Microsemi® SmartFusion®: Cortex™-M3 Lab: ARM® Keil™ MDK toolkit *featuring Serial Wire Viewer*

Spring 2013 Version 3.0 *by* Robert Boys bob.boys@arm.com

Hands-on lab using Microsemi SmartFusion and Keil MDK

The latest version of this document is here: www.keil.com/appnotes/docs/apnt_208.asp

Introduction:

This note describes the process of operating the ARM® Keil™ MDK toolkit featuring µVision® and Microsemi's SmartFusion family which contains an embedded ARM® Cortex™-M3 processor. SmartFusion implements ARM Serial Wire Viewer (SWV) debug technology that is available on Cortex-M*x* processors and supported by Keil. This note describes how to get all the components of this important technology working with µVision. This article needs MDK® 4.70 or later.
This lab is for the A2F processors. For SmartFusion2, that lab is here: www.keil.com/appnotes/docs/apnt_238.asp

Keil MDK:

MDK-ARM™ toolkit components include the µVision4 IDE, ARM C/C++ compiler, debugger and Keil RTX™ RTOS.

The Keil ULINK® JTAG/SWD adapter family includes the ULINK2, ULINK-ME and ULINK*pro*.

RTX now comes with a BSD type license. Source code is provided with all versions of MDK. See www.arm.com/cmsis.

A quality TCP stack is provided in MDK Professional. See the last page for details.

Why Use Keil MDK ?

MDK provides these features particularly suited for Microsemi MSS products: See www.keil.com/microsemi

1. µVision IDE with Integrated Keil Debugger, eNVM (Flash) programmer and the ARM compiler.
2. Support for Microsemi ARM7, Cortex-M1 and Cortex-M3. Nothing more to buy. 8051 support with PK51 is a separate product and it also uses µVision as its IDE.
3. A full feature RTOS is included with MDK: RTX by Keil. No royalty payments are required.
4. RTX Kernel Awareness windows. They are updated in real-time and use no system resources.
5. Serial Wire Viewer capability is included.
6. Choice of debug adapters: ULINK2, ULINK-ME, ULINK*pro* or Segger J-Link (black version).

Keil ULINK*pro* connected to A2F-EVAL-KIT-2

7. Keil Technical Support is included for one year. This helps you get your project completed faster. It is easy to renew.
8. MDK is compatible with Microsemi development products. Microsemi Libero® can create µVision projects.

Serial Wire Viewer:

Serial Wire Viewer (SWV) displays PC Samples, Exceptions (including interrupts), data writes, ITM, CPU counters and a timestamp. This information comes from the ARM CoreSight™ debug module integrated into the SmartFusion Cortex-M3. SWV does not steal any CPU cycles, is non-intrusive (except for ITM printf) and requires no stubs in your source code.

This document details these features:

1. Serial Wire Viewer (SWV).
2. Real-time Read and Write to memory locations for Watch, Memory and RTX Tasks windows.
3. Hardware Breakpoints and Watchpoints (Access Breaks).
4. RTX Viewer: a kernel awareness program for the Keil RTOS – RTX.
5. Keil TCP/IP stack: An http server example is described on page 29.
6. A DSP example using the ARM DSP Libraries. See www.arm.com/cmsis and forums.arm.com.

Part 1: Introduction and Getting Ready: 3
1) Hardware: Microsemi evaluation boards: 3
2) Software: Keil MDK toolset: 3
3) Connecting the Hardware: Setting the jumpers: 4
4) CoreSight Definitions: 5
5) SmartFusion and MDK Alerts: 5

Part 2: Blinky Example and Useful Information: 6
1) The Blinky example program using Microsemi SmartFusion MSS: 6
2) Setting Hardware Breakpoints: 6
3) Call Stack and Locals window: 7
4) Watch and Memory Windows: 8

Part 3: Configuring the Serial Wire Trace: 9
1) Selecting SWD or JTAG ports: 9
2) Configuring the Serial Wire Viewer (SWV): 10
3) Trace Configuration Fields: *for reference:* 11

Part 4: Serial Wire Viewer Trace Exercises: 13
1) Logic Analyzer: Display variables graphically: 13
2) Data Read/Write events display in the Trace Records window: 14
3) Watchpoint example: 15
4) Exception Tracing: 16
5) PC Samples: Tracing: 17
6) Debug (printf) Viewer: 18

Part 5: Keil RTX Real Time Operating System: 19
1) RTX: Keil's RTOS: RTX_Blinky example: 19
2) RTX Viewers: Kernel Awareness: 20
   a) Tasks and System window: 20
   b) Event Viewer window: 20
3) Configuring SWV for the Event Viewer: 21
4) Logic Analyzer Window: View RTX task variables real-time in a graphical format: 22

Part 6: DSP Example using ARM CMSIS-DSP Libraries: 23
1) DSP example: 23
2) Displaying DSP variables in the Logic Analyzer (LA) 24
3) RTX Viewer for DSP example: 24
4) RTX Tasks and System Viewer: 25
5) RTX Event viewer: 26

Part 7: …more Useful Information 26
1) Creating a new µVision project from scratch: 26
2) Running Blinky from RAM: 27
3) TCP/IP: Keil http demonstration: 29
4) ARM JTAG/SWD adapter connector schematics and the New ULINK-ME: 30
7) Serial Wire Viewer (SWV) and Serial Wire Output (SWO) information: 31
9) Programming the A2F-EVAL-KIT and DEV-KIT Boards (FPGA Flash with STAPL) 33
10) Serial Wire Summary and what is it good for: Useful Documentation: 35
11) Keil Contact and Product Information: 36

1) Hardware: Microsemi evaluation boards:

We will examine two SmartFusion boards and will focus on the smaller one: the A2F-EVAL-KIT.

A2F-DEV-KIT: The Big Board. See below. It is connected to a ULINK-ME in the upper right corner. This is referred by Microsemi as the SmartFusion Development Kit.

This kit is populated with a SmartFusion A2F500M3G device. This board was once available with an A2F200 device but this option is no longer available. The A2F500 Serial Wire Viewer (SWV) works at full speed.

A2F-EVAL-KIT: The Small Board. See below. It is shown with a Keil ULINK2 USB to JTAG/SWD adapter. This is referred by Microsemi as the SmartFusion **Eval**uation **Kit**. This kit is populated with an AF200M3G. The A2F200 device has the Serial Wire Viewer (SWV) speed limited to 98 KHz.

A2F-EVAL-KIT2: Rev 2 is similar to its predecessor except for the addition of jumpers for the leds. The default STAPL file (stp) works with the Keil Blinky examples. You can also use the Keil provided stp file.

Debug Adapters: A Keil ULINK2, ULINK-ME, ULINK*pro* or a Segger J-Link (black case, version 6 or later) can be used. ULINK2 and ULINK-ME are functionally the same. The ULINK*pro* is used the same way as aULINK2 or ULINK-ME, but faster flash programming and with the addition of ARM ETM trace support. SmartFusion A2F does not support ETM.

2) Software: Keil MDK toolset:

The current version of MDK is 4.70 in March 2013. You can use previous versions as far back as 4.20, but this document used 4.70. The evaluation version (MDK-Lite) is available free from www.keil.com. A license is not needed.

Please install MDK into the default directory of C:\Keil to easier follow the directories in this note. Normally, you can install MDK in any directory you want.

STAPL file: You do not need a FPGA design programmed into SmartFusion parts to program and run the Cortex-M3 processor with µVision. You do need the FPGA programmed in order to have the leds blink as used by the Keil examples. The A2F-EVAL-KIT2 Rev 2 default STAPL file works properly with the Keil Blinky examples. Or you can use mss_101.stp.

The STAPL file mss_101.stp is available here: www.keil.com/appnotes/docs/apnt_208.asp

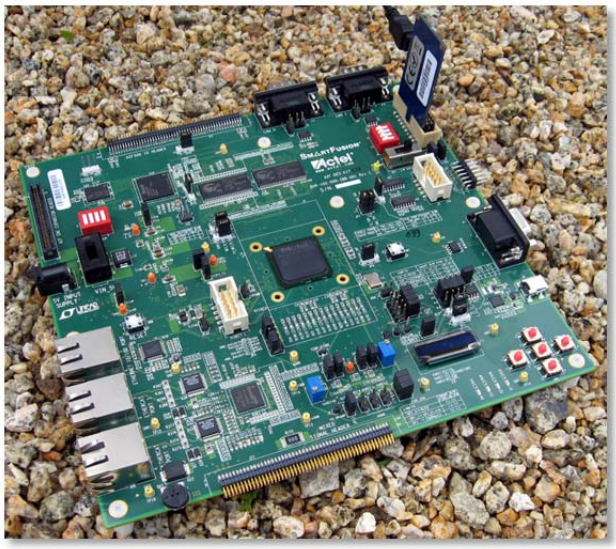
If the leds do not light and the Blinky program is running correctly, program mss_101.stp into the SmartFusion FPGA.

Instructions to program mss_101.stp into the FPGA Flash (eNVM) using the Microsemi program FlashPro are on page 33.

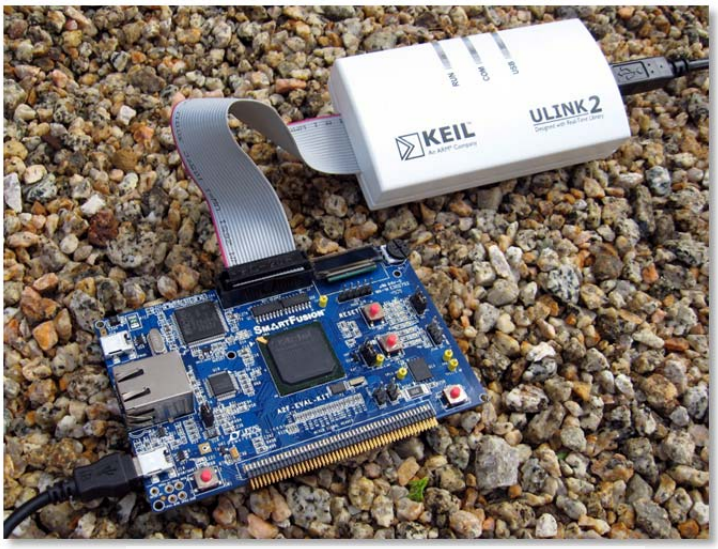
FlashPro is available stand-alone or as part of the Libero IDE FPGA development software.

TIP: You must change two jumpers to program the FPGA Flash. Remember to switch these jumpers back in order to connect successfully with a ULINK. These jumpers are JP7 and JP10 in the EVAL board. This is a common error.

On the DEV-KIT, switch SW9 must be set to "S/W Debug ON", JP7 set to RVI and JP6 to V1P5_EXT in order for any ULINK to work with the examples in this lab. The other jumpers on both boards must be in their default positions.



DEV-KIT with Keil ULINK-ME



A2F-EVAL-KIT with Keil ULINK2

3) Connecting the Hardware: setting the jumpers:

A2F-EVAL-KIT: the Small Blue Board: See Figure 1.

1. Connect the debugging adapter to J3 RVI Header, the standard JTAG adapter as shown on the previous page.
2. Set JP7 and JP10 jumpers set to position 2-3. They are labeled RVI and M3 respectively.
3. JP6 2-3, J6 installed, J5 and J8 are not installed.
4. Connect a USB cable to J14 to power the board.

A2F-EVAL-KIT Rev 2: See Page 1 and Figure 2.

1. Connect the debugging adapter to J3 RVI Header, the standard JTAG adapter as shown on the previous page.
2. Set JP7 and JP10 jumpers to position 2-3. They are labeled RVI and JTAG SEL respectively. There are six jumpers in a row. The two outside ones are JP7 and JP10. These must be moved towards the board center of the board to use a debug adapter such as a ULINK2.
3. JP6 2-3, J6, Vusb, J25 installed. J5 and J8 are not installed.
4. Connect a USB cable to power the board to J14.

A2F-DEV-KIT: the Big Green Board: See Figure 3.

1. Connect the ULINK to J3, the standard JTAG adapter as shown:
2. SW9 to S/W Debug ON. (the right most position)
3. JP7 set to 2-3 RVI. (the other position is to program the FPGA Flash)
4. JP6 must be in position 2-3 V1P5_EXT.
5. JP5 both jumpers to C1 – Single Prog. See Figure 4 for a close-up of these jumpers.
6. Power the board with the Microsemi provided 5 volt AC adapter

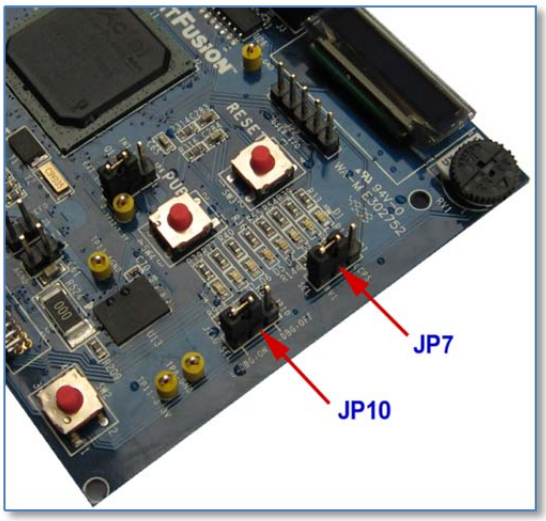


Figure 1: A2F-EVAL-KIT Jumper Settings for Keil ULINK operation.

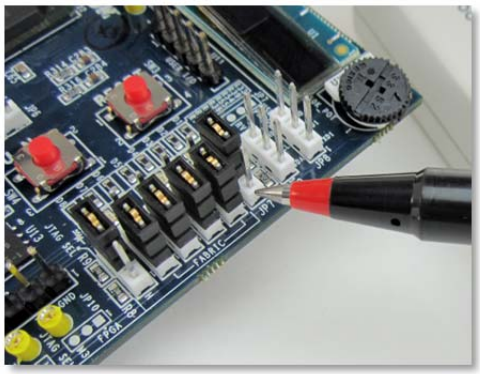


Figure 2: A2F-EVAL-KIT Rev 2 Jumper Settings for Keil ULINK

TIP: If you see a Found New Hardware Wizard message when you connect to the USB port, click Cancel as we will not be using it.

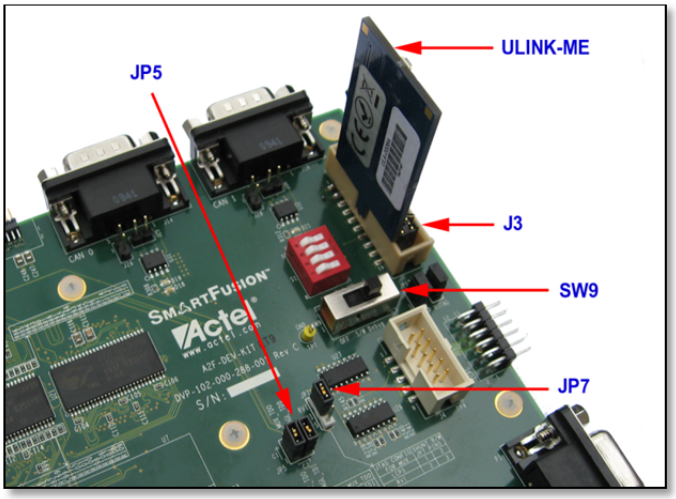


Figure 3: A2F DEV-KIT Jumper Settings for Keil ULINK

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

- **JTAG:** Provides access to the CoreSight debugging module located on the Cortex processor. It uses 4 to 5 pins.
- **SWD:** Serial Wire Debug is a two pin alternative to JTAG and has about the same capabilities except for no Boundary Scan. SWD is referenced as SW in the µVision Cortex-M Target Driver Setup. See page 6, second screen in the Port: box. The SWJ box must be selected to be able to enable SWD.
- **SWV:** Serial Wire Viewer: A trace capability providing display of reads, writes, exceptions, PC Samples and printf. SWV must use SWD because of the TDIO and SWO conflict when JTAG is used.
- **DAP:** Debug Access Port. A component of the ARM CoreSight debugging module that is accessed via the JTAG or SWD port. One of the features of the DAP are the memory read and write accesses which provide on-the-fly memory accesses without the need for processor core intervention. µVision uses the DAP to update Memory, Watch and a RTOS kernel awareness window 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 code stubs are needed in your sources. You do not need to configure or activate DAP. µVision does this automatically when you select the function.
- **SWO:** Serial Wire Output: SWV frames usually come out this one pin output. It shares the JTAG signal TDIO.
- **ITM:** Instruction Trace Macrocell: 32 32 bit registers that output data to the outside world using SWV. µVision uses ITM 0 for a printf feature and ITM 31 for the RTX Event Viewer window. The others are not used. ITM usage is the only part of SWV that is slightly intrusive. The ASCII writes to ports require some CPU cycles.
- **Trace Port:** A 4 bit port (4 data and 1 clock) that ULINK*pro* uses to collect ETM frames and optionally SWV (rather than with the SWO pin). These signals are available only on the SmartFusion2.
- **ETM:** Embedded Trace Macrocell: Provides all the program counter values. Only ULINK*pro* provides ETM. SmartFusion does not have ETM. SmartFusion2 provides ETM.

