	Delta
Time:	27.65625 s	0 s	27.65625 s = 0.036158 Hz
Value:	0	1	-1
PC $:	N/A	0x388	

TIP: You can also enter these variables into the Watch and Memory windows to display and change them in real-time.

15) Serial Wire Viewer (SWV) and how to use it:

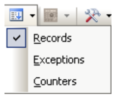
a) Data Reads and Writes: (Note: Data Reads but not Writes are disabled in the current version of µVision**).**

You have configured Serial Wire Viewer (SWV) in Section 6 under **RTX Viewer: Configuring the Serial Wire Viewer:**

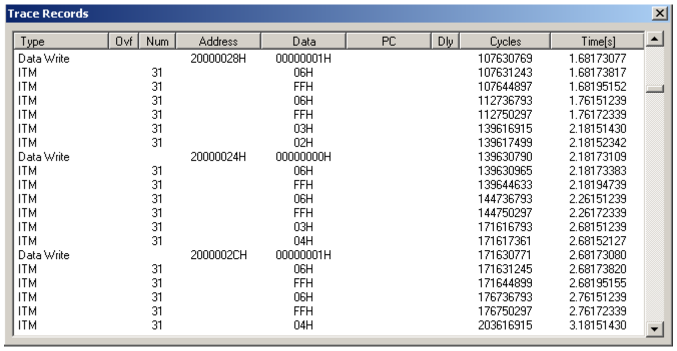
Now we will examine some of the features available to you. SWV works with µVision and a ULINK2, ULINK-ME, ULINK*pro* or a Segger J-Link V6 or higher. SWV is included with MDK and no other equipment must be purchased.

Everything shown here is done without stealing any CPU cycles and is completely non-intrusive. A user program runs at full speed and needs no code stubs or instrumentation software added to your programs.

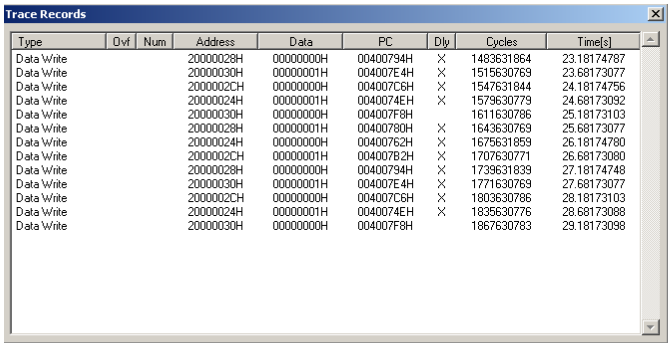
1. Use RTX_Blinky from the last exercise. Enter Debug mode and run the program if not already running.
2. Select View/Trace/Records or click on the Trace icon and select Records.



3. The Trace Records window will open up as shown here:
4. The ITM frames are the data from the RTX Kernel Viewer which uses Port 31 as shown under Num. To turn this off select Debug/Debug Settings and click on the Trace tab. Unselect ITM Stimulus Port 31. TIP: Port 0 is used for Debug `printf` Viewer.
5. Unselect EXCTRC and Periodic.
6. Select On Data R/W Sample.
7. Click on OK to return.
8. Click on the RUN icon.
9. Double-click anywhere in the Trace records window to clear it.
10. Only Data Writes will appear now.
    TIP: You could have right clicked on the Trace Records window to filter the ITM frames out.


Trace Records

Type	Ovf	Num	Address	Data	PC	Dly	Cycles	Time[s]
Data Write			20000028H	00000001H			107630769	1.68173077
ITM		31		06H			107631243	1.68173817
ITM		31		FFH			107644897	1.68195152
ITM		31		06H			112736793	1.76151239
ITM		31		FFH			112750297	1.76172339
ITM		31		03H			139616915	2.18151430
ITM		31		02H			139617499	2.18152342
Data Write			20000024H	00000000H			139630790	2.18173109
ITM		31		06H			139630965	2.18173383
ITM		31		FFH			139644633	2.18194739
ITM		31		06H			144736793	2.26151239
ITM		31		FFH			144750297	2.26172339
ITM		31		03H			171616793	2.68151239
ITM		31		04H			171617361	2.68152127
Data Write			2000002CH	00000001H			171630771	2.68173080
ITM		31		06H			171631245	2.68173820
ITM		31		FFH			171644899	2.68195155
ITM		31		06H			176736793	2.76151239
ITM		31		FFH			176750297	2.76172339
ITM		31		04H			203616915	3.18151430


