
Cortex™-M3 Hands-On LAB featuring Serial Wire Viewer

RealView® MDK Microcontroller Development Kit featuring Keil µVision®3

Luminary Evaluation Board with ULINK2® USB to JTAG Adapter

with the Luminary LM3S1968 Cortex™ -M3 Processor

Version 1.8 May, 2008



Table of Contents	Page
1. Introduction	2
2. Installation	2
3. Lab exercises introduction:	3
4. Graphics Example:	4
5. Finding Programming Errors	5
6. Finding the Definition and the Reference of a Variable	5
7. Changing text on the OLED Display	6
8. Watch Window <i>in Real-Time</i>	6
9. ITM Viewer: Instrumentation Trace Macrocell	7
10. PC Sample Example	7
11. Logic Analyzer <i>using Serial Wire Viewer</i>	8
12. Cortex-M3 Interrupts <i>using Serial Wire Viewer</i>	9
Appendix A: Configuring the Serial Wire Viewer (SWV):	10
Appendix B: Serial Wire Viewer: What is Real Time Trace good for ?	12

Contacts:

Robert Boys robert.boys@arm.com

Copyright © 2008 ARM, Ltd.

www.keil.com



Introduction:

1. The purpose of these hands-on exercises is to give you exposure to the RealView MDK[®] microprocessor development kit and enhance familiarity with the Cortex-M3 core. Emphasis is placed on demonstrating the Serial Wire Viewer (SWV) which is a key component of CoreSight[™] on-chip debug and trace technology.
2. The target processor is the Luminary LM3S1968 32 bit ARM Cortex-M3 processor. The Cortex-M3 is the latest ARM processor family and plugs directly into the embedded MCU market.
3. You will use Luminary example programs to configure μ Vision, compile and load these programs into the processor Flash memory. You will then put μ Vision into debug mode and you can start and stop the program and observe various data inside the processor. Most data will be viewed in real-time with the Serial Wire Viewer.
4. Where possible, we will use the Keil default settings. MDK is designed to offer “out of the box” operation. It takes a minimal effort to get projects configured, compiled, loaded and running. This saves much time. You do not need to write any new code.
5. This document requires the use of ARM[®] MDK Version 3.20 or higher. The IDE (Integrated Development Environment) used is Keil μ Vision[®]3. MDK uses the ARM RealView toolset including C, C++ compilers, libraries, assembler and linker and with the full version, the Keil RTX RTOS kernel is included.
6. Later versions of MDK will work but the example programs are subject to change as are μ Vision features. Earlier versions of MDK do not have the Serial Wire Viewer incorporated into μ Vision. Please contact R Boys or Keil technical support for the latest version of this document.

You can download MDK free from www.keil.com. Without a license key, μ Vision will run in evaluation mode. Code size is then restricted to 16 Kbytes. This is sufficient to compile and run the examples in this document.



Installation: (note: software might be pre-installed by Arrow engineers)

1. **Keil/ARM Software:**
When installing MDK version 3.20, please use the default directory c:\Keil.
2. **Luminary Software:**
Run the file PDL-LM3S-uvision-2293.exe to install the example programs. This file is available at http://www.luminarymicro.com/products/software_updates.html. Select the Keil update.
3. **Hardware:**
A Luminary LM3S1968 evaluation board and a ULINK2 or ULINK2-ME USB to JTAG adapter are required. Both the board and the ULINK are connected to the PC with a total of two (2) USB cables. The ULINK2 and ULINK-ME are electrically identical – the ME lacks a case, input protection devices and multiple connector sizes.
4. **Luminary USB:** Luminary provides an internal USB to JTAG adapter. This works with both JTAG and Serial Wire Debug modes of the Cortex-M3. However, Serial Wire Viewer is currently not supported. When connecting the board to the PC, select Cancel when the USB is detected by Windows. If the USB drivers are installed and the ULINK locks up – this is probably because somehow the Luminary USB is activated and controls the board and not the ULINK. Power cycle the board to release it.

5. All these examples will compile and link with the MDK evaluation version (no license is required). When the dialog box in Figure 1 appears when you enter debug mode, please click on OK. This box does not appear on a licensed copy of MDK.
6. Connect the hardware to your PC as shown in Figure 2. The USB cable connected to the LM3S1968 board is only to power it. All communication to and from the board is through the Keil ULINK[®]2 or ULINK-ME USB-JTAG adapter. The ME adapter has no JTAG cable and plugs directly into the Luminary board. A USB cable then plugs into your PC. This will be obvious how this is done.
7. There must be two (2) USB cables attached to your computer as shown in Figure 2.
8. When all the connections are correct, the LM3S1968 OLED displays some graphics or letters and a green LED will be on. On the ULINK2, only the red LED will be illuminated. If the red Hibernate LED is illuminated on the Luminary board – please press the RESET button or cycle the power to the board by unplugging its USB cable.
9. When you start μ Vision, if the ULINK2 needs its firmware upgraded – this will happen at this time. It will notify you of this fact and will only take a minute or so and then you can continue with the exercises.
10. Please send any corrections and comments to robert.boys@arm.com.