5) SmartFusion and MDK Alerts:

Here are a few minor items that might cause issues when using Keil MDK and SmartFusion boards. They are not critical and easy to implement work-arounds exist as described below: Normal SmartFusion operation is not affected by these issues.

nTRST pulled low: This prevents JTAG operation in most cases. SWD can be used successfully.

SmartFusion eval boards sometimes have a 510 or 1K Ω resistor from the nTRST pin (# 3) on the JTAG connector to ground. This keeps the JTAG TAP controller in RESET. This is out of the JTAG spec (nTRST should be pulled high with 10K Ω or so) and it might cause a Debug adapter to not function with JTAG selected. Either remove this resistor or put a 330 ohm between nTRST (pin 3) and VTref (pin 1). Alternatively, just use SWD. This is the easiest fix. You can do everything with SWD that you can with JTAG except for boundary scan. SWD is required for SWV.

RTX: RTX needs a special RTX library to operate because two ARM instructions are not implemented.

SmartFusion processors do not have the LDREX or STREX instructions used by many RTOSs including Keil RTX. MDK has a special version of one RTX library file that does not require these instructions. The RTX exercise in this lab instructs you how to specify this library. See www.keil.com/support/docs/3551.htm for more information. SmartFusion2 has these instructions implemented and is not affected.

Serial Wire Viewer (SWV) Speed: (A2F200 only) SWV frames will be lost at high data rates.

The A2F200 device has the Serial Wire Viewer (SWV) speed limited to 98 KHz. You must take care not to overload the SWO pin by selecting too many SWV elements. The A2F500 and SmartFusion2 SWO run at full speed. This is normally in excess of 1 MHz. In all cases, the CPU is running at full speed. In the Keil examples, this is 100 MHz.

ULINK-ME supplying 3.3 volts to target: Board Power ON sequence can be disturbed: mostly on the DEV KIT)

Early versions of ULINK-ME provide 3.3 volts power to a target board through JTAG connector pin 1. You might want to remove this by cutting the pin 1 lead or remove the diode on the ME that is connected to pin 1. SmartFusion boards have complex power supplies that can be confused when 3.3 volts is applied before 5 volts. If your ULINK-ME has a 10 pin Cortex connector on it or says Vers 5: the diode D1 has been removed. ULINK2 is not affected.

1) The *Blinky* example program using Microsemi SmartFusion MSS:

Now we will run the Keil MDK development system using a SmartFusion EVAL-KIT evaluation board. You can adapt these instructions for the DEV-KIT or your own target board.

You do not need anything programmed in the FPGA Flash to run Cortex-M3 programs. You do if you want the leds to blink.

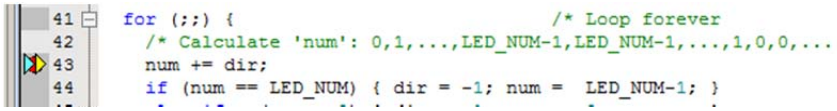
1. Set the jumpers for debug operation.See page 4. Connect a ULINK2 to the board. If using a ULINK*pro* or J-Link, you will have to select these devices in the Target Options menu. Set the eNVM programming in the Utilities tab.
2. Start µVision by clicking on its desktop icon. 
3. Select Project/Open Project from the main menu.
4. Open the file C:\Keil\ARM\Boards\Actel\SmartFusion\Blinky\Blinky.uvproj.
5. On DEV-KIT change the "6" near line 16 in Blinky.c to a 0, 1, 2 or 3: **const unsigned long led_mask[] = { 1 << 6 };**
6. Click on the Target Options icon  or select "Project/Options for Target".
7. Select the Debug tab.
8. Click Settings beside the USE: ULINK2/ME Cortex Debugger box.
9. Confirm SWJ and SW are selected. Click OK twice to return to the main menu. JTAG will probably not work.
10. Compile the source files by clicking on the Rebuild icon.  .
11. Program the SmartFusion eNVM flash by clicking on the Load icon:  Progress is indicated in bottom left corner.
12. Enter Debug mode by clicking on the Debug icon.  Select OK if the Evaluation Mode notice appears.
13. Click on the RUN icon to start the Blinky program.  . ***Blinky is now running !***
14. If the FPGA is programmed to connect the GPIO port to the leds, at least one led will blink.
15. Stop the program with the STOP icon. 

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

LEDs: No leds will blink on the EVAL-KIT or the DEV-KIT when the Blinky program is running unless the FPGA is programmed with a hardware design to connect the GPIO port to the led. This is the one exception where you need some design programmed into the FPGA. A suitable design is provided with this document on the web.

2) Setting a Hardware Breakpoint (on-the-fly):

1. Run the program again.  .
2. Open the Blinky.c tab. It is probably already open.
3. Scroll down to near Line 43: **`num += dir;`**
4. Note on the left of the line numbers are darker grey blocks. This indicates there is assembly code present and you can set a hardware breakpoint on this line. You can also see these in the Disassembly window.
5. *While the program is still running*, click to the left of the 43 and a red circle will be created. This is a hardware breakpoint. SmartFusion has 6. µVision will warn you if you set too many. Ctrl-B lists all breakpoints set.
6. The cyan arrow is where the source and disassembly windows are matching. You set this with your mouse.
7. Note that the program will soon stop at line 43. The yellow arrow is the current program counter position. This will be the next instruction executed when the CPU is started again.



TIP: You can set and unset breakpoints on-the-fly. No need to stop the CPU. This is an important feature.

TIP: ARM CoreSight Hardware breakpoints are no-skid. They do not execute the instruction they land on.

8. Click on the red circle to remove the breakpoint. You can also use Ctrl-B to access the Breakpoint table.

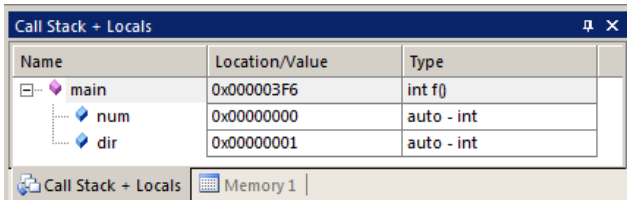
3) Call Stack + Locals Window:

Local Variables:

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

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

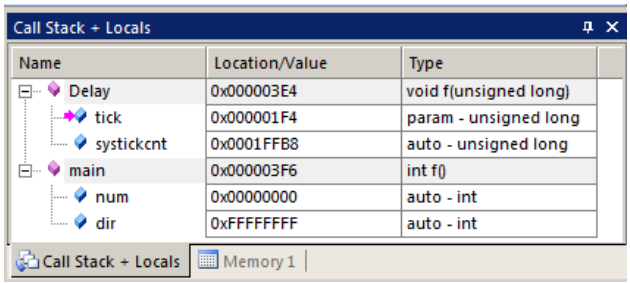
1. We will use the breakpoint you set on the previous page near line 43 in Blinky.c: **`num += dir;`**
2. Run Blinky. The program will soon stop on this breakpoint.
3. Click on the Call Stack + Locals tab if necessary to open it.
4. Shown is the Call Stack + Locals window.
5. The contents of the local variables are displayed as well as function names.
6. In this example, two local variables **`num`** and **`dir`** are displayed in the window here with their values:



7. Click on the Step icon or F11: Make sure the Blinky.c window is in focus else you will step in assembly.
8. The program enters the **`Delay`** function. The Call Stack + Locals window will now display this event as shown here:
9. Click on the Step icon or F11 multiple times until you re-enter the Delay function. Other functions would be similarly displayed but this is a very simple example program.

TIP: If the Disassembly window is in focus the steps will be by assembly instruction. If a source window is in focus: the steps will be by C or C++ source line.

10. Click on the Step Out icon or CTRL-F11 to exit all function(s) to return to main(). This will be indicated in the Call Stack window.
11. **When you are ready to continue, remove the hardware breakpoint by clicking on its red circle ! You can also type Ctrl-B and select Kill All.**



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

TIP: You can set and unset breakpoints on-the-fly. No need to stop the CPU. A CoreSight hardware breakpoint does not execute the instruction it breaks on. These are rather important features.

TIP: This is standard "Stop and Go" debugging. ARM CoreSight debugging technology can do much better than this. You can display global or static variables updated in real-time while the program is running. No additions or changes to your code are required. Variable update while the program is running is not possible with local variables because they are usually stored in a CPU register. They must be converted to global or static variables so they always remain in scope: normally in RAM.

Call Stack:

The list of stacked functions is displayed when the program is stopped as you have seen. This is useful when you need to know which functions have been called and are stored on the stack. This can be quite useful for finding program crashes.

!! Do not forget to remove the hardware breakpoint(s) before continuing.

TIP: You can access the Hardware Breakpoint table by clicking on Debug/Breakpoints or Ctrl-B. This is also where Watchpoints (also called Access Points) are configured. You can temporarily disable entries in this table by unchecking them. You can permanently delete them by using one of the Kill buttons.

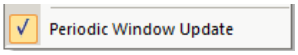
4) Watch Window: Updated in Real-time ! These two features do not use SWV.

We want to display a variable. The Watch and Memory windows can be displayed while the program is running. These must be global, static, a physical address, a structure or array or anything that visible in all functions. Locals will not be displayed until the program is stopped in the function in which they exist.

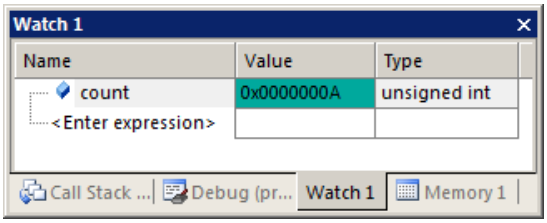
Create a Global Variable:

1. Exit Debug mode  . You can edit source code while in Debug mode but you must be in Edit mode to compile.
2. In Blinky.c, declare a global variable near line 15: **`unsigned int count = 0;`**
3. Just after the C source line **`Delay(500);`** near line 48 add these lines:

```
count++;
if (count > 0x0F) count = 0;
```

4. Compile the source files by clicking on the Rebuild icon.  .
5. Program the SmartFusion2 eNVM flash by clicking on the Load icon:  Progress is indicated in bottom left corner.
6. Enter Debug mode by clicking on the Debug icon.  Select OK if the Evaluation Mode notice appears.
7. Click on the RUN icon to start the Blinky program.  .
8. Select View and confirm Periodic Windows Update is selected as shown: 

9. Right-click on the variable name **`count`** and select Add 'count' to… and then select Watch 1.
10. The value of **`count`** will be displayed and updated in real-time without stealing any CPU cycles as shown here:
11. The value of **`count`** displayed depends on when it is sampled.

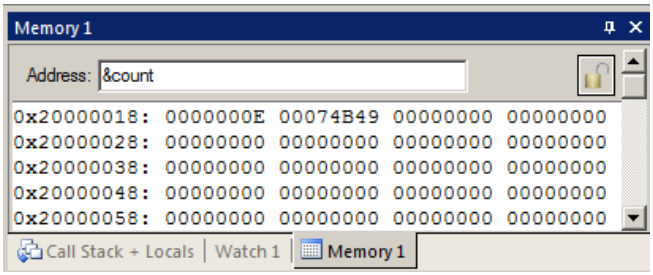


TIP: To Drag 'n Drop into a tab that is not active such as the Logic Analyzer, Watch or Memory windows, pick up the variable by blocking it, click and hold, and move it over the tab you want to open; when the tab opens, move your mouse into the window and release the variable.

5) Memory Window: it is also updated in Real-time:

1. Click on the Memory 1 tab to open it.
2. Enter **`count`** in the space Address: provided. The program can be running.
3. Right click anywhere inside the memory data display area and select Unsigned Long.
4. Note that as **`count`** increments, the address it points to changes. This is useful for work with pointers.
5. Add an ampersand (&) in front of **`count`**. i.e. **`&count`** The window below opens:
6. Note the memory address changes to 0x2000 0018 in this case.
7. This is the physical address in RAM where the variable **`count`** is stored.
8. Right-click on the data field (0x0E in this case) and select Modify Memory at 0x2…18 and enter 0x0 and press Enter.
9. This value is inserted into **`count`** in real time using CoreSight DAP technology.

How It Works: Read and Write to Memory Locations:
µVision has the ability to read and write memory locations on-the-fly and without stealing CPU cycles by using the Debug Access Port (DAP). The Cortex-M3 is a Harvard architecture: this means it has separate code and data address busses. While the CPU is fetching instructions at full speed from the code space, µVision can access the data space via JTAG or SWD. This feature is active even while the CPU is halted. SWV is not used for this feature and can be disabled or not.

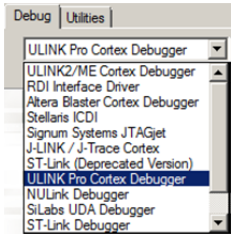


This feature is used to display data in the Watch and Memory windows, Event Counters as well as an RTX Awareness window. Not only can values of memory locations be displayed and updated in real-time: you can also insert values while the program is running in the Memory window as demonstrated above. The only instance real-time will be violated is in the unlikely event the CPU accesses a memory location the exact same time as µVision does.

1) Selecting SWD or JTAG ports:

In the next few exercises, we will demonstrate several Serial Wire Viewer features. You must use Single Wire Debug (SWD) mode to use Serial Wire Viewer. JTAG will not work because of the TDIO and SWO pin conflict. SWD is called SW in µVision. It is configured in µVision. You must first have a ULINK connected to a Cortex-M3 target.

1) If necessary, stop the CPU and exit debug mode.
2) Click on the Target Options icon or select "Project/Options for Target.
3) Click on the Debug tab.
4) Select your debug adapter here:
5) Click Settings beside the USE: ULINK Cortex Debugger box to open the screen below:



The box labeled SW Device (or labeled JTAG Device Chain if JTAG is selected) will display the CoreSight devices detected in the processor or an error or be blank if none are found.

6) Select SWJ and SW in the Port: box. Confirm ARM CoreSight SW-DP is displayed in the SW Device box. You must see ARM CoreSight SW-DP before you continue to Step 6.
7) To refresh this window switch between JTAG and SW in the Port: drop down menu.

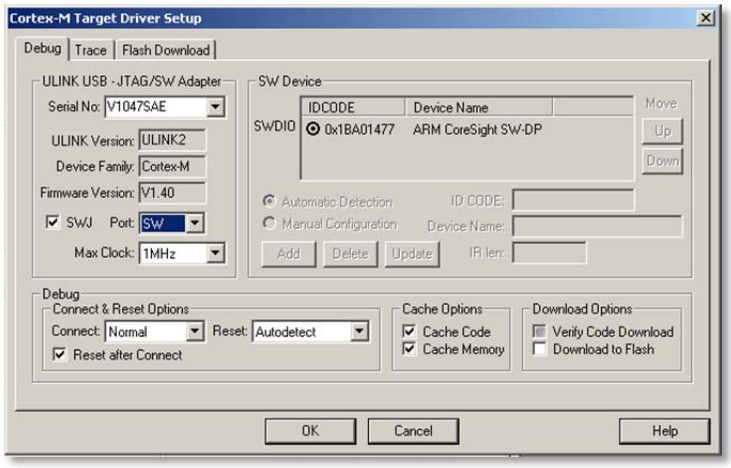
A ULINK is now successfully been connected to the CoreSight SWD port in the screen shown below.

TIP: With SmartFusion or SmartFusion2 boards, JTAG will not operate unless any 1K pulldown resistor is removed. In these cases, just use SWD (SW).

Urgent TIP: If you don't see either SW-DP or JTAG-DP in this window and get an error message or device not found – stop right now and fix this. This section ***must*** work before you can continue any debugging. **Make sure the jumpers are set correctly** and the board is properly powered. See **TRAP** below. Try re-powering the board.

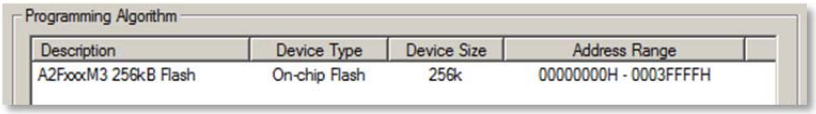
TRAP: Top reason people can't get SW and/or JTAG working is a wrong setting of the jumpers JP7 and JP9 (on EVAL board) and JP5, JP6, JP7 and SW9 on the larger DEV board. This is because these must be switched to enable FPGA programming. By default they are set to FPGA programming and not set for standard SWD or JTAG operation.