Trace Records

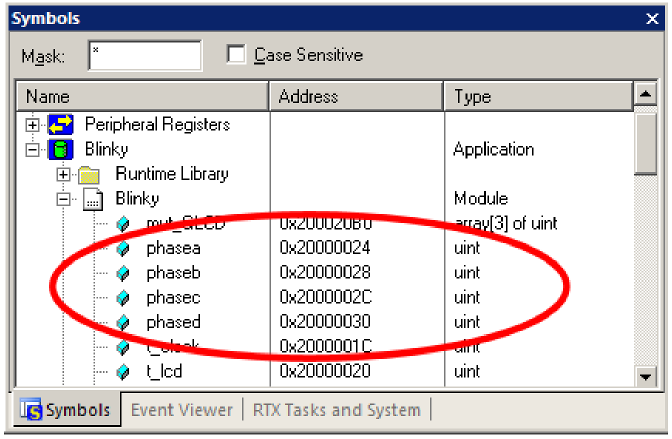
Type	Ovf	Num	Address	Data	PC	Dly	Cycles	Time[s]
Data Write			20000028H	00000000H	00400794H	X	1483631864	23.18174787
Data Write			20000030H	00000001H	004007E4H	X	1515630769	23.68173077
Data Write			2000002CH	00000000H	004007C6H	X	1547631844	24.18174756
Data Write			20000024H	00000001H	0040074EH	X	1579630779	24.68173092
Data Write			20000030H	00000000H	004007F8H		1611630786	25.18173103
Data Write			20000028H	00000001H	00400780H	X	1643630769	25.68173077
Data Write			20000024H	00000000H	00400762H	X	1675631859	26.18174780
Data Write			2000002CH	00000001H	004007B2H	X	1707630771	26.68173080
Data Write			20000028H	00000000H	00400794H	X	1739631839	27.18174748
Data Write			20000030H	00000001H	004007E4H	X	1771630769	27.68173077
Data Write			2000002CH	00000000H	004007C6H	X	1803630786	28.18173103
Data Write			20000024H	00000001H	0040074EH	X	1835630776	28.68173088
Data Write			20000030H	00000000H	004007F8H		1867630783	29.18173098

What is happening here ?

1. When variables are entered in the Logic Analyzer (remember phasea through phased ?), the reads and/or writes will appear in Trace Records.
2. The Address column shows where the four variables are located.
3. The Data column displays the data values written to phasea through phased.
4. PC is the address of the instruction causing the writes. You activated it by selecting On Data R/W Sample in the Trace configuration window.
5. The Cycles and Time(s) columns are when these events happened.

TIP: You can have up to four variables in the Logic Analyzer and subsequently displayed in the Trace Records window.

TIP: If you select View/Symbol Window you can see where the addresses of the variables are.



Name	Address	Type
Peripheral Registers		
Blinky		Application
Runtime Library		
Blinky		Module
mut_GLCD	0x200020B0	array[3] of uint
phasea	0x20000024	uint
phaseb	0x20000028	uint
phasec	0x2000002C	uint
phased	0x20000030	uint
t_clock	0x2000001C	uint
t_lcd	0x20000020	uint