TIP: The box "Verify Code Download" ***must*** be unchecked for SmartFusion. Code verify is done inside the programmer. This means µVision does ***not*** check the program when you enter Debug as is usual with other processors.



eNVM Flash Programming:

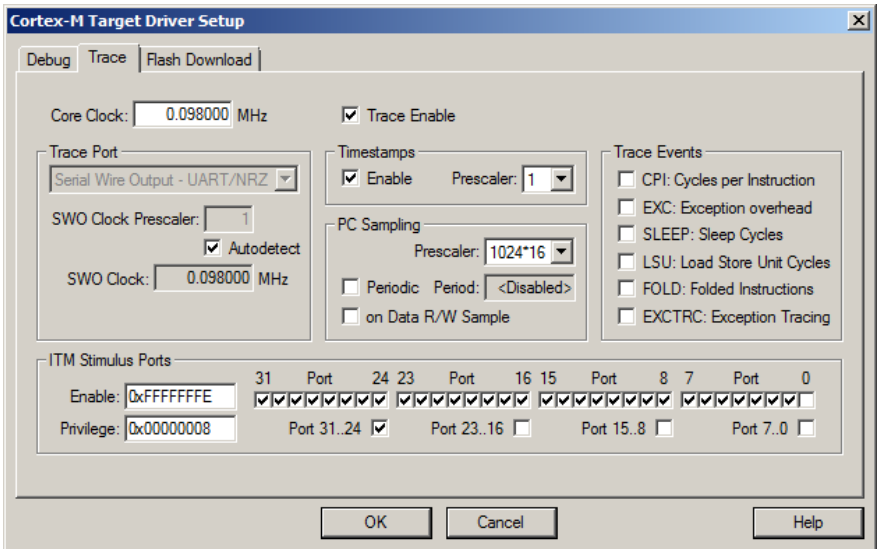
8) Click on OK once return to the Target Options window.
9) Select the Utilities tab. This is where you select the debug adapter to perform eNVM Flash programing.
10) Click on Settings to select the appropriate Flash algorithm for your processor as shown below:
11) Click on OK twice to return to the main µVision menu.



2) Configuring the Serial Wire Viewer (SWV):

In order to use the Serial Wire Viewer features, it must be configured.

1) If necessary, stop the CPU and exit Debug mode.
2) Click on the Target Options icon or select "Project/Options for Target".
3) Click on the Debug tab.
4) Click Settings beside the USE: ULINK Cortex Debugger box. Confirm SWJ and SW are selected.
5) Click on the Trace tab. The window below opens up.
6) For A2F200 parts set Core Clock to 0.098 MHz. For the A2F500, set Core Clock: to 50 MHz (J-Link 75 MHz).
7) Select Trace Enable. Unselect the EXCTRC, Periodic and on Data R/W Sample boxes.
8) Set the Timestamps Prescaler to 1 to lessen data overruns.
9) Click on OK twice to return to the main screen.
10) The Serial Wire Viewer is now activated.
11) Click on File/Save All to save these settings.



Trace Configuration window.

TIP: How to open this window again to modify trace selections:

1) **In Edit mode:** The Target Options icon as already described.
2) **In Debug Mode:** Select Debug/Debug Settings from the main menu.

You select various SWV elements. It is important to note that it is easy to overload the single wire Serial Wire Output (SWO) pin. This is more pronounced with the A2F200 than the A2F500 device. The general rule is to activate only those features you need. A ULINK*pro* can better handle SWV because it uses the faster Manchester rather than UART/NRZ as with ULINK2.

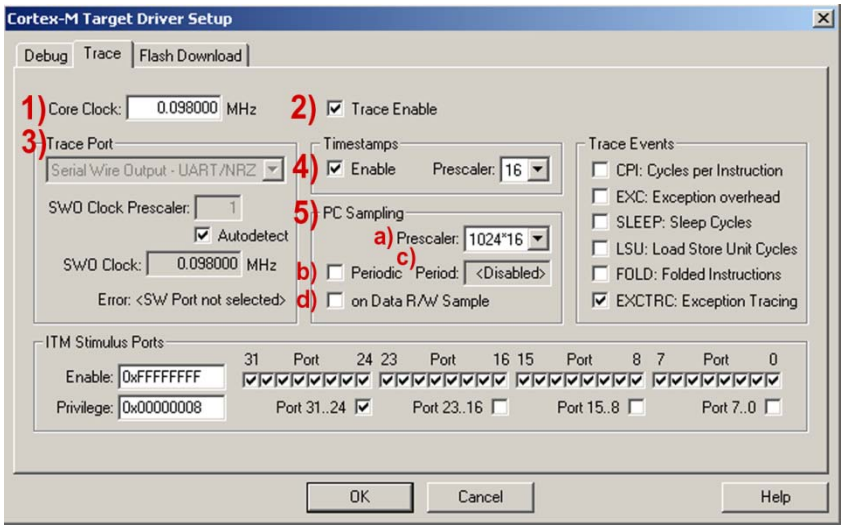
TIP: You can only enable/disable trace from Edit mode. You can change the rest while in either mode.

The next page describes in detail the various fields of the Trace Configuration window.

Testing SWV:

Enter Deug mode and RUN the program and open the Trace Records window. It should like similar to the one on page 16. If it is not working correctly, usually you get either nothing or spurious frames. If you see ITM frames with numbers other than 0 or 31, or other frames that do not make sense, the Core Clock speed is probably wrong. This test assumes you have SYSTICK running and EXCTRC enabled. If not, you can select PC Samples to get some familiar SWV frames. ULINK*pro* automatically detects the Core Clock: value. It uses the Core Clock: value you should enter to determine timing values that will e displayed.

3) Trace Configuration Fields: *for reference…*



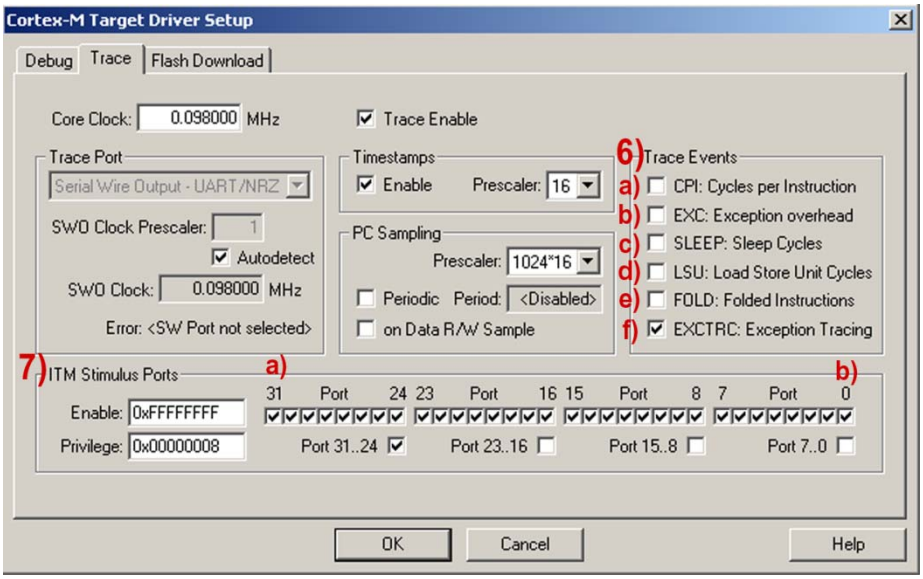
Trace Configuration window Fields 1) through 5).

1) **Core Clock:** The CPU clock speed for SWV. SWO Clock signal is derived from and is a ratio of the Core Clock. Current ratio for SmartFusion A2F200 is CPU/1024. 100 MHz/1024 = 98 KHz. For the A2F500 the ratio is 100/2 = 50 MHz. (J-Link = 75 MHz). If you change the speed the CPU is running at, these values will obviously change.
2) **Trace Enable:** Enables SWV and ITM which is essentially everything on this window except Trace Port. This does not affect the DAP Watch and Memory window display updates.
3) **Trace Port:** Selects the SWO trace output UART or Manchester protocol.
    a. **Serial Wire Output – UART/NRZ:** ULINK2 and ULINLK-ME must use this. UART/NRZ encoding is not supported by ULINK*pro*. An error will result when you enter debug mode with ULINK*pro*.
    b. **Serial Wire Output – Manchester**: Use Manchester encoding. ULINK*pro* only.
4) **Timestamps:** Enables timestamps and selects a Prescaler. 1 is the default. Selecting a higher value can, but not always lessen SWO overloads. Completely disabling the timestamps can lessen data overruns but can disable other SWV features. It is worth a try if you are having overload problems.
5) **PC Sampling:** Samples the program counter and displays in the Trace Records window.
    a. **Prescaler** 1024*16 (the default) means every 16,384th PC is displayed. The rest are lost.
    b. **Periodic:** Enables PC Sampling.
    c. **Period:** Automatically derived from Prescaler and Core Clock settings.
    d. **On Data R/W Sample:** Displays the address of the instruction that made a data read or write of a variable entered in the Logic Analyzer in the Trace Records window. This is not connected with PC Sampling.

TIP: It is important to ensure the Serial Wire Output (SWO) pin is not overloaded. µVision will alert you when an overflow occurs with a "X" in the Trace Records window or with a "D" or a "O" in the ULINK*pro* Instruction Trace window. µVision easily recovers from these overflows and immediately continues displaying the next available trace frame. Dropped frames are somewhat the normal situation especially with many data reads and/or writes.

TIP: ULINK*pro* can process SWV information much faster than the ULINK2 or ULINK-ME can. This results in fewer dropped frames especially with high data transfer rates out the SWO pin. Data overruns are often associated with a fast stream of data reads and writes which are set in the Logic Analyzer. Minimize these issues by displaying only the information you really need.

The A2F500 can output SWV frames faster than the A2F200 therefore much less susceptible to overloading and dropping frames at high data rates. The A2F200 SWO speed is fixed at 98 KHz and the A2F500 is usually near 1.25 MHz. This speed is determined by the Debug adapter and µVision in the Trace Configuration window. You can set it manually.



Trace Configuration window Fields 6) through 8)

6) **Trace Events:** Enables various CPU counters. All except EXCTRC are 8 bit counters. Each counter is cumulative and an event is created when this counter overflows every 256 cycles. These values are displayed in the Counter window. The event created when a counter wraps around is displayed in the Trace Records window or the Instruction Trace window (ULINK*pro*).
Event Counters are updated using the DAP and not SWV. These events are memory mapped and can be read by your program or the µVision Memory window.
   a. **CPI: Cycles per Instruction:** The cumulative number of extra cycles used by each instruction beyond the first one plus any instruction fetch stalls.
   b. **Fold:** Cumulative number of folded instructions**.** This will result from a predicted branch instruction removed and flushed from the pipeline giving a zero cycle execution time.
   c. **Sleep:** Cumulative number of cycles the CPU is in sleep mode. Uses FCLK for timing.
   d. **EXC:** Cumulative cycles CPU spent in exception overhead not including total time spent processing the exception code. Includes stack operations and returns.
   e. **LSU:** Cumulative number of cycles spent in load/store operations beyond the first cycle.
   f. **EXCTRC:** Exception Trace. This is different than the other items in this section. This enables the display of exceptions in the Instruction Trace and Exception windows. It is not a counter. This is a very useful feature and is often used in debugging.

7) **ITM Stimulus Ports**: Enables the thirty two 32 bit registers used to output data in a *printf* type statement to µVision. Port 0 is used for the Debug (printf) Viewer, Port 31 is used for the Keil RTX real-time kernel awareness window. Only Ports 0 and 31 are currently implemented in µVision and should normally be checked. Ports 1 through 30 are not currently implemented and are Don't Care.
   a. **Port 31:** Enables the ITM port used for the RTX Viewer.
   b. **Port 0:** Enables the ITM port used for the Debug (printf) Viewer. A small amount of instrumentation code is needed in your project. See 10) Debug (printf) Viewer in Part 4 for information on using this feature.

1) Logic Analyzer Window (LA): *This feature uses SWV which must be configured properly:*

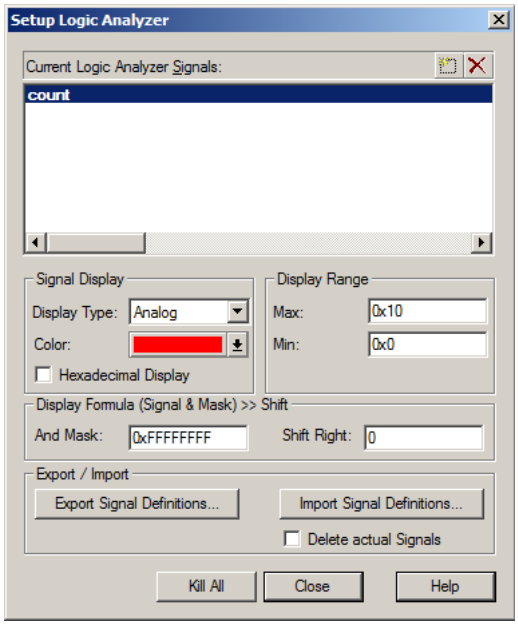
The Logic Analyzer (LA) displays up to four variables in a graphical format. When a variable is placed in the LA, it is also displayed in the Trace Records window.

1. You should have configured SWV on page 10: **Configuring the Serial Wire Viewer:**
2. µVision must be in Debug mode. The program can be running.
3. Find the global variable `count` you created in Blinky.c.
4. Right click on `count` and select Add 'count' to… and select Logic Analyzer.
5. This action will open the Logic Analyzer window if not already visible.
6. Click on 'Setup…' and the Setup Logic Analyzer window below will open up:
7. Select the Display Range max = 0x10. Click on Close.
8. Click on RUN if the program is not already running. LA window will be updating as shown below:
9. If necessary, click on Zoom: Out or All for an appropriate X axis. Try 2 or 5 seconds or for the A2F200, much higher.
10. Add **`count`** to the Watch and Memory windows if it is not already there. You can right click on `count` and select Add 'count' and select Watch 1 and then repeat and select Memory 1.
11. In the Memory window: change the value of count to various values to change LA values. Right click on the correct Data portion in Memory 1 and select Modify Memory at 0x20000018 and hit Enter.
12. You should see your new values in the LA.

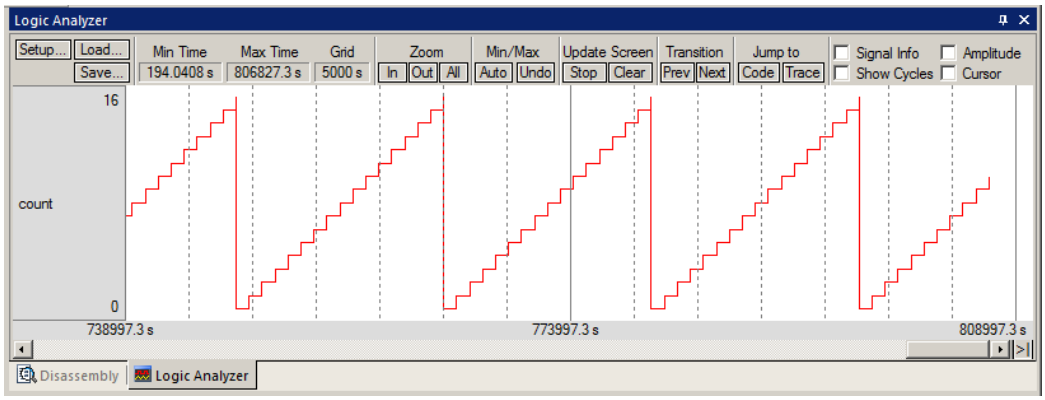
TIP: If the LA does not display any waveforms: make sure there are no extra elements in the Trace Configuration are set. Check EXCTRC as this is often enabled in demonstration programs. It must be off for this exercise.

TIP: The A2F500 works better with SWV than does the A2F200 and its slower SWO pin. A ULINK*pro* can often provide better SWV performance.

TIP: You can enter up to four variables in the LA window.



Logic Analyzer Setup window.



Logic Analyzer windows displaying global variable count.

Crucial TIP: If the Logic Analyzer data does not increment but the variable does in the Watch or Memory windows, this can mean the SmartFusion SWO is overloaded. Sometimes the LA will update only by clicking on the scroll bar or stopping the program. Disable some of the SWV functions such as Exceptions (EXCTRC). Disabling Timestamps often stops the LA.

Logic Analyzer Notes:

1. The Logic Analyzer displays up to four variables in a graphical format.
2. SWV is used to obtain the data displayed. SWV must be configured. It comes out the Serial Wire Output pin.
3. You will not be able to enter variables if the SWV Trace configuration is not enabled and configured properly. The Core Clock needs to be accurately set. ULINK*pro* uses your Core Clock: value for displaying timing values.
4. Each variable will have its read and write operations displayed in the Trace Records window.
5. You must enter a variable name. Raw memory addresses are allowed. See the TIP: below.
6. You might have to fully qualify a variable. An example is \Blinky\AD_dbg.
7. Number 1 reason you can't enter a variable is Trace not properly set. Test SWV using EXCTRC or PC Samples.
8. You can drag and drop variables into the LA. Variables must be global, static or a structure.
9. You may need to modify the range to greater than 5 seconds to clearly display most variables.
10. The LA is updated at a fixed rate of approximately 500 msec.

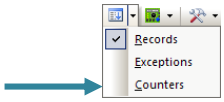
TIP: Raw addresses can also be entered into the Logic Analyzer.

An example is: ***((unsigned long *)0x20000000)**

5) Data Reads and Writes in the Trace Records window for the variable count:

When a variable is entered into the Logic Analyzer, the associated data reads and/or writes are displayed in the Trace Records window. This is the method used to enter reads and writes in the Trace Records window: put them in the LA.