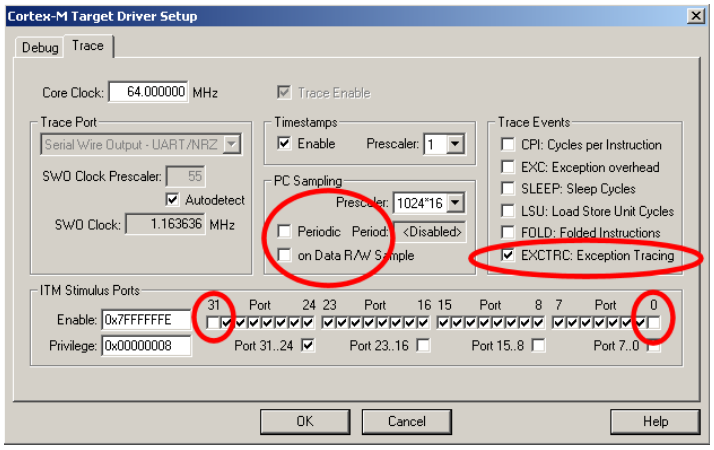
Note: You must have Browser Information selected in the Options for Target/Output tab to use the Symbol Browser.

TIP: ULINK*pro*, J-Link and SAM-ICE adapters display the trace frames in a different style trace window.

b) Exceptions and Interrupts:

The Kinetis family using the Cortex-M4 processor has many interrupts and it can be difficult to determine when they are being activated. SWV on the Cortex-M4 processor makes this easy.

1. Open Debug/Debug Settings and select the Trace tab.
2. Unselect On Data R/W Sample, PC Sample and ITM Ports 31 and 0. (this is to minimize overloading the SWO port)
3. Select EXCTRC as shown here:
4. Click OK twice.
5. The Trace Records should still be open. Double click on it to clear it.
6. Click RUN to start the program.
7. You will see a window similar to the one below with Exceptions frames displayed.


Cortex-M Target Driver Setup

Debug | Trace

Core Clock: 64.000000 MHz — [✓] Trace Enable

Trace Port: Serial Wire Output - UART/NRZ; SWO Clock Prescaler: 55; [✓] Autodetect; SWO Clock: 1.163636 MHz

Timestamps: [✓] Enable; Prescaler: 1

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

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

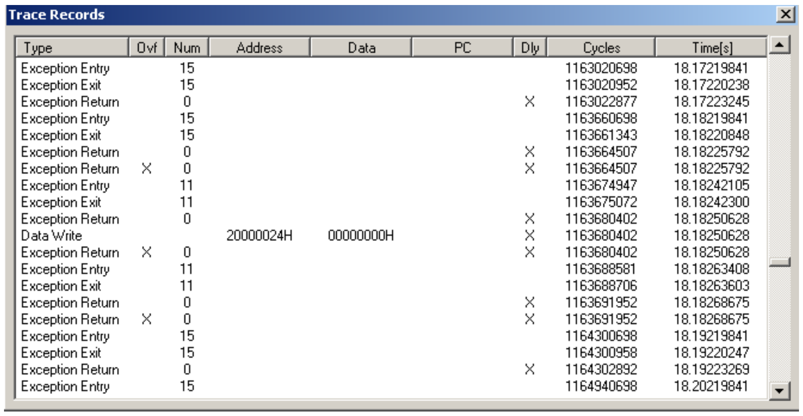
ITM Stimulus Ports: Enable: 0x7FFFFFFE; Privilege: 0x00000008; Port 31..24 [✓]; Port 23..16 [ ]; Port 15..8 [ ]; Port 7..0 [ ]

What Is Happening ?

1. You can see two exceptions happening.

- **Entry:** when the exception enters.
- **Exit:** When it exits or returns.
- **Return:** When all the exceptions have returned and is useful to detect tail-chaining.

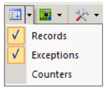
2. Num 11 is SVCall from the RTX calls.
3. Num 15 is the Systick timer.
4. In my example you can see one data write from the Logic Analyzer.
5. Note everything is timestamped.
6. The "X" in Ovf is an overflow and some data was lost. The "X" in Dly means the timestamps are delayed because too much information is being fed out the SWO pin.