This feature uses SWV which must be configured properly:

1. Select Debug/Debug Settings and the Trace tab. Confirm EXCTRC and PC Samples are not enabled.
2. Click on OK twice to return to the main menu.
3. Click on RUN.
4. Open the Trace Records window with View/Trace/Records or the pull-down menu: 

5. The window below opens up. Data Reads and Writes to variable count (at 0x2000_0018 in this case) are displayed.
6. Note the data values are incrementing. Double-click in the Trace Records window to clear it.
7. Open Debug/Debug Settings and select the Trace tab.
8. Select on Data R/W sample and click on OK twice. This will add the PC column.
9. Click on RUN. 

TIP: UKINK*pro* has a different trace window and currently the program must be stopped to update it.

Trace Records

Type	Ovf	Num	Address	Data	PC	Dly	Cycles	Time[s]
Data Read			20000018H	0000004CH	000004ECH	X	66675153245	680358.70658...
Data Write			20000018H	0000004DH	000004F4H	X	66675286511	680360.06643...
Data Read			20000018H	0000004DH	000004ECH	X	66775153257	681379.11486...
Data Write			20000018H	0000004EH	000004F4H	X	66775287166	681380.48128...
Data Read			20000018H	0000004EH	000004ECH	X	66875153245	682399.52290...
Data Write			20000018H	0000004FH	000004F4H	X	66875286811	682400.88582...
Data Read			20000018H	0000004FH	000004ECH	X	66975153257	683419.93119...
Data Write			20000018H	00000050H	000004F4H	X	66975287077	683421.29670...
Data Read			20000018H	00000050H	000004ECH	X	67075153245	684440.33923...
Data Write			20000018H	00000051H	000004F4H	X	67075286923	684441.70329...
Data Read			20000018H	00000051H	000004ECH	X	67175153257	685460.74752...
Data Write			20000018H	00000052H	000004F4H	X	67175287249	685462.11478...
Data Read			20000018H	00000052H	000004ECH	X	67275153245	686481.15556...
Data Write			20000018H	00000053H	000004F4H	X	67275287245	686482.52290...
Data Read			20000018H	00000053H	000004ECH		67375153257	687501.56384...
Data Write			20000018H	00000054H	000004F4H	X	67375286399	687502.92243...
Data Read			20000018H	00000054H	000004ECH	X	67475153245	688521.97188...
Data Write			20000018H	00000055H	000004F4H	X	67475287167	688523.33843...
Data Read			20000018H	00000055H	000004ECH	X	67575153257	689542.38017...
Data Write			20000018H	00000056H	000004F4H	X	67575286729	689543.74213...

Trace Records displaying read and write operations to global variable count.

TIP: If you right click in the Trace Records window, you can filter out various types of frames. Doing this does not relieve the pressure on the SWO pin. Unselect the element in the Trace Configuration window to reduce such overloads.

2) Watchpoints: (also known as Access Points)

This is an excellent opportunity to show how the Watchpoints work in Keil µVision.

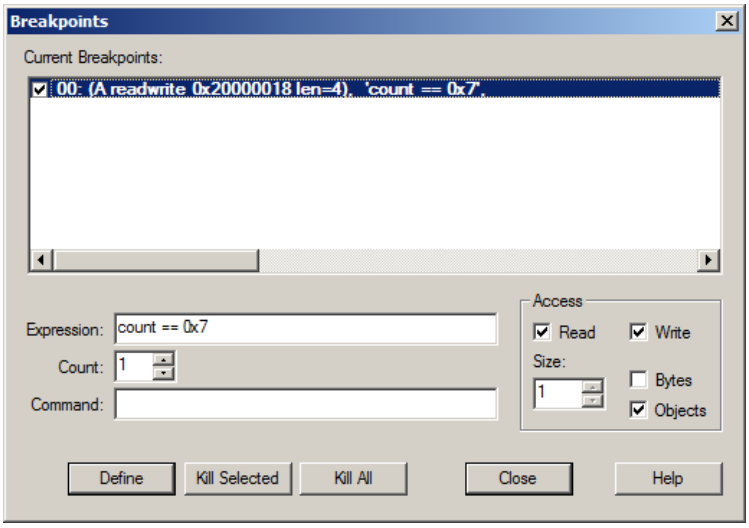
Watchpoints are also known as Access Breaks and are indicated by an **(A** in Figure 16. They are set in the Breakpoints Window. Watchpoints can be thought of as conditional breakpoints when compared to an Execution breakpoint. Execution breakpoints stop the CPU when a specified instruction is fetched (but not executed). Watchpoints stop the CPU when a specified data access occurs and any specified expression becomes true. There might be some Program Counter skid. This is useful for locating read and write operations you do not expect. Watchpoints are not intrusive.

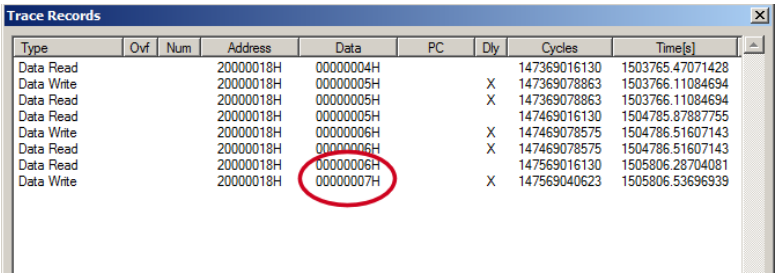
SmartFusion has 4 Watchpoints. Watchpoints must be configured in Debug mode and with the CPU halted.

TIP: You can set only one data comparison Watchpoint as described below in a Cortex-M3 processor.

How to set a Watchpoint:

1. Stop the program and remain in Debug mode.
2. Open the Breakpoint window through View/Breakpoints or Ctrl-B. The window below opens up:
3. Watchpoints are created in the bottom half and are defined and listed in the top half of this window. The screen below is a composite view.
4. Enter count == 0x07 in the bottom half of the Breakpoints window: Click both Read and Write.
5. Click on Define and the Watchpoint moves to the upper half of the window.
6. The Watchpoint is now configured to: the program will be halted if the value of 0x07 is written to or read ***once*** from the variable count. (Count = 1) The other entries are the default settings.
7. Click on Close.
8. Set `count` to something other than 0x07 in the Watch or Memory window.
9. The Trace Records window can be open. Double-click on it to clear it.
10. Click on RUN and wait for the program to stop. 
11. Scroll to the bottom of the Trace Records window and the data write is displayed as in Figure 17.
12. Enter Ctrl-B and delete (kill) this Watchpoint else it will interfere with your other examples. Click on Close.



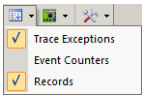


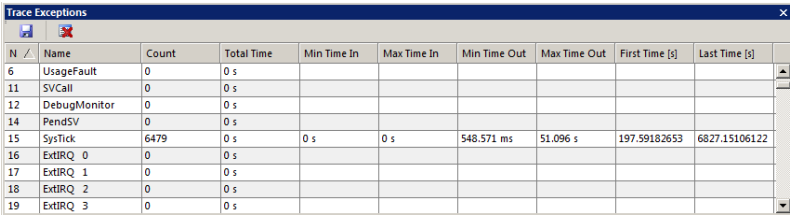
4) Exception Tracing: *This feature uses SWV which must be configured properly:*

Serial Wire Viewer displays exceptions and interrupts in real-time without stealing any CPU cycles or needing instrumentation code. Note that ARM interrupts are a subset of exceptions. SWV displays informative information in real-time.

The Keil examples often have the SysTick timer running. We will use this exception as an example.

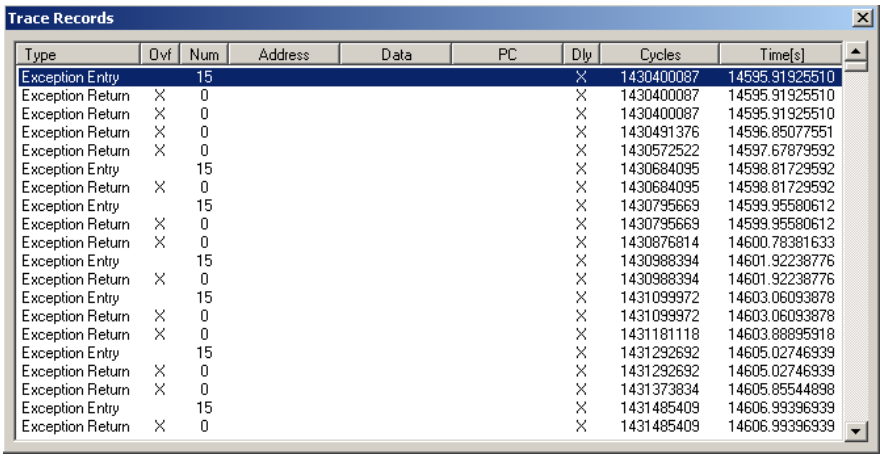
1. Assume Blinky.c is loaded and µVision is in debug mode. The CPU can be running or not.
2. Open the Trace Configuration window by clicking on Debug/Debug Settings.
3. Enable EXCTRC: Unselect Periodic and On R/W Sampling. Click on OK twice to return.
4. Click on Setup: in the LA and delete all variables. Click on Close.
5. Click on RUN to start the program.
6. Open the Exception Trace window and ensure the Trace Records window is still open.
7. The Exception Trace window will display the SysTick timer 15 as shown below: Double-click to reset.
8. Scroll up and down and you can see the listing of all potential exceptions available in the NVIC.





Trace Exceptions showing the SysTick Timer.

1. Note the X in the OVF column. Some frames were lost due to overloading the SWO pin.
2. Remove TimeStamp in the Trace Configuration window. Return and run the program and overflows will be gone. But so will the timestamps.



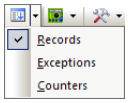
Exceptions in the Trace Records window.

Explanation of Exception frames:

- **Entry:** when the exception enters.
- **Exit:** When it exits or returns.
- **Return:** When all the exceptions have returned including any Cortex-M3 tail-chaining.

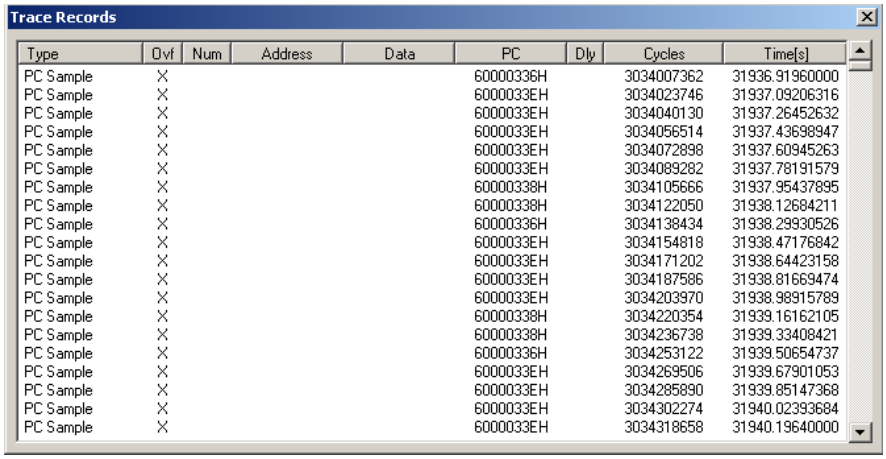
5) PC Samples Tracing: *This feature uses SWV which must be configured properly:*

µVision can display a sampling of the program counter values in the Trace Records window. PC Samples is configured in the Trace Records window. This window is opened with View/Trace/Records or the drop down menu shown here:



1. Open the Trace Configuration window by clicking on Debug/Debug Settings.
2. Enable PC Samples by checking Periodic in the PC Sampling area.
3. Disable EXCTRC:
4. Click on OK twice to return.
5. Click on RUN to start the program.
6. Open the Trace Records window. PC Samples will be displayed as shown as below:

The values displayed in the PC column gives you an indication where the program is spending its time.



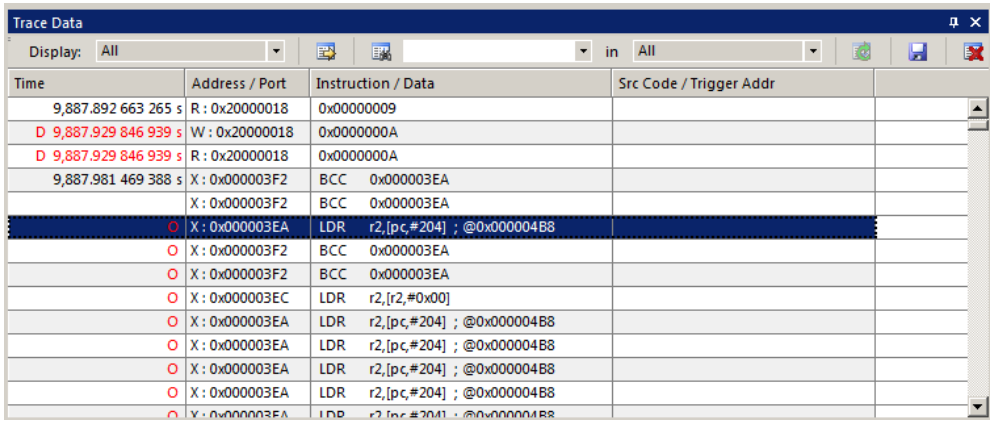
Type	Ovf	Num	Address	Data	PC	Dly	Cycles	Time[s]
PC Sample	X				60000336H		3034007362	31936.91960000
PC Sample	X				6000033EH		3034023746	31937.09206316
PC Sample	X				6000033EH		3034040130	31937.26452632
PC Sample	X				6000033EH		3034056514	31937.43698947
PC Sample	X				6000033EH		3034072898	31937.60945263
PC Sample	X				6000033EH		3034089282	31937.78191579
PC Sample	X				60000338H		3034105666	31937.95437895
PC Sample	X				60000338H		3034122050	31938.12684211
PC Sample	X				60000336H		3034138434	31938.29930526
PC Sample	X				6000033EH		3034154818	31938.47176842
PC Sample	X				6000033EH		3034171202	31938.64423158
PC Sample	X				6000033EH		3034187586	31938.81669474
PC Sample	X				6000033EH		3034203970	31938.98915789
PC Sample	X				60000338H		3034220354	31939.16162105
PC Sample	X				60000338H		3034236738	31939.33408421
PC Sample	X				60000336H		3034253122	31939.50654737
PC Sample	X				6000033EH		3034269506	31939.67901053
PC Sample	X				6000033EH		3034285890	31939.85147368
PC Sample	X				6000033EH		3034302274	31940.02393684
PC Sample	X				6000033EH		3034318658	31940.19640000

PC Samples from the ULINK2 or ULINK-ME

TIP: PC Samples are very liable to overload the SWO pin because of the relatively low 98 KHz speed of the current A2F200 silicon. The A2F500 displays PC Samples much easier. All other functions, including the Logic Analyzer, must be turned off in order for the PC Samples to update with the A2F200. Otherwise, the program must be stopped to display the PC Samples. Note the X in the Ovf column indicates many PC Sample frames were lost. This is still useful information.

The A2F500 SmartFusion can display PC Samples without any overloading conditions. It is still a good idea to limit those SWV features selected to only those you actually need to minimize overloading and subsequently lost frames.

ULINK*pro* Instruction Trace Window: The ULINK*pro* provides the Instruction Trace window shown below. Shown are a Data Write, two reads and eleven PC Samples. Note disassembled instructions are shown and if available, source code will also be displayed. If you double-click on a PC Sample you will be taken to that instruction in the disassembly and/or source window. The ULINK*pro* can process SWV data very fast and therefore is much less susceptible to data overruns.



Trace Data window from ULINK*pro*.

6) Debug (printf) Viewer: *This feature uses SWV which must be configured properly:*

ITM Stimulus 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 the µVision Debug (printf) Viewer for display. You write ASCII values to Port 0.

Add the *printf* Source Files to Blinky:

1. Stop the program and exit debug mode if necessary.
2. Add this code to Blinky.c. A good place is right after the place where you declared count variable:
   This line is a pointer to ITM Port 0):

```
#define ITM_Port8(n)   (*((volatile unsigned char *)(0xE0000000+4*n)))
```

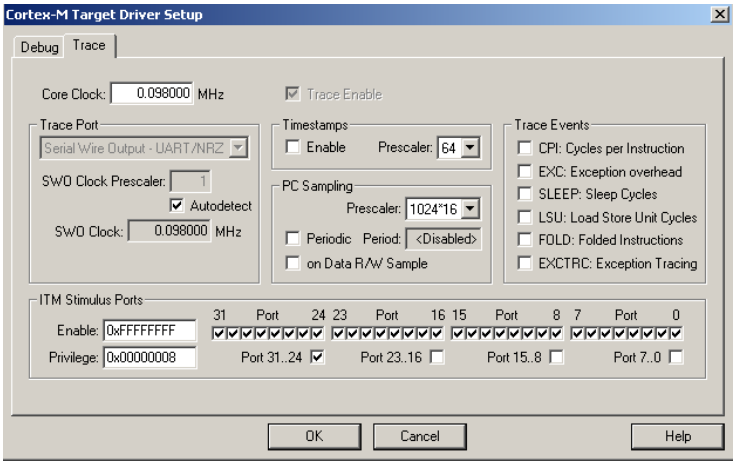
3. Near line 51 enter these three lines: Just after the line `count++;` that you entered is a good place. This is not critical.

```
ITM_Port8(0) = count + 0x30;
while (ITM_Port8(0) == 0);
ITM_Port8(0) = 0x0A;
while (ITM_Port8(0) == 0);
ITM_Port8(0) = 0x0D;
```

4. Rebuild the source files  , program the Flash memory  and enter Debug mode  .

Configure the Serial Wire Viewer (SWV):

5. SWV should be configured from previous steps.
6. Open Debug/Debug Settings and select the Trace tab. Confirm ITM Port 0 is selected, unselect Periodic, EXCTRC, on Data R/W Sample and TimeStamps Enable as shown here: ITM Port 0 is the most important item here.