Trace Records

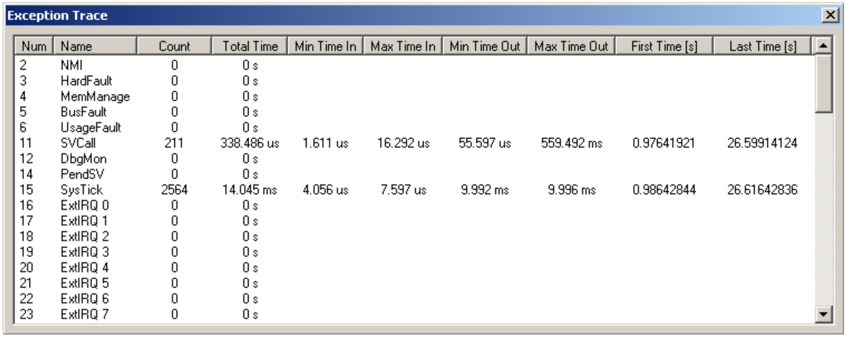
Type	Ovf	Num	Address	Data	PC	Dly	Cycles	Time[s]
Exception Entry		15					1163020698	18.17219841
Exception Exit		15					1163020952	18.17220238
Exception Return		0				X	1163022877	18.17223245
Exception Entry		15					1163660698	18.18219841
Exception Exit		15					1163661343	18.18220848
Exception Return		0				X	1163664507	18.18225792
Exception Return	X	0				X	1163664507	18.18225792
Exception Entry		11					1163674947	18.18242105
Exception Exit		11					1163675072	18.18242300
Exception Return		0				X	1163680402	18.18250628
Data Write			20000024H	00000000H		X	1163680402	18.18250628
Exception Return	X	0				X	1163680402	18.18250628
Exception Entry		11					1163688581	18.18263408
Exception Exit		11					1163688706	18.18263603
Exception Return		0				X	1163691952	18.18268675
Exception Return	X	0				X	1163691952	18.18268675
Exception Entry		15					1164300698	18.19219841
Exception Exit		15					1164300958	18.19220247
Exception Return		0				X	1164302892	18.19223269
Exception Entry		15					1164940698	18.20219841

TIP: The SWO pin is one pin on the Cortex-M4 family processors that all SWV information is fed out. There are limitations on how much information we can feed out this one pin. These exceptions are happening at a very fast rate.

1. Select View/Trace/Exceptions or click on the Trace icon and select Exceptions.



2. The next window opens up and more information about the exceptions is displayed as shown.
3. Note the number of times these have happened under Count. This is very useful information in case interrupts come too fast or slow.
4. ExtIRQ are the peripheral interrupts.
5. You can clear this trace window by double-clicking on it.
6. All this information is displayed in real-time and without stealing CPU cycles !


Exception Trace

Num	Name	Count	Total Time	Min Time In	Max Time In	Min Time Out	Max Time Out	First Time [s]	Last Time [s]
2	NMI	0	0 s						
3	HardFault	0	0 s						
4	MemManage	0	0 s						
5	BusFault	0	0 s						
6	UsageFault	0	0 s						
11	SVCall	211	338.486 us	1.611 us	16.292 us	55.597 us	559.492 ms	0.97641921	26.59914124
12	DbgMon	0	0 s						
14	PendSV	0	0 s						
15	SysTick	2564	14.045 ms	4.056 us	7.597 us	9.992 ms	9.996 ms	0.98642844	26.61642836
16	ExtIRQ 0	0	0 s						
17	ExtIRQ 1	0	0 s						
18	ExtIRQ 2	0	0 s						
19	ExtIRQ 3	0	0 s						
20	ExtIRQ 4	0	0 s						
21	ExtIRQ 5	0	0 s						
22	ExtIRQ 6	0	0 s						
23	ExtIRQ 7	0	0 s						

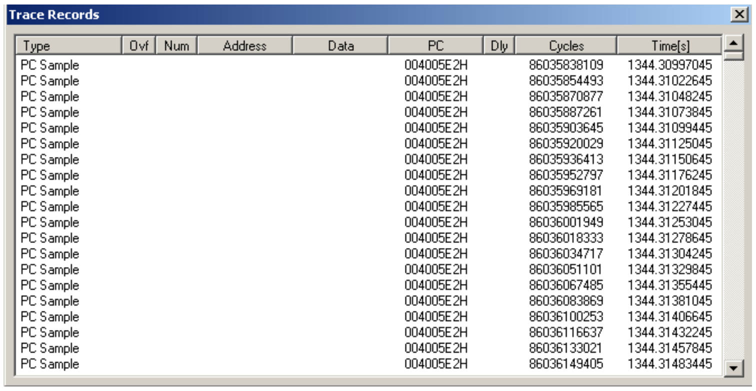
TIP: Num is the exception number: RESET is 1. External interrupts (ExtIRQ), which are normally attached to peripherals, start at Num 16. For example, Num 41 is also known as 41-16 = External IRQ 25. Num 16 = 16 – 16 = ExtIRQ 0.