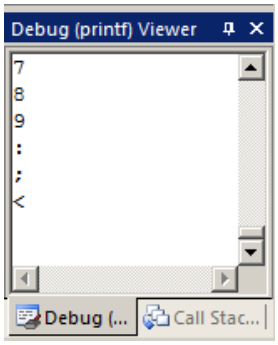


Open The Debug Viewer:

7. Click on View/Serial Windows and select Debug (printf) Viewer.
8. Make sure Periodic Update is enabled or this window will only update when you stop the program.

Run Blinky:

9. Click on RUN  .
10. In the Debug (printf) Viewer you will see the value of `count` appear every few seconds as shown below: You could add more ASCII characters to format your own messages.



Trace Records

These writes to the ITM will be displayed in the Trace Records window.

Open the Trace Records window and ITM 0 frames will be visible.

ITM Conclusion

The writes to ITM Stimulus Port 0 are intrusive and are usually one cycle. It takes zero additional CPU cycles to get the data displayed in the Debug (printf) Viewer.

This method is much faster than using a standard UART. No UART hardware is used.

TIP: You have turned off the Timestamps in the Trace Configuration. Some SWV items need them enabled. If you turn off Timestamps and the item you are looking for also stops, you will need to re-enable them.

This is the end of this set of exercises. Stop the program  **and exit Debug mode**  **.**

TIP: ITM_SendChar is a Keil supplied function you can use to send characters to the Debug Viewer. It is found in the header core_CM3.h in the directory C:\Keil\ARM\RV31\INC.

1) RTX: Keil's RTOS: RTX_Blinky example:

Keil has its own full feature RTOS called RTX. It is provided as part of the Keil MDK full tool suite for no charge and can have up to 255 tasks. RTX now has a BSD type license. This means it is free. See www.keil.com/demo/eval/rtx.htm.

RTX source code is included in MDK. See www.arm.com/cmsis. µVision will work with any RTOS.

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 memory area by the RTOS. Keil provides two Task Awareness windows for RTX and they both update live.

The Tasks and System window uses the DAP (Debug Access Port) which is also used by the Watch and Memory windows to obtain its information. No configuration is needed to get this feature enabled. The Event Viewer uses the SWV ITM Stimulus Port 0. This means SWV must be configured to get this window working.

RTX is easy to configure with SmartFusion. This integration is made easy by the Cortex-M3 architecture.

Open and Run RTX_Blinky:

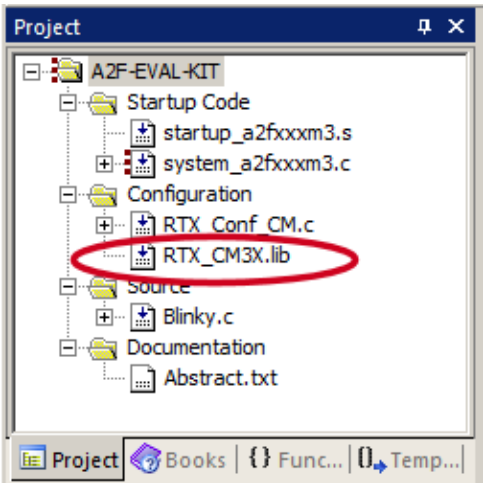
1) Select Project/Open Project and open: C:\Keil\ARM\Boards\Actel\SmartFusion\RTX_Blinky\Blinky.uvproj.

Select SWD instead of JTAG:

2) Select Target Options. Click on the Debug tab.
3) The ULINK2/ME will be selected. If you are using a different adapter, select it now.

TIP: If you change the adapter remember you must also change it in the Flash programmer under the Utilities tab.

4) Click on Settings:
5) Select SWJ and SW to use SWD and not JTAG.



Add the SmartFusion RTX Library file:

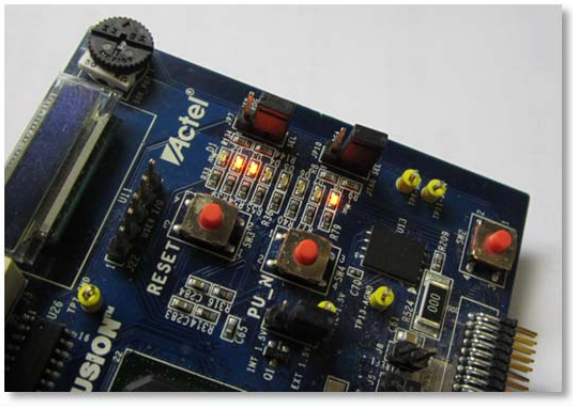
6) In the Project window a shown here: right click on Configuration.
7) Select **Add Files to Group 'Configuration'…**
8) In the window that opens: select in Files of type: Library file *.lib.
9) Go to C:\Keil\ARM\RV31\LIB and highlight the file RTX_CM3X.lib.
10) Click on Add (only once) and then Close.
11) RTX_CM3X.lib should display as shown here:

Compile and Run the RTX Program:

12) Compile and load the project into the eNVM Flash.
13) Enter Debug mode and click on the RUN.
14) Four leds will blink sequentially and one clock led. See the photo below:
15) If they do not: The wrong or no STAPL file is loaded or the program.
16) If you click on STOP and if the program counter is on near line 145 for (;;) in the function __task void os_idle_demon (void), RTX is probably running correctly.
17) Leave RTX_Blinky running for the next steps.

TIP: If RTX_Blinky does not run and when you stop it the PC is on the Hard Fault vector, the library file RTX_CM3X.lib is not entered as described above correctly. See www.keil.com/support/docs/3551.htm for an explanation.

This step will be incorporated into a future version of MDK to make this library file automatically selected at compile time.

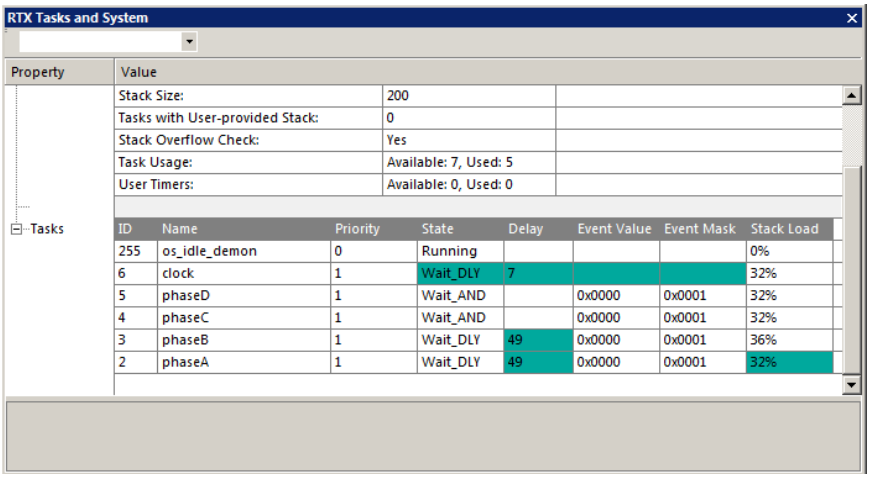


In this photo the program is stopped when the Clock led (D8 on right) is turned on. The two left leds that are on (D2 and D3) phaseB and phaseC tasks respectively.

2) RTX Viewers: RTOS Kernel Awareness windows:

a) RTX Tasks and System:

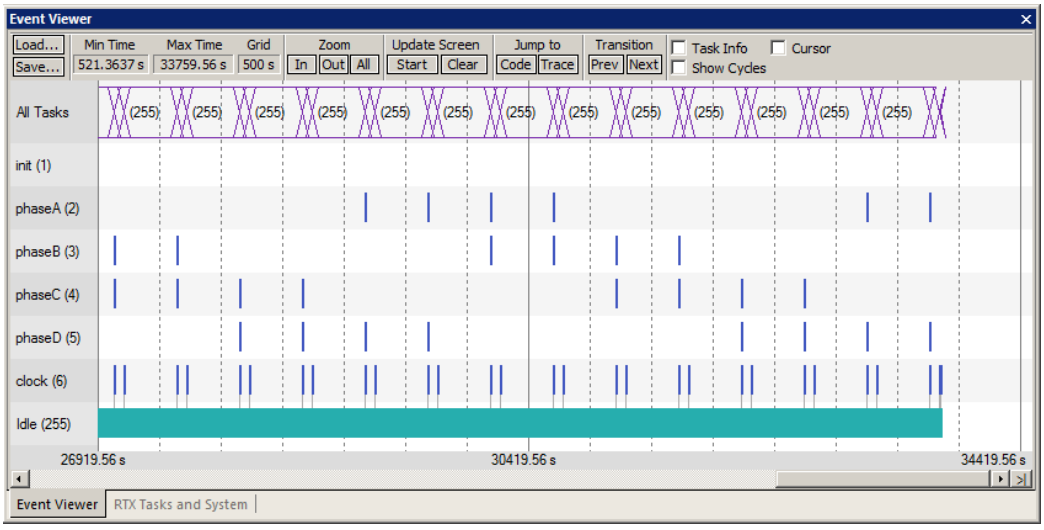
1. Open Debug/OS Support and select RTX Tasks and System and the window below opens up. RTOS visibility is updated in real-time using the same CoreSight DAP technology as used to update the Watch and Memory windows. You probably have to drag this window into the open to conveniently view it.



2. The RTX example that is running simulates a stepper motor controller with four phases.
3. Note as RTX switches task, only the idle daemon is displayed as running. This because most time is spent here.
4. In Blinky.c, find the four tasks phaseA through phaseD: they are near lines 46, 58, 70 and 82 respectively.
5. Set a breakpoint anywhere in each of these four tasks (except for the line **`for (;;) {`**) where there is a grey block which indicates assembly is present.
6. The program will soon stop in a task and this will be reflected in the RTX Tasks and System window: Running.
7. Click on RUN and each task in turn will show as Running when the program was stopped by the breakpoints.
8. Remove all breakpoints when you are done. You can use Ctrl-B Kill All or click on each breakpoint in Blinky.c.
9. If the RTX Tasks and System window is updated only when you stop the program: check View/Update Periodic Window. This setting will not affect the Event Viewer.

B) Event Viewer:

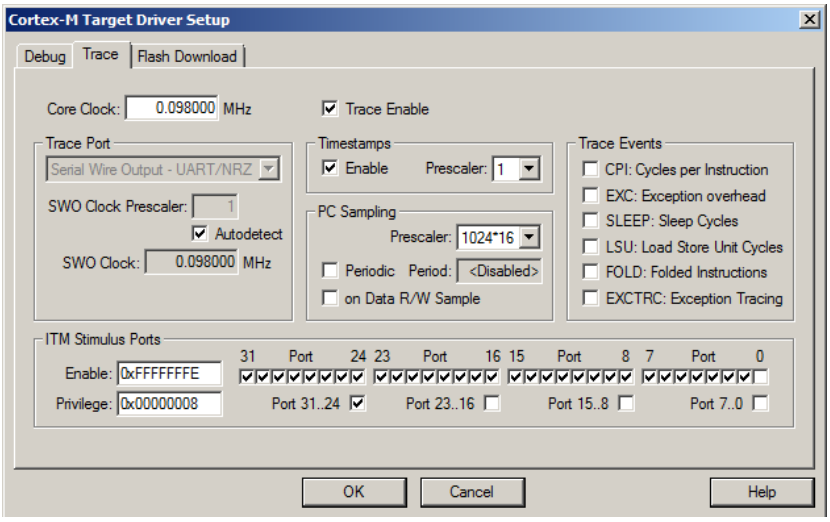
1. Open Debug/OS Support and select Event Viewer and the window below opens up. You probably do not see any data as shown below because the Event Viewer needs SWV working. SWV is not configured yet.
You will do this on the next page.
2. Note the tabs at the bottom left to switch between the two windows.



3) Configure SWV for the Event Viewer:

Configure SWV Trace:

1) Click on STOP and exit Debug mode.
3) Click on Target Options. Select the Debug tab and then click the Settings: button. Select the Trace tab.
4) This window opens up: Set Core Clock to 0.098 for EVAL-KIT and 50 for DEV-KIT (75 J-Link). Set Trace Enable.
5) Select ITM Port 31, Timestamp Prescalar to 1. Unselect EXCTRC. Everything else as default as shown below:
6) Click OK twice to return.
7) SWV is now configured.



Run RTX_Blinky:

8) Enter Debug mode.
9) Click on RUN.
10) The Event Viewer will work now.
11) Click on the ALL icon and adjust the range with IN and OUT Zoom.

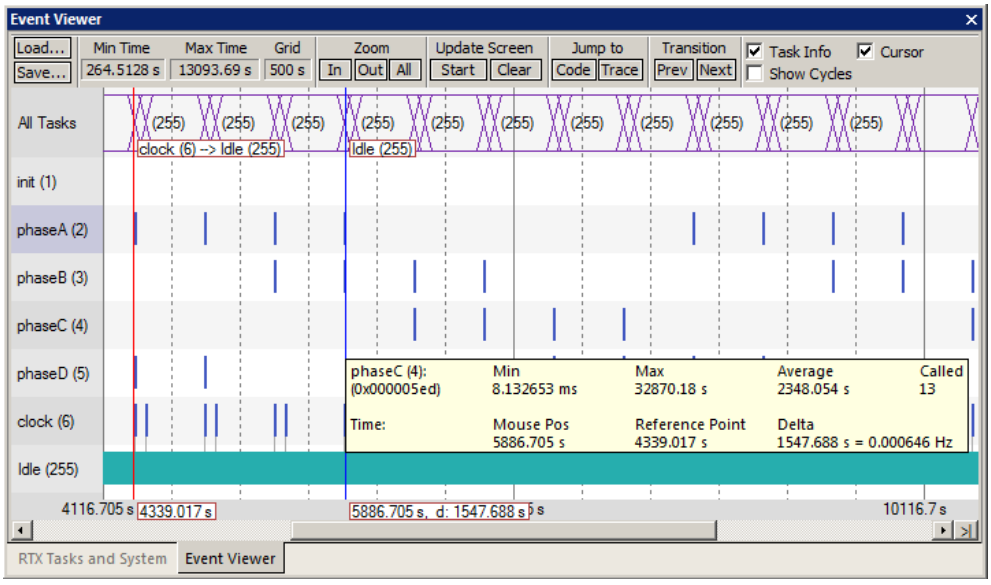
RTX_Blinky Timings:

2) Select Task Info and Cursor in the Event Viewer as shown below.
3) Click Stop in the Update Screen box.
4) Click on a point and move the cursor. Note the Timing box appears.
5) In this case: the Delta represents 1547/100,000,000*98000 = 1.5 seconds which is the time a led is on. The big space after is 2.5 seconds the led is on.

TIP: The reason for the time discrepancy is µVision uses the Core Clock: value you enter to calculate this. Since the CPU is running at 100 MHz, and the Core Clock is 0.098 MHz: the times will be off by this ratio. ULINK*pro* does not use Core Clock: to calculate the SWO pin frequency so you can enter 100 MHz into Core clock and the correct times are then displayed. This is an issue with the A2F200 parts because the SWO pin is fixed at 98 KHz. A2F500 timing values should be divided by 2. This is because the Core Clock: value is 50 MHz while the CPU is running at 100 MHz. ULINKpro is not affected.

TIP: To view the exception SysTick, enable EXCTRC in the Trace Configuration window. Select View/Trace and select the Exception window. SysTick will also be displayed in the Instruction Trace window. This could overload the SWO pin and corrupt some values in the Event Viewer. This is a bigger issue with the A2F200 processor and not so much on the A2F500. SysTick is a dedicated Coretx-M3 timer that is used here to perform the context switches in RTX.

TIP: The RTX Tasks and System window gets its data from the DAP read/write port. This is not SWV. The Event Viewer gets its data from SWV therefore SWV must be configured. Remember A2F200 Core Clock: = 0.098 MHz and A2F500 = 50 MHz when using a ULINK2 or a ULINK-ME. An A2F500 with a J-Link is 75 MHz.

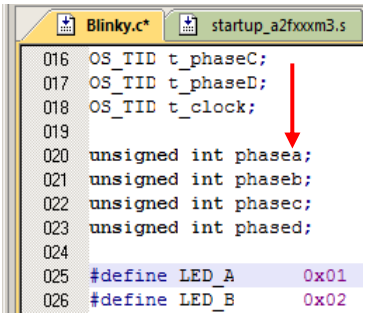


Note:
The Event Viewer is good at showing you *at a glance* if your RTX timings and context switches happen as you configured them to.

4) Logic Analyzer Window: View RTX task variables real-time in a graphical format:

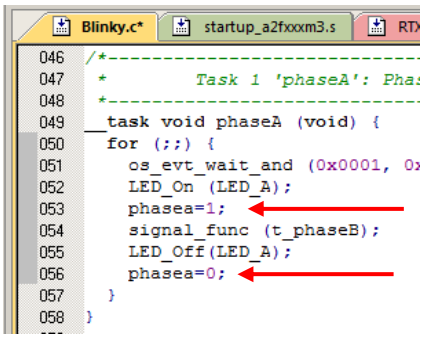
µVision has a graphical Logic Analyzer window. Up to four variables can be displayed in real-time using the Serial Wire Viewer in the SmartFusion. RTX_Blinky uses four tasks to create the waveforms. We will graph these four waveforms.

1. Stop the program Close both the RTX Viewer windows. Exit debug mode .
2. Add 4 global variables **`unsigned int phasea`** through **`unsigned int phased`** to Blinky.c as shown here:
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 053 and 056 (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.



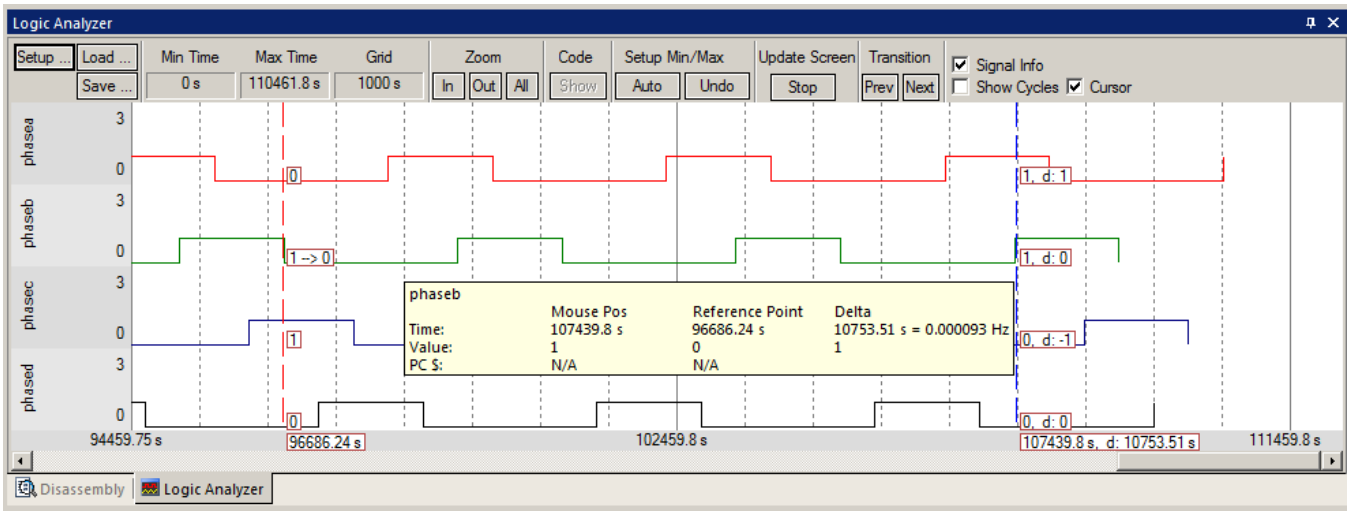
TIP: The Logic Analyzer can display static and global variables, structures and arrays. It can't see locals: just make them global or static. To see peripheral registers merely read or write to them and enter them into the Logic Analyzer.

5. Rebuild the project. Program the Flash .
6. Enter debug mode .
7. You can run the program at this point.



Enter the Variables into the Logic Analyzer:

8. Right click on your variable **phasea** and select Add 'phasea" to… and then select Logic Analyzer. The Logic Analyzer window will open if not already.
9. 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.
10. Click on the Setup icon and click on each of the four variables and set Max. in the Display Range: to 0x3.
11. Click on Close to go back to the LA window. Resize this window for an appropriate height.
12. Using the OUT and In buttons set the range to 1000 seconds. (or 1000 sec for the A2F500).
13. You will see the following waveforms appear. Click to mark a place. See below. Select Signal Info and Cursor. Hover over one of the waveforms and get timing and other information as shown in the inserted box labeled phaseb:
14. These timing values will have the same issues as discussed on the previous page.
15. Stop the program and exit Debug mode when you are finished.



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

1) DSP SINE example using ARM CMSIS-DSP Libraries:

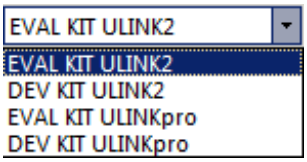
ARM CMSIS-DSP libraries are offered for ARM Cortex-M3 processors. DSP libraries are provided in MDK in C:\Keil\ARM\CMSIS. README.txt describes the location of various CMSIS components. See www.arm.com/cmsis and forums.arm.com for more information. CMSIS is an acronym for Cortex Microcontroller Software Interface Standard.

This example creates a **sine** wave, then a second to act as **noise**, which are then added together (**disturbed**), and then the noise is filtered out (**filtered**). The waveform in each step is displayed in the Logic Analyzer using Serial Wire Viewer.

This example incorporates Keil RTX. RTX is available free with a BSD type license. RTX source code is provided in MDK.

To obtain this example file, go to **www.keil.com/appnotes/docs/apnt_208.asp** Extract \DSP to …\Boards\Actel\SmartFusion\

1. Open the project file sine: C:\Keil\ARM\Boards\Actel\SmartFusion\DSP\sine.uvproj.
2. Select the board and debug adapter you are using in the Target selection menu:



3. Compile the source files by clicking on the Rebuild icon.  .
4. Program the SmartFusion eNVM flash by clicking on the Load icon:  Progress is indicated in bottom left corner.
5. Enter Debug mode by clicking on the Debug icon.  Select OK if the Evaluation Mode notice appears.

TIP: The default Core Clock: is 98 KHz (0.098 MHz) for the EVAL KIT and 50 MHz (75 for J-Link) for the DEV KIT.

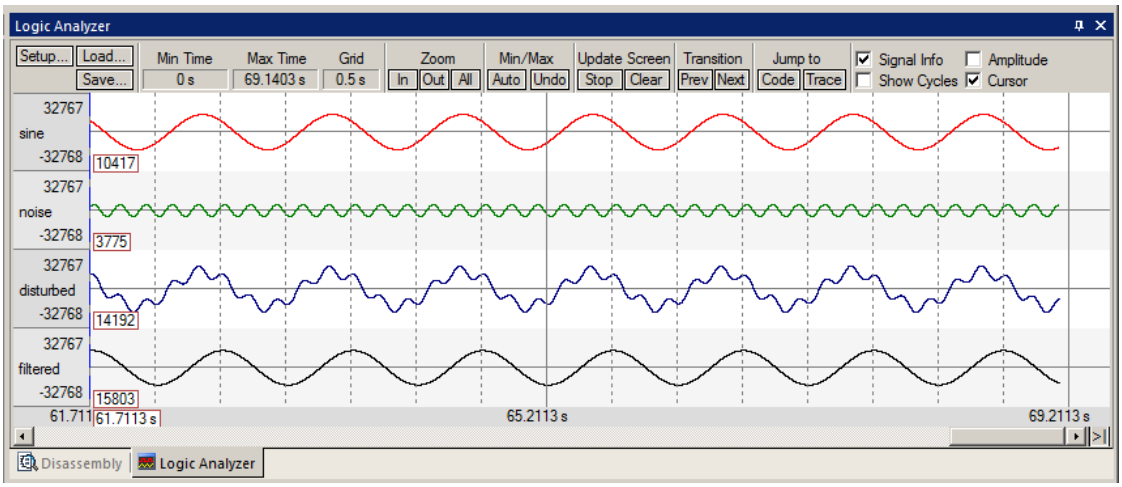
6) Click on the RUN icon.  Open the Logic Analyzer window.  Size it appropriately.

Four waveforms will be displayed in the Logic Analyzer using the Serial Wire Viewer as shown below. Adjust Zoom Out for an appropriate display. Displayed are 4 global variables: **`sine`**, **`noise`**, **`disturbed`** and **`filtered`**.

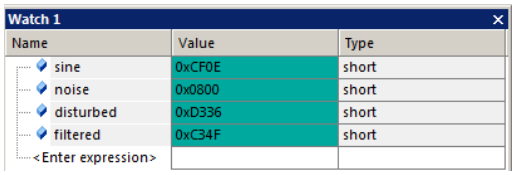
TIP: If one or two variables shows no waveform: the SWO pin is overloaded. If Disturbed has no data, remove Sine. The issue is the CoreSight Trace implemented in the A2F200 and A2F500 parts outputs both Reads and Writes out the SWO pin.

7) Open the Trace Records window and you can see the Read and Write operations. If you are using a ULINK*pro*, you must stop the program to update the Data Trace window.

6. The project provided has Serial Wire Viewer configured and the Logic Analyzer loaded with the four variables.



7. Select View/Watch Windows and select Watch 1. The four variables are displayed updating as shown below: They are pre-configured in this project.
8. Leave the program running.
9. Close the Trace Records window.

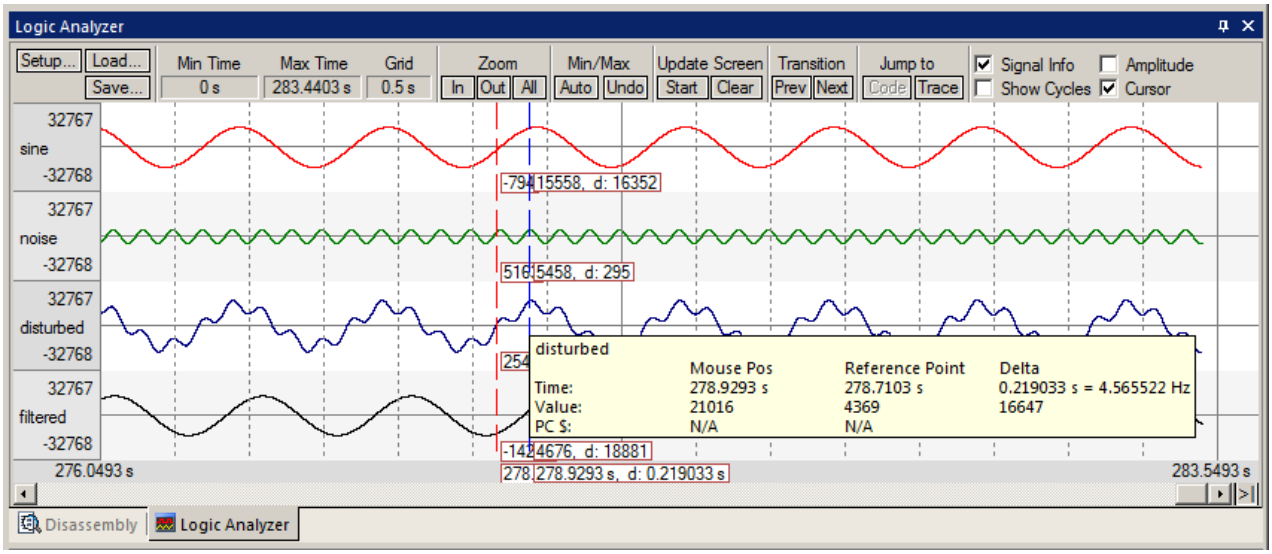


TIP: The ULINK*pro* trace display is different and the program must be stopped to update it.

2) Displaying DSP variables in the Logic Analyzer (LA):

1. In the LA window, select Signal Info, Show Cycles, Amplitude and Cursor.
2. Click on STOP in the Update Screen box. You could also stop the program but leave it running in this case.
3. Click somewhere in the LA to set a reference cursor line.
4. Note as you move the cursor various timing information is displayed as shown below:

TIP: The timings with AF200 will not be correct with a ULINK2/ME since the 98 KHz SWO speed is not set relative to the CPU speed. You must adjust times by: (time displayed)/100,000,000*98000. Divide the timing values displayed by 2 with the A2F500 processors. ULINK*pro* can be set to 100 MHz and will display the timings correctly.



3) RTX Tasks and System Awareness window:

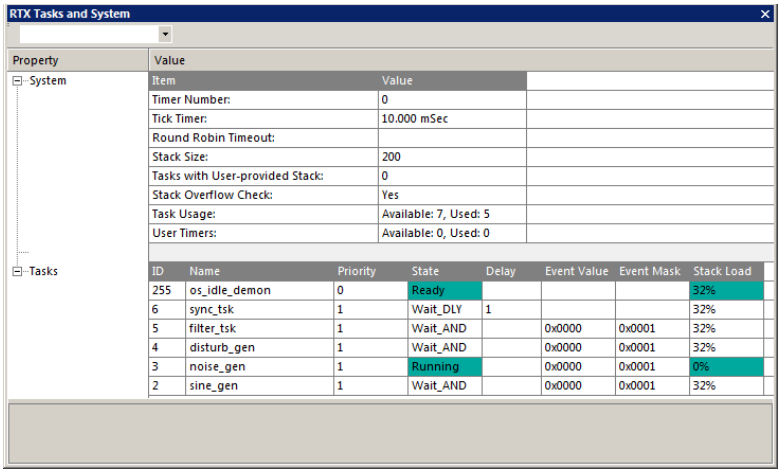
5. Click on Start in the Update Screen box to resume the collection of data.
6. Open Debug/OS Support and select RTX Tasks and System. A window similar to below opens up. You probably have to click on its header and drag it into the middle of the screen.
7. Note this window does not update: nearly all the processor time is spent in the idle daemon: it is shown as Running. The processor spends relatively little time in other tasks. You will see this illustrated clearly on the next page.
8. Set a breakpoint in one of the four tasks in DirtyFilter.c by clicking in the left margin on a grey area.
9. Click on Run and the program will stop here and the Task window will be updated accordingly. Below, I set a breakpoint in the noise_gen task: You can set a breakpoint on the other three tasks if you like.
10. Clearly you can see that noise_gen was running when the breakpoint was activated.
11. Remove the breakpoint(s).
12. Close RTX Tasks and System window.

TIP: Do not exit Debug mode with this window open. The Tasks will collapse and you will not be able to expand it. We are looking into this issue.

TIP: Remember you can set hardware breakpoints while the program is running.

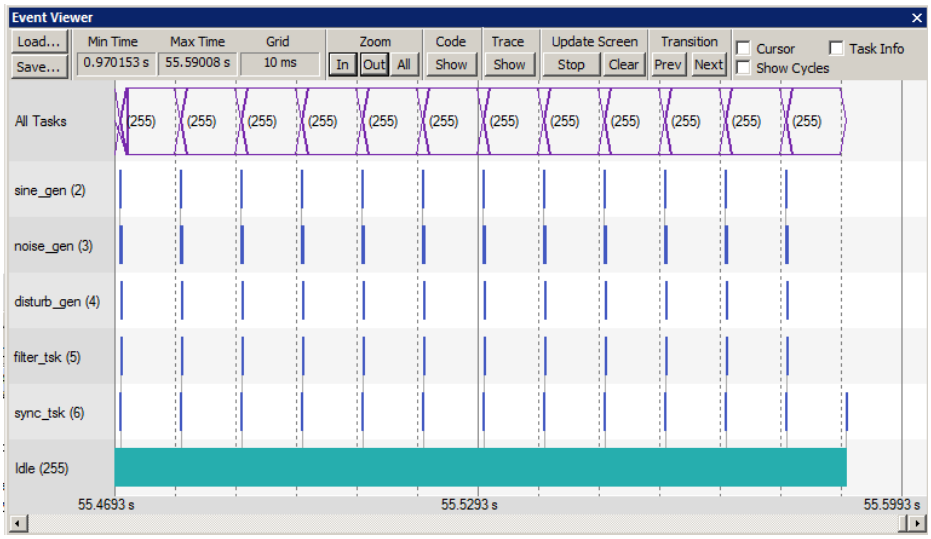
TIP: Recall this window uses the CoreSight DAP read and write technology to update this window. Serial Wire Viewer is not used and is not required to be activated for this window to display and be updated.

The Event Viewer does use SWV and this is demonstrated on the next page.



4) RTX Event Viewer:

1. Stop the program. Click on Setup... in the Logic Analyzer. Select Kill All to remove all variables. This is necessary because the SWO pin will likely be overloaded when the Event Viewer is opened up and inaccuracies might occur. If you like – you can leave the LA loaded with the four variables to see what the Event Viewer will look like. Later, delete them to see the effect, if any, on the Event Viewer.
2. Select Debug/Debug Settings.
3. Click on the Trace tab.
4. Enable ITM Stimulus Port 31. Event Viewer uses this to collect its information.
5. Click OK twice.
6. Make sure RTX Tasks and System is closed.
7. Exit and re-enter Debug mode to refresh the Trace Configuration.
8. Click on RUN.
9. Open Debug/OS Support and select Event Viewer. The window here opens up: Note the idle daemon is a solid bar indicating the CPU spends most of its time here.