c) PC Samples:

Serial Wire Viewer can display a sampling of the program counter. If you need to see all the PC values, use the ETM trace with a Keil ULINK*pro*. This also provides Code Coverage, Execution Profiling and Performance Analysis.

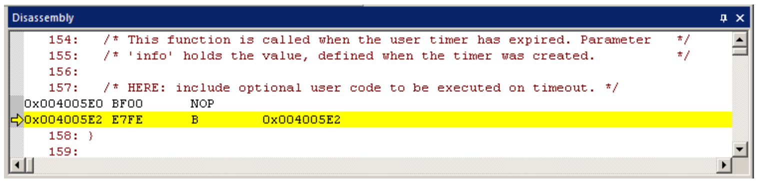
SWV can display at best every 64th instruction but usually every 16,384 is more common. It is best to keep this number as high as possible to avoid overloading the Serial Wire Output (SWO) pin. This is easily set in the Trace configuration.

1. Open Debug/Debug Settings and select the Trace tab.
2. Unselect EXCTRC, On Data R/W Sample and select Periodic in the PC Sampling area.
3. Click on OK twice to return to the main screen.
4. Close the Exception Trace window and leave Trace Records open. Double-click to clear it.
5. Click on RUN and this window opens:
6. Most of the PC Samples are 0x0040_05E2 which is a branch to itself in a loop forever routine.
7. Stop the program and the Disassembly window will show this Branch:
8. Not all the PCs will be captured. Still, PC Samples can give you some idea of where your program is; especially if it is caught in a tight loop like in this case.
9. **Note:** you can get different PC values depending on the optimization level set in µVision.
10. Set a breakpoint in one of the tasks.
11. Run the program and when the breakpoint is hit, you might see another address at the bottom of the Trace Records window. See the screen below:
12. Scroll to the bottom of the Trace Records window and you might see the correct PC value displayed. Usually, it will be a different PC depending on when the sampling took place.
13. Remove the breakpoint for the next step.