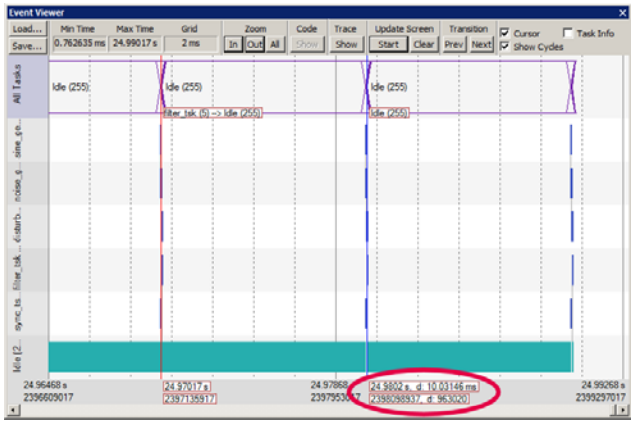


10. Note there is no Task 1 listed. Task 1 is main_tsk and is found in DirtyFilter.c near line 169. It runs some RTX initialization code at the beginning and then deletes itself with os_tsk_delete_self(); found near line 188.

TIP: If Event Viewer is blank or erratic, or the LA variables are not displaying or blank: this is likely because the Serial Wire Output pin is overloaded and dropping trace frames. Solutions are to delete some or all of the variables in the Logic Analyzer to free up some SWO or Trace Port bandwidth. It depends on how much data is sent to the ports. You might not be able to get Event Viewer working with a A2F200 part because of the SWV speed limitation.

ULINK*pro* is better with SWO bandwidth issues. These have been able to display both the Event and LA windows. ULINK*pro* uses the faster Manchester format than the UART mode that ST-Link, ULINK2 and J-Link uses.

11. Note on the Y axis each of the 5 running tasks plus the idle daemon. Each bar is an active task and shows you what task is running, when and for how long.
12. Click Stop in the Update Screen box.
13. Click on Zoom In so three or four tasks are displayed.
14. Select Cursor. Position the cursor over one set of bars and click once. A red line is set here:
15. Move your cursor to the right over the next set and total time and difference are displayed.
16. Note, since you enabled Show Cycles, the total cycles and difference is also shown.



The 10 msec shown is the SysTick timer value. This value is set in RTX_Conf_CM.c.

TIP: ITM Port 31enables sending the Event Viewer frames out the SWO port. Disabling this can save bandwidth on the SWO port if you are not using the Event Viewer and this is normally a good idea if you are running RTX with high SWO loading.

Even if the Event Viewer is closed, the data is still being sent out the SWO pin or the Trace Port. This will contribute to SWV overloading.

TIP: RTX is easily configured using the uVison Configuration Wizard. Open file RTX_Conf_CM.c and click on the Config Wizard tab. Expand the items and note how easy it is to set RTX.

1) Creating a new project from scratch: Using the Blinky source files:

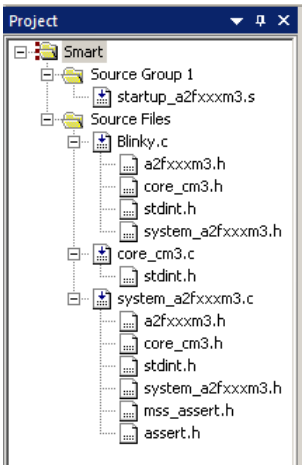
All examples provided by Keil are pre-configured. All you have to do is compile them. You can use them as a starting point for your own projects. However, we will start this example project from the beginning to illustrate how easy this process is. We will use the existing source code files so you will not have to type them in. Once you have the new project configured; you can build, load and run the Blinky example as usual. You can use this process to create any new project from your own source files created with µVision's editor or any other editor.

Create a new project called Mytest:

1. With µVision running and not in debug mode, select Project/New µVision Project.
2. In the window Create New Project go to the folder C:\Keil\ARM\Boards\Actel\SmartFusion.
3. Right click and create a new folder by selecting New/Folder. I named this new folder FAE.
4. Double-click on the newly created folder "FAE" to enter it. You should be in this folder now: C:\Keil\ARM\Boards\Actel\SmartFusion\FAE
5. Name your project. I called mine Mytest. You can choose your own name.
6. Click on Save.
7. "Select Device for Target 1" window opens up.
8. This is the Keil Device Database® which lists all the devices Keil supports (plus some secret ones).
9. Locate the Microsemi directory, open it and select A2F200M3F. Note the device features are displayed.
10. Click on OK.

11. A window opens up asking if you want to insert the default SmartFusion startup code to your project. Click on "Yes". This will save you a great deal of time.
12. In the Project Workspace in the upper left hand of µVision, open up the folders by clicking on the "+" beside each folder.
13. We have now created a project called Mytest and the target hardware called Target 1 with one source file startup_A2FxxxM3.s.
14. Click once (slowly) on the name "Target 1" (or twice slowly if not already highlighted) in the Project Workspace and rename Target 1 to something else. I chose Smart as shown in Figure 30. Press Enter to accept this. Note the Target selector also changes to Smart. Click on the + to open up the directory structure if necessary. You can create many target hardware configurations including a simulator and easily select them.

Select the source files:



1. Using MS Explorer open C:\Keil\ARM\Boards\Actel\SmartFusion\Blinky\.
2. Copy Blinky.c, core_cm3.c and system_A2FxxxM3x.c to the folder C:\Keil\ARM\Boards\Actel\SmartFusion\FAE\.
3. In the Project Workspace in the upper left hand of µVision, right-click on "Smart" and select "Add Group". Name this new group "Source Files" and press press Enter.
4. Right-click on "Source Files" and select **Add files to Group "Source Files**".
5. Select the files Blinky.c, core_cm3.c and system_A2FxxxM3.c and click on Add (once) and then Close. These will show up in the Project Workspace when you click on the + beside Source Files. You can enter these separately or block them and add add them together. Just makes sure you don't add them more than once.
6. Select Target Options  and select the Debug tab. Select ULINK2/ME Cortex Debugger. Enable this selection by checking the circle just to the left of the word "Use:".
7. Click on the Settings: button. Select both SWJ and SW. Click close twice to return.
8. Flash programming in the Utilities tab is probably already set to the eNVM.
9. Select Files/Save All.
10. At this point you could build this project. You can program the flash and run it on your SmartFusion board.
11. You will be able to add source files the same way you added Blinky.c. You can use Blinky.c as a template for your own main() program.

2) Running Blinky from RAM:

It is possible to run your program in the SmartFusion Cortex-M3 RAM rather than Flash.

µVision makes it easy to create different target settings. We will create one using RAM in the Blinky project.

There are six simple steps to accomplish this:

1) Create a New Target Configuration in µVision:

1. Open the project Blinky.uvproj. Stay in Edit mode.
2. Open Project from the main menu and select Manage and then select Components, Environment, Books…
3. In the Project Targets box, click on the Insert icon. 
4. Enter the name of your target (I chose RAM) and press the Enter key. Click on OK.
5. In the main menu, click on the Select target window and select your RAM target: 

6. You will now configure the target RAM which will be separate from A2F200M3F Flash.

2) Create an initialization file to configure and load SmartFusion when Debug mode is entered:

7. We will use µVision to create an initialization file to configure SmartFusion to use RAM.
8. Select File/New and a blank text window will be created. Enter the following text:

```
FUNC void Setup (void) {
  _WDWORD(0x40006010, 0x4C6E55FA);      // Disable watchdog
  SP = _RDWORD(0x60080000);             // Setup Stack Pointer
  PC = _RDWORD(0x60080004);             // Setup Program Counter
  _WDWORD(0xE000ED08, 0x20000000);      // Setup Vector Table Offset Register
}
LOAD %L INCREMENTAL                     // Download
Setup();                                // Setup for Running
g, main
```

9. Select File/Save As and enter ram.ini and press the Enter key or click on the Save button.

3) Enter the memory addresses into µVision:

10. Open the Target Options menu.  The Target tab will be selected as shown below:
11. Read/Only Memory Areas configures the program space. Read/Write Memory Areas allocates data.
12. We will divide up the SmartFusion RAM (64K in this case) between the program and data spaces.
13. RAM in the SmartFusion we are using starts at 0x2000_0000 and ends at 0x2001 0000. (64 K bytes)
14. Enter ROM Start 0x2000 0000 and size 0x8000 as shown in Figure 31. Enter RAM Start 0x2000_8000 and size 8000 as shown in Figure 31. ***Do not click on OK yet.***

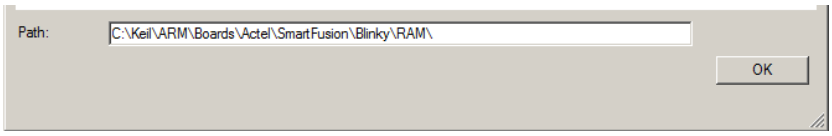
Configuring Memory Area (partial screen)

15. 32K of RAM is now available for your program code and 32 K of RAM for data. You can use other values. You must make sure there is enough RAM for your program and variables to fit into.

The rest of the configuration for RAM operation is on the next page. Do not clock on OK yet.

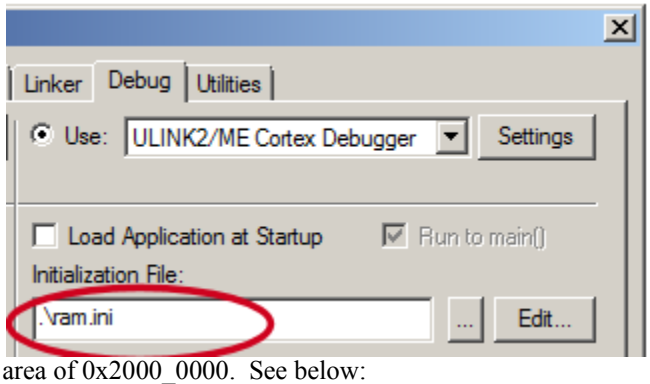
4) Create a directory for the compiled files to be stored:

16. Now we will create a directory for the compiled files. Select the Output tab.
17. Click on the Select Folder for Objects… box.
18. This window will be pointing to C:\Keil\ARM\Boards\Actel\SmartFusion\Blinky\Flash.
19. Click on the Up one directory icon to point to C:\Keil\ARM\Boards\Actel\SmartFusion\Blinky\
20. Right-click in this window and select New and then Folder to create a new folder.
21. Name this new folder RAM and press Enter.
22. Double-click on this new folder to add it in the Path box: C:\Keil\ARM\Boards\Actel\SmartFusion\Blinky\RAM\ See below for the Path box: Click on OK. Your compiled files will go here. Click OK to return to the main menu.
23. Select File/Save All.



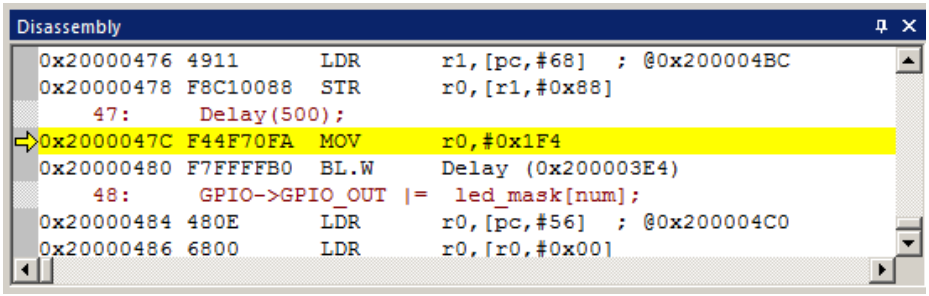
5) Select ram.ini to be executed when Debug mode is entered:

24. Select the Debug tab. Options for Target should still be open. Use if you closed it.
25. In the Initialization File: box, enter ram.ini as shown below. You can use the Browse icon.
26. ram.ini will be executed every time you enter Debug mode.
27. Unselect Load Application at Startup: This makes sure your program isn't loaded twice. Run to main() will be grayed out. See below: This function will be done by the last command in the ram.ini file.
28. Click on Utilities and uncheck Update Target before Debugging to turn off automatic Flash programming when entering Debug mode.
29. Click OK. Select File/Save All.



6) Compile and RUN the program:

30. Rebuild and enter Debug mode. Do not use Load. Load is only for programming Flash.
31. Note the disassembly window will show an address in the area of 0x2000_0000. See below:
32. This is the Blinky program located in RAM.
33. Breakpoints set will be software breakpoints and can be as many as you want. You cannot set them on-the-fly as with hardware breakpoints. You must stop the program to set/unset them.



TIP: There is a small cycle penalty for running in RAM. You are not likely to wear out the eNVM by debugging there.

4) TCP/IP: Keil http demonstration:

Keil has ported its TCP/IP stack to the SmartFusion Ethernet peripheral. This TCP/IP is part of the Keil RL-ARM software suite. Contact Keil sales for more information on RL-ARM.

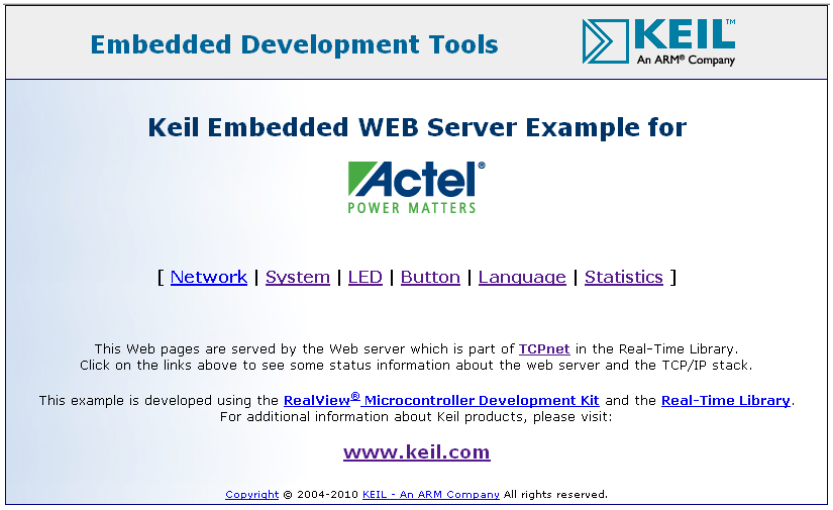
A web server (http) demonstration is available on: www.keil.com/download/docs/404.asp

Instructions are included as well as a pre-compiled sample executable. (Http_demo.axf).

Do not attempt to rebuild the project unless you have MDK-Pro installed and licensed. Http_demo.axf will be deleted and no new version will be created. You must then reload Http_demo.axf from the example files again.

You can load this axf file into the evaluation version of µVision, run and debug the http program.

A http demon will be installed on the SmartFusion and you can connect to it with your favourite browser as shown:



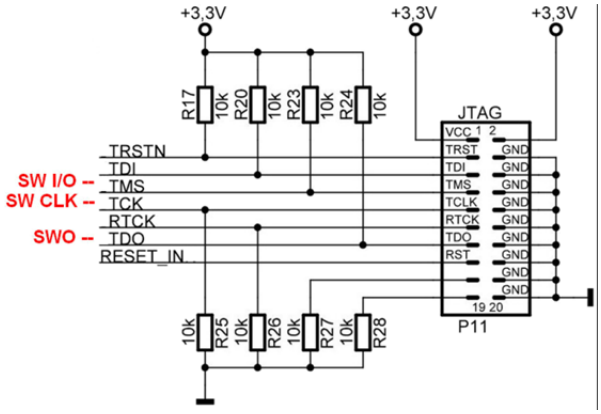
5) ARM JTAG/SWD/SWO Adapter Connector Schematics and the new ULINK-ME:

For the complete description of CoreSight and the new Hi-Density debug connector search www.arm.com for this document: DDI0314F_coresight_component_trm.pdf. Many customers are now using this new, smaller connector.

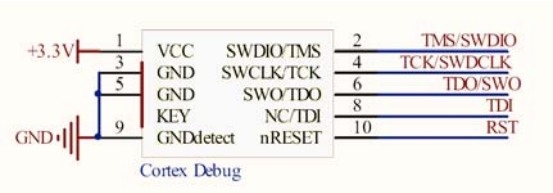
Hi-Density connector part numbers can be found here: www.samtec.com/ftppub/pdf/ftsh_mt.pdf.

The 10 pin part number that Keil uses in its boards is Samtec FTSH-105. A 20 pin is available: FTSH-110.

TIP: TRSTN resets the JTAG module inside the processor. Most better debug adapters do not need it.



Legacy JTAG Connector showing location of the Serial Wire signals. Top view.

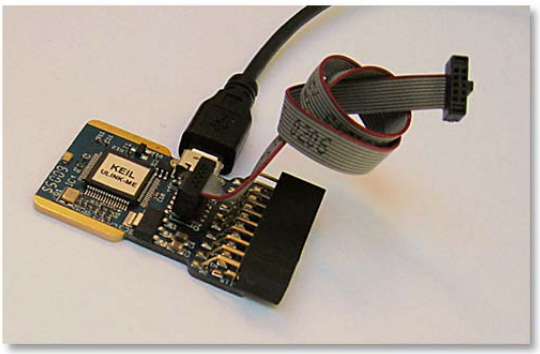


10 pin Hi-Density JTAG/SWD/SWO connector. Top view.

The new ULINK-ME:

The current ULINK-ME does not supply 3.3 volts to a target board through JTAG connector pin 1. Older versions of the ME did. You can remove this feature by cutting the pin 1 lead or remove the diode on the ME. This diode is on the bottom right corner of the ME beside the JTAG connector. This will not affect the EVAL-KIT. The DEV-KIT is affected more with its complicated power supplies. The ULINK2 or ULINK*pro* are not affected as neither provides power to a target.

The new ULINK-ME also incorporates a connector and cable as shown below for the new compact 10 pin JTAG/SWD connector. It does not supply the 3.3 volts to pin 1. It is shipping now as part of Keil or OEM boards only.



ULINK-ME with the new 10 pin connector and cable.

6) Acronym List

BP Break Point
DAP Debug Access Port
eNVM SmartFusion Flash – embedded nonvolatile memory
ETM Embedded Trace Macrocell
GPIO General Purpose Input/Output
ITM Instrumentation Trace Macrocell
SWD Serial Wire Debug
SWV Serial Wire Viewer
SWO Serial Wire Output
TPIU Trace Port Interface Unit
WP Watch Points. They are also called Access Breaks.

Macrocell: A macrocell is a pre-built "black box" of gates used by the chip designer to add a function or peripheral to a processor. Examples are ETM, ITM and memory. Macrocells can also be thought of as "building blocks". This is a term used by chip designers and not end users.

Hardware Design: This is Microsemi's terminology for the FPGA program. It is similar to a microcontroller project.

7) Overloading the SWO pin:

The Serial Wire Output pin (SWO) is just that – a single bit high speed line that is expected to handle all the data sent to it. A Cortex-M3 can read and write data much faster than the SWO can process.

SWV is a useful feature and is very versatile. It provides a unique view into your program while it is running. Because of the single pin, it is important to take steps to not overload it. µVision recovers gracefully from such overloads.

Overloads have an X in the Ovf or Dly (overflow or Delay) column in the Trace Records window: but not always. Sometimes the frames have corrupt, missing or have wrong information and this is easy to spot.

ITM frames with IDs other than 0 or 31 are a sure sign of either overloading or an incorrect Core Clock setting.

Here are some hints on reducing traffic to minimize SWO overruns:

1. Reduce the number of variables in the Logic Analyzer.
2. Reduce the number of items selected in the Trace Configuration window: These include:
    a. PC Sampling. Turn this off or increase the Prescaler to reduce the sampling rate.
    b. On Data R/W Sample.
    c. EXCTRC: Exception Tracing
    d. Any Trace Events. (the Counters)
    e. ITM Stimulus Ports 31 and 0. (Viewer and RTX Awareness respectively)
3. Set the Timestamp Prescaler to 1.
4. Turn the timestamps off. Sometimes this will turn the trace off though. But it is worth a try.
5. Lower the rate of reads, writes and exception calls in your software.
6. Turn off the SYSTICK timer in your software if you are not using it.
7. Reduce the Range setting in the Logic Analyzer. Sometimes a very large number overloads µVision.
8. The General Rule is: activate only those SWV features you need.
9. Using a faster computer can help in some cases. SWV can create a great deal of information that must be processed quickly. This is especially true where the Logic Analyzer window is concerned.
10. We will be making continuous improvements to µVision to provide you with a better SWV experience.

TIP: If you are seriously overloading the SWO, there will usually be at least some data coming out. µVision recovers easily and painlessly from data overruns. You are made aware when frames are lost.

8) Top Seven Reasons why you can't get SWV working:

Symptoms can range from an inability to enter a known variable in the Logic Analyzer, corrupted trace frames or no information at all.

1. Core Clock is wrong. This is the number 1 reason. The SmartFusion A2F500 has a higher SWO speed than the current 98 KHz. Normally, the SWO speed is derived from the Core Clock speed. For the next version of SmartFusion devices, you would enter 50 MHz or whatever your clock speed is in Core Clock: and the SWO speed will be calculated from that.

   With A2F200 silicon code date 0952 calculate Core Clock speed as (CPU Speed/1024). 100 MHz/1024 = 98 KHz. With A2F500 devices running at a CPU speed of 100 MHz, set Core Clock to 50 MHz. 75 MHz for J-Link.
2. Periodic Update is not enabled. Your windows update only when the processor is stopped.
3. Trace Enable is not checked.
4. Using JTAG instead of SWD. µVision will complain about this if you try it. SWV needs SWD (or SW) selected.
5. SWD speed too high. Try a slower speed. This is not usually the problem.
6. You have selected too many SWV items and it is extremely seriously overloaded. This effect often manifests itself when you stop the program and the Trace Records window is updated.
7. Certain portions of µVision in MDK are in the process of being updated. This is particularly true of the Logic Analyzer and ULINK*pro* features. With the current limited bandwidth of the SmartFusion SWO pin, a few Serial Wire Viewer features might not work consistently as described. These will be updated as soon as possible.

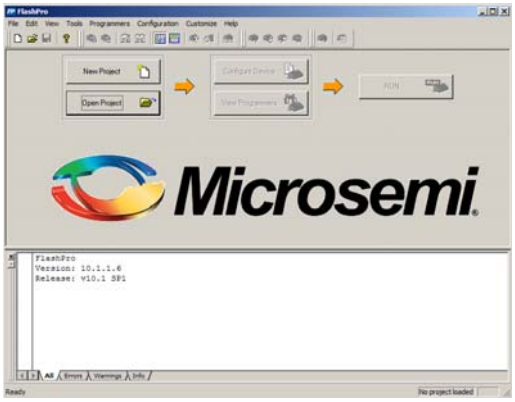
9) Programming the A2F-EVAL-KITBoard (F2 device) (FPGA eNVM) with STAPL

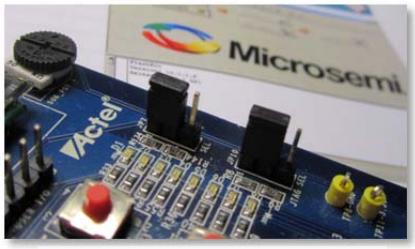
This is to program the FPGA eNVM (Flash) to connect SmartFusion to the board LEDs in order for the Keil examples to be able to blink the leds. The Keil example programs will still operate correctly.

A sample STAPL file, stp, mss_101.stp is provided with this document: www.keil.com/appnotes/docs/apnt_208.asp

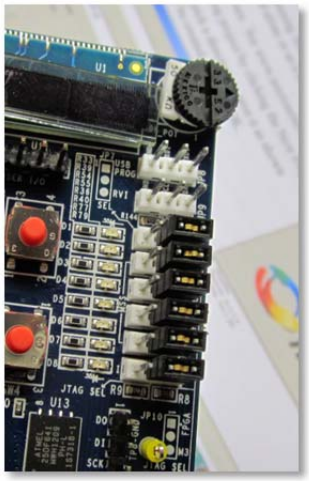
This stp file also programs the Cortex-M3 eNVM flash with a Microsemi example. Once you have programmed the STAPL file into the fabric, you can use µVision to program your executable Cortex-M3 code multiple times into the eNVM flash.

1. Ensure that jumpers JP7 and JP10 are populated in the 1-2 position and that the 2 USB cables are connected. See the two photos for jumper positions for the two EVAL-KIT boards:
2. Open the FlashPro Programming Software. FlashPro is available stand-alone or as part of the Libero IDE FPGA development software. Both are free from www.microsemi.com.
3. The EVAL-KITs have a built-in FlashPro hardware programmer.
4. Create a new programming project by clicking the New Project button:



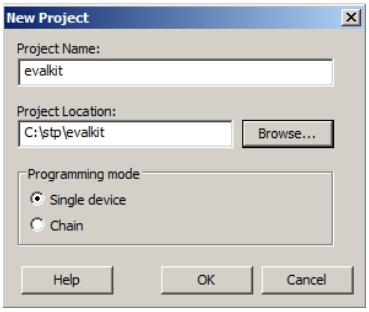


A2F-EVAL-KIT Jumpers

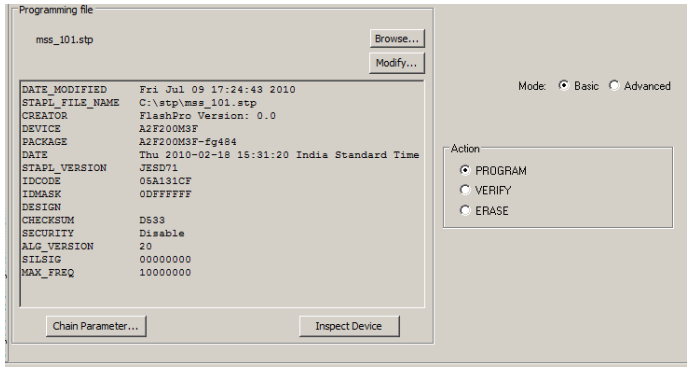


A2F-EVAL-KIT **REV2** Jumpers

5. Enter a new for this programming project. I chose evalkit and C:\stp as my directory.
6. Click on OK to close this widow.



7. Click on the "Configure Device" button. This will open the "Load Programming File" window.
8. Browse the PC file system to find the SmartFusion Eval Kit mss_101.stp file and enter it. It will display like here:



9. If the board is connected with two USB cables FlashPro will see the FlashPro3 hardware on the EVAL-KIT as shown here:

	Programmer Name	Programmer Type	Port	Programmer Status	Programmer Enabled
1	77857	FlashPro3	usb77857 (USB 2.0)		☑

10. Click the "Program" button to program the A2F-EVAL-KIT SmartFusion device.
11. If there is a warning about Vpump – you can ignore this message.
12. Programmer Status will display appropriate messages.
13. When FlashPro is completed programming this is a good result:

	Programmer Name	Programmer Type	Port	Programmer Status	Programmer Enabled
1	77857	FlashPro3	usb77857 (USB 2.0)	RUN PASSED	☑

14. Remove both USB cables **and return the jumpers to their original positions.**

Forgetting this step is a very common mistake !

15. If the jumpers are not properly set, µVision will not be able to access the SWD port. You will get errors.
16. Plug one USB cable into J14 to power the board. Or cycle the power on the DEV-KIT.

```
Rescanning for Programmers...
programmer '77857' : FlashPro3
Rescanning for Programmers DONE.
programmer '77857' : Scan Chain...
Warning: programmer '77857' : Vpump has been selected on programmer AND an externally provi
programmer '77857' : Scan Chain PASSED.
programmer '77857' : Executing action PROGRAM
programmer '77857' : EXPORT FSN[48] = 0114ab28b05c
programmer '77857' : Erase ...
programmer '77857' : Completed erase
programmer '77857' : Programming FPGA Array
programmer '77857' : Verifying FPGA Array
programmer '77857' :         Verifying FPGA Array -- pass
programmer '77857' : Program System Init and Boot Clients...
programmer '77857' : Program Embedded Flash Memory Module 0....
programmer '77857' : Verify System Init and Boot Clients...
programmer '77857' : Verify Embedded Flash Memory Module 0...
programmer '77857' : Finished: Thu Mar 21 17:44:45 2013 (Elapsed time 00:01:17)
programmer '77857' : Executing action PROGRAM PASSED.

                     o - o - o - o - o - o
```

All | Errors | Warnings | Info

Programming the A2F-DEV-KIT (Big board)

You must use a Microsemi LCPS to program the DEV-KIT. It does not have an on-board programming adapter.

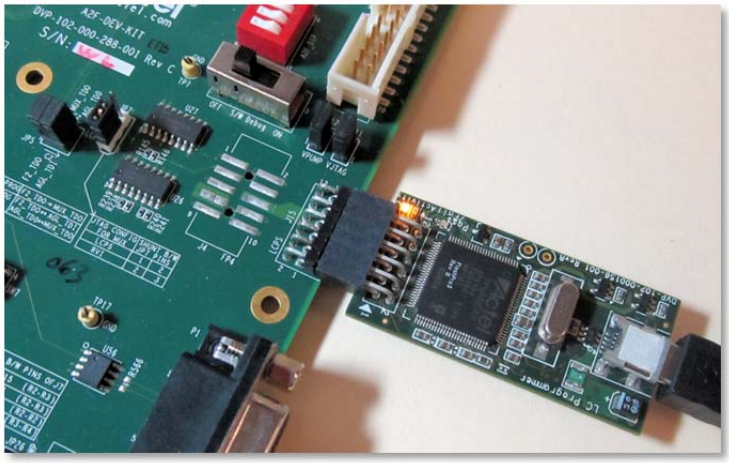
FlashPro software, the same as used with the FlashPro4, also works with the LCPS. The LCPS is shown below:

Refer to your board documentation for specific instructions and requirements.

A STAPL file that can be used with the Keil Blinky examples is here:
www.keil.com/appnotes/docs/apnt_208.asp

For FPGA Programming a STAPL (stp) file:

1. SW9 to S/W Debug OFF. (the left position)
2. JP7 set to 1-2-3 LCPS. See the photo:
3. JP6 must be in position 2-3 V1P5_EXT.
4. JP5 both jumpers to C1 – Single Prog.
5. Follow the instructions as for the EVAL-KIT.



10) Serial Wire Summary:

We have three basic debug systems as implemented in SmartFusion Cortex-M3 devices:

1. SWV and ITM data output on the SWO pin located on the standard JTAG debug connector.
2. ITM is a printf type viewer. ASCII characters are displayed in the Debug printf Viewer in µVision.
3. Memory Reads and Writes in/out the JTAG/SWD ports. The Memory and Watch windows use this technology.
4. Breakpoints and Watchpoints are set/unset through the JTAG/SWD ports.

These are all completely controlled through µVision via a ULINK.

11) Usefulness of the Trace:

SWV Trace adds significant power to debugging efforts. Problems which may take hours, days or even weeks in big projects can often be found in a fraction of these times with a trace. Especially useful is 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) or RTOS task switches.

Usually, these techniques allow finding bugs without stopping the program. These are the types of problems that can be found with a quality trace:

1) Pointer problems.
2) Illegal instructions and data aborts (such as misaligned writes).
3) Code overwrites – writes to Flash, unexpected writes to peripheral registers (SFRs)
4) A corrupted stack.
5) Out of bounds data. Uninitialized variables and arrays.
6) Stack overflows. What causes the stack to grow bigger than it should ?
7) Runaway programs: your program has gone off into the weeds and you need to know what instruction caused this. ***This is probably the most important use of trace. Needs ETM to be most useful.***
8) Communication protocol and timing issues. System timing problems.
9) Profile analysis and code coverage. Available only with ETM trace. SmartFusion2 has this feature.

12) What kind of data can the Serial Wire Viewer display ?

1) Global variables.
2) Static variables.
3) Structures.
4) Can see Peripheral registers – just read or write to them. The same is true for memory locations.
5) Can see executed instructions. SWV only samples them.
6) CPU counters. Folded instructions, extra cycles and interrupt overhead.

13) What Kind of Data the Serial Wire Viewer can't display…

7) Can't see local variables. (just make them global or static).
8) Can't see register to register operations. PC Samples records some of the instructions but not the data values.
9) SWV can't see DMA transfers. This is because by definition these transfers bypass the CPU and SWV can only see CPU actions.

14) Useful Documents:

1. **The Definitive Guide to the ARM Cortex-M3** by Joseph Yiu. (he also has one for the Cortex-M0) Search the web.
2. **MDK-ARM Compiler Optimizations: Appnote 202:** www.keil.com/appnotes/files/apnt202.pdf
3. **Lazy Stacking and Context Switching Appnote 298:** www.arm.com and search for Lazy Stacking.
4. **A list of resources is located at:** www.arm.com/products/processors/cortex-m/index.php
   Click on the Resources tab. Or search for "Cortex-M3" on www.arm.com and click on the Resources tab.
5. **ARM Infocenter: http://infocenter.arm.com** www.arm.com/cmsis forums.arm.com

15) Keil Products and Contact Information:

Keil Microcontroller Development Kit (MDK-ARM™)

- MDK-Lite (Evaluation version – 32K code/data limit) $0
- ***NEW !!*** MDK-ARM-CM™ (for Cortex-M series processors only – unlimited code limit) - $3,200
- MDK-Standard (unlimited compile and debug code and data size) - $4,895
- MDK-Professional (Includes Flash File, TCP/IP, CAN and USB driver libraries) $9,500

USB-JTAG/SWD 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* - $1,250 – Cortex-M*x* SWV & ETM trace.

- ***For special promotional or quantity pricing and offers, please contact Keil Sales.***

The Keil RTX RTOS is now provided under a Berkeley BSD type license. This makes it free.

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

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

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

All products are available from stock.

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

Call Keil Sales for special university pricing. Go to www.arm.com and search for university to view various programs and resources.

See the Keil Device Database® on www.keil.com/dd for the complete list of ARM supported devices. This information is also included in MDK in Target Options.

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

Prices are for reference only and are subject to change without notice.

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

For more information:

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

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

International Distributors: www.keil.com/distis/ See www.embeddedsoftwarestore.com

For more Microsemi specific information: please visit www.keil.com/microsemi

CMSIS information: www.arm.com/cmsis and forums.arm.com