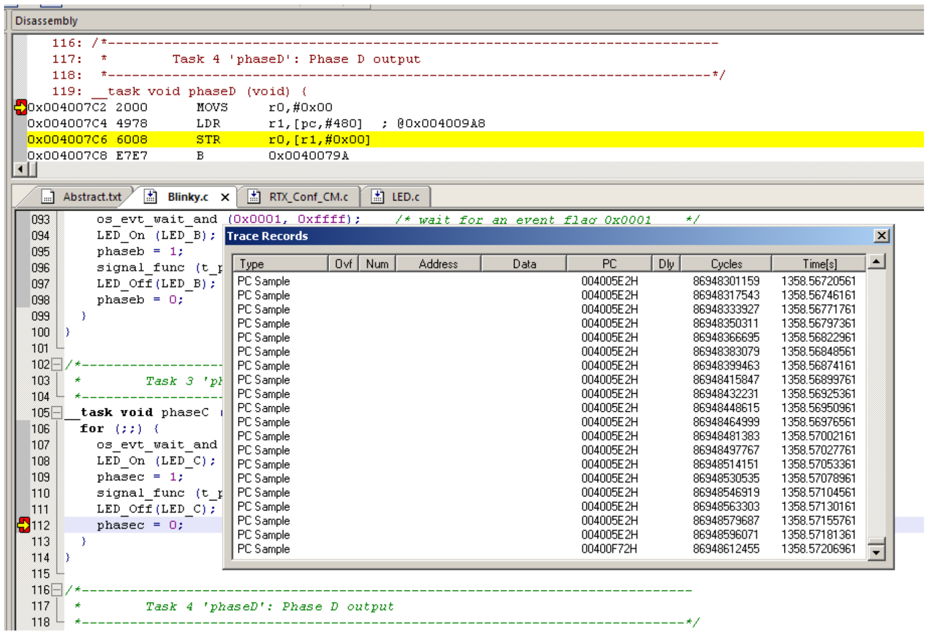
Trace Records

Type	Ovf	Num	Address	Data	PC	Dly	Cycles	Time[s]
PC Sample					004005E2H		86035838109	1344.30997045
PC Sample					004005E2H		86035854493	1344.31022645
PC Sample					004005E2H		86035870877	1344.31048245
PC Sample					004005E2H		86035887261	1344.31073845
PC Sample					004005E2H		86035903645	1344.31099445
PC Sample					004005E2H		86035920029	1344.31125045
PC Sample					004005E2H		86035936413	1344.31150645
PC Sample					004005E2H		86035952797	1344.31176245
PC Sample					004005E2H		86035969181	1344.31201845
PC Sample					004005E2H		86035985565	1344.31227445
PC Sample					004005E2H		86036001949	1344.31253045
PC Sample					004005E2H		86036018333	1344.31278645
PC Sample					004005E2H		86036034717	1344.31304245
PC Sample					004005E2H		86036051101	1344.31329845
PC Sample					004005E2H		86036067485	1344.31355445
PC Sample					004005E2H		86036083869	1344.31381045
PC Sample					004005E2H		86036100253	1344.31406645
PC Sample					004005E2H		86036116637	1344.31432245
PC Sample					004005E2H		86036133021	1344.31457845
PC Sample					004005E2H		86036149405	1344.31483445



Disassembly

```
   154:     /* This function is called when the user timer has expired. Parameter    */
   155:     /* 'info' holds the value, defined when the timer was created.            */
   156:
   157:     /* HERE: include optional user code to be executed on timeout. */
0x004005E0 BF00      NOP
0x004005E2 E7FE      B        0x004005E2
   158: }
   159:
```



Disassembly

```
   116: /*-----------------------------------------------------------------------------
   117:  *        Task 4 'phaseD': Phase D output
   118:  *---------------------------------------------------------------------------*/
   119: __task void phaseD (void) {
0x004007C2 2000      MOVS     r0,#0x00
0x004007C4 4978      LDR      r1,[pc,#480]  ; @0x004009A8
0x004007C6 6008      STR      r0,[r1,#0x00]
0x004007C8 E7E7      B        0x0040079A
```

```
    os_evt_wait_and (0x0001, 0xffff);    /* wait for an event flag 0x0001     */
    LED_On (LED_B);
    phaseb = 1;
    signal_func (t_p
    LED_Off(LED_B);
    phaseb = 0;
  }
}

/*-------------------
 *        Task 3 'ph
 *-------------------
__task void phaseC (
  for (;;) {
    os_evt_wait_and
    LED_On (LED_C);
    phasec = 1;
    signal_func (t_p
    LED_Off(LED_C);
    phasec = 0;
  }
}

/*-----------------------------------------------------------------------------
 *        Task 4 'phaseD': Phase D output
 *---------------------------------------------------------------------------*/
```

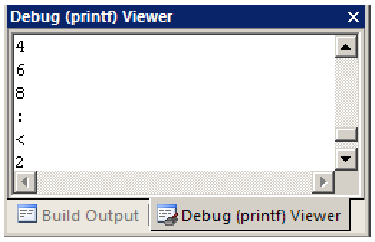
Trace Records

Type	Ovf	Num	Address	Data	PC	Dly	Cycles	Time[s]
PC Sample					004005E2H		86948301159	1358.56720561
PC Sample					004005E2H		86948317543	1358.56746161
PC Sample					004005E2H		86948333927	1358.56771761
PC Sample					004005E2H		86948350311	1358.56797361
PC Sample					004005E2H		86948366695	1358.56822961
PC Sample					004005E2H		86948383079	1358.56848561
PC Sample					004005E2H		86948399463	1358.56874161
PC Sample					004005E2H		86948415847	1358.56899761
PC Sample					004005E2H		86948432231	1358.56925361
PC Sample					004005E2H		86948448615	1358.56950961
PC Sample					004005E2H		86948464999	1358.56976561
PC Sample					004005E2H		86948481383	1358.57002161
PC Sample					004005E2H		86948497767	1358.57027761
PC Sample					004005E2H		86948514151	1358.57053361
PC Sample					004005E2H		86948530535	1358.57078961
PC Sample					004005E2H		86948546919	1358.57104561
PC Sample					004005E2H		86948563303	1358.57130161
PC Sample					004005E2H		86948579687	1358.57155761
PC Sample					004005E2H		86948596071	1358.57181361
PC Sample					00400F72H		86948612455	1358.57206961

16) ITM (Instruction Trace Macrocell)

Recall that we showed you can display information about the RTOS in real-time using the RTX Viewer. This is done through ITM Stimulus Port 31. ITM Port 0 is available for a *printf* type of instrumentation that requires minimal user code. After the write to the ITM port, zero CPU cycles are required to get the data out of the processor and into µVision for display in the Debug (printf) Viewer window.

1. Open the project Blinky.uvproj (not RTX Blinky).
2. Add this code to Blinky.c. A good place is near line 16, just after the declaration of **`count`**.
```
#define ITM_Port8(n)   (*((volatile unsigned char *)(0xE0000000+4*n)))
```
3. In the main function in Blinky.c right after the second Delay(500) enter these lines after near line 61:
```
ITM_Port8(0) = count + 0x30;     /*  displays count value: +0x30 converts to ASCII */
while (ITM_Port8(0) == 0);
ITM_Port8(0) = 0x0D;
while (ITM_Port8(0) == 0);
ITM_Port8(0) = 0x0A;
```
4. Rebuild the source files, program the Flash memory and enter debug mode.
5. Open Debug/Debug Settings and select the Trace tab.
6. Unselect On Data R/W Sample, PC Sample and ITM Port 31. (this is to help not overload the SWO port)
7. Select EXCTRC and ITM Port 0. ITM Stimulus Port "0" enables the Debug (prinftf) Viewer.
8. Click OK twice.
9. Click on View/Serial Windows and select Debug (printf) Viewer and click on RUN.
10. In the Debug (printf) Viewer you will see the ASCII value of count appear.



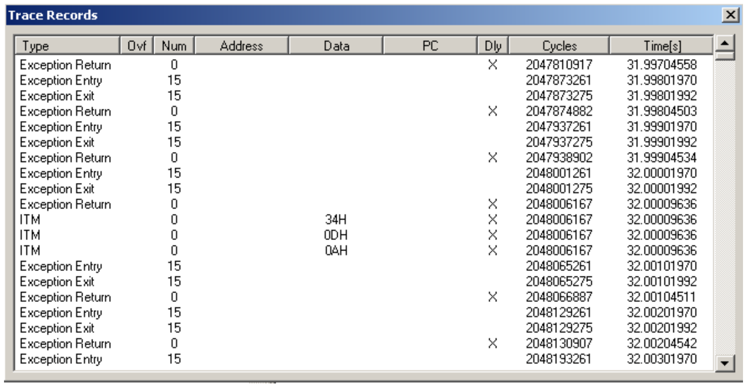
```
4
6
8
:
<
2
```

Trace Records

1. Open the Trace Records if not already open. Double click on it to clear it.
2. You will see a window such as the one below with ITM and Exception frames.

What Is This ?

1. You can see Exception 15 Entry, Exit, Return and the three ITM writes. You probably have to scroll down.
2. ITM 0 frames (Num column) are our ASCII characters from **`count`** with carriage return (0D) and line feed (0A) as displayed the Data column.
3. All these are timestamped in both CPU cycles and time in seconds.
4. Note the "X" in the Dly column. This means the timestamps might not be correct due to SWO pin overload.
5. Right click in the Trace Records window and deselect Exceptions. Now you will see only the ITM writes.



Type	Ovf	Num	Address	Data	PC	Dly	Cycles	Time[s]
Exception Return		0				X	2047810917	31.99704558
Exception Entry		15					2047873261	31.99801970
Exception Exit		15					2047873275	31.99801992
Exception Return		0				X	2047874882	31.99804503
Exception Entry		15					2047937261	31.99901970
Exception Exit		15					2047937275	31.99901992
Exception Return		0				X	2047938902	31.99904534
Exception Entry		15					2048001261	32.00001970
Exception Exit		15					2048001275	32.00001992
Exception Return		0				X	2048006167	32.00009636
ITM		0		34H		X	2048006167	32.00009636
ITM		0		0DH		X	2048006167	32.00009636
ITM		0		0AH		X	2048006167	32.00009636
Exception Entry		15					2048065261	32.00101970
Exception Exit		15					2048065275	32.00101992
Exception Return		0				X	2048066887	32.00104511
Exception Entry		15					2048129261	32.00201970
Exception Exit		15					2048129275	32.00201992
Exception Return		0				X	2048130907	32.00204542
Exception Entry		15					2048193261	32.00301970

ITM Conclusion

The writes to ITM Stimulus Port 0 are intrusive and are usually one cycle. It takes no CPU cycles to get the data out the Kinetis processor via the Serial Wire Output pin.

TIP: It is important to select as few options in the Trace configuration as possible to avoid overloading the SWO pin. Enter only those features that you really need.

Super TIP: ITM_SendChar is a useful function you can use to send characters. It is found in the header core.CM3.h.

17) Serial Wire Viewer Summary:

Serial Wire Viewer can see:

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

Serial Wire Viewer displays in various ways:

- PC Samples.
- Data reads and writes.
- Exception and interrupt events.
- CPU counters.
- Timestamps for these.

Trace is good for:

- Trace adds significant power to debugging efforts. Tells where the program has been.
- A recorded history of the program execution *in the order it happened.*
- Trace can often find nasty problems very quickly.
- Weeks or months can be replaced by minutes.
- Especially where the bug occurs a long time before the consequences are seen.
- Or where the state of the system disappears with a change in scope(s).
- Plus - don’t have to stop the program. Crucial to some.

These are the types of problems that can be found with a quality trace:

- Pointer problems.
- Illegal instructions and data aborts (such as misaligned writes).
- Code overwrites – writes to Flash, unexpected writes to peripheral registers (SFRs), corrupted stack.
  How did I get here ?
- Out of bounds data. Uninitialized variables and arrays.
- Stack overflows. What causes the stack to grow bigger than it should ?
- Runaway programs: your program has gone off into the weeds and you need to know what instruction caused this.
  Is very tough to find these problems without a trace. **ETM trace with the ULINK*pro* is best for this.**
- Communication protocol and timing issues. System timing problems.

For complete information on CoreSight for the Cortex-M4: Search for **DDI0314F_coresight_component_trm.pdf** on www.arm.com.

18) Keil Products:

Keil Microcontroller Development Kit (MDK-ARM™)

- MDK with included RTX RTOS – $4,895 (MDK has a great simulator)
- MDK-ARM-B: 256K code limit, no RTOS – $2,895

Keil Real Time Library (RL-ARM™)

- RTX sources, Flash File, TCP/IP, CAN, USB driver libraries - $4,195

USB-JTAG adapter (for Flash programming too)

- ULINK2 - $395 *(ULINK2 and ME - SWV only – no ETM)*
- ULINK-ME – sold only with a board by Keil or OEM.
- ULINK-*pro* - $1,395 – Cortex-M*x* SWV & ETM trace

Note: USA prices. Contact sales.intl@keil.com for pricing in other countries.

For the entire Keil catalog see www.keil.com or contact Keil or your local distributor.

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

For more information:

Keil Sales In USA: sales.us@keil.com or 800-348-8051. Outside the US: sales.intl@keil.com

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

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

For the latest version of this document, contact the author, Keil Technical support or www.keil.com.

CMSIS COMPLIANT
ARM® Cortex™ Microcontroller Software Interface Standard

Cortex™
Intelligent Processors by ARM®

KEIL™
Tools by ARM