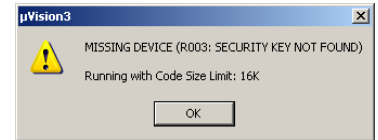




Figure 1 Evaluation Mode



Figure 2 Hardware Connections: LM3S1968 and ULINK2

Lab exercises introduction: General Information needed to run the exercises.

1. If μ Vision locks up, temporarily disconnect the USB cable to the ULINK2 and the connection will be automatically restored. Control will then be restored to μ Vision. You might also have to cycle the Luminary board.
2. Edit and Debug mode. μ Vision has two modes: Edit and Debug. Clicking on the debug icon  toggles between these modes. When this icon has a square box around it – μ Vision is in debug mode.
 - a. **Edit mode** is essentially for configuring much of μ Vision, creating and modifying source files and compiling them (rebuild) and programming the processor Flash memory.
 - b. **Debug mode** is for running and stopping the program and viewing registers and memory locations of various types. The RUN and STOP icons are available only in debug mode. The core registers are available only during debug mode. You will quickly learn to know which mode μ Vision is currently in.






If you have problems check Appendix A first. The fastest method to get to configuration menus is to first make sure μ Vision is in Edit mode (not debugging) and click on the Options for Target icon. 

Many of these exercises will run with minor modifications on other Luminary Cortex-M3 evaluation boards.

1) Graphics:

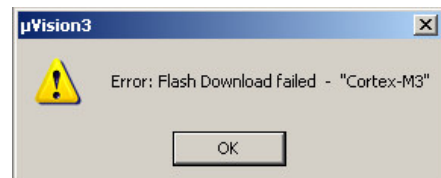
This example displays a graphic on the Luminary OLED screen. You will learn how to open a project, compile (rebuild) the source files into an executable file *.AXF, load this into Flash and run it.

It is a good idea to use this exercise to get acquainted with μ Vision and the Luminary evaluation board.


1. Start μ Vision by clicking on its icon.
2. Select Project/Open Project. Open project C:\Keil\ARM\Boards\Luminary\ek-lm3s1968\Graphics\graphics.Uv2.
3. Configure μ Vision as described in Appendix A.
4. Everything else about this project is pre-configured but you must first compile it with the Rebuild command.
5. Click on Rebuild All Target Files. 
6. This will return 0 Error(s) and 0 Warning(s) in the Build window at bottom left of your screen.
7. Now, the FLASH in the processor must be programmed with the executable file you just created.
8. Click on the FLASH load icon. . μ Vision will go through its erase, program and verify cycles.
9. If you get a JTAG Communication Error, repeat step 8 in Appendix A. (SWV might be reset back to JTAG). See below under **Problems**.
10. Put μ Vision into debug mode. Click on this  icon. **Missing Device:** - click on OK.
11. Run the program by clicking on the Run  icon.
12. Some graphics will appear on the screen and some sound will be emitted. This is correct. Congratulations ! Your first μ Vision project !
13. Stop the program execution by clicking on the stop icon. 

Problems:

1. It is possible the Luminary USB drivers are in control of the board rather than the ULINK. Cycle the power for both the ULINK and board. This problem can be avoided by not installing (or removing) the FTDI USB drivers.
2. **If you get this or a similar error from a failed attempt to Flash or Reload the processor** – the USB drivers might be in control of the processor – recycle the power on the board to release it.
3. If the FLASH programming continues to fail – remove and reinstall the FLASH programming algorithms again as described in Appendix A.
4. Make sure there is a USB cable attached to the ULINK2 or ULINK-ME and another to the Luminary board and then both are connected to your PC. The Luminary USB cable supplies only power 5 volts to the board.



Finding Programming Errors

1. μ Vision must be out of the debugging mode. Click on this icon if necessary to enter Edit mode. 
2. Insert a programming error into a file. As an example, I added two underscores `__` behind the variable `ulRow` on line 651 of `graphics.c`. (click on `graphics.c` in the Project Workspace in upper left hand corner in necessary).
3. Rebuild the project. You will get Figure 3.
4. Note the target file was not created because of the error described in the Output window in the line 651 `graphics.c` (this line is outlined in Figure 3). Click on this line and μ Vision takes you to the error in the source file.
5. A turquoise arrow will display the offending line in order for you to easily locate and repair it. This is shown in Figure 3 in the file `graphics.c`. Note: On some computers this arrow will not show but you are still taken to the line.
6. Correct the error and rebuild.
7. **Neat Feature:** Click once on any curly bracket or parenthesis and note that it turns grey and so does its counterpart. This helps you keep track of which braces are related to each other.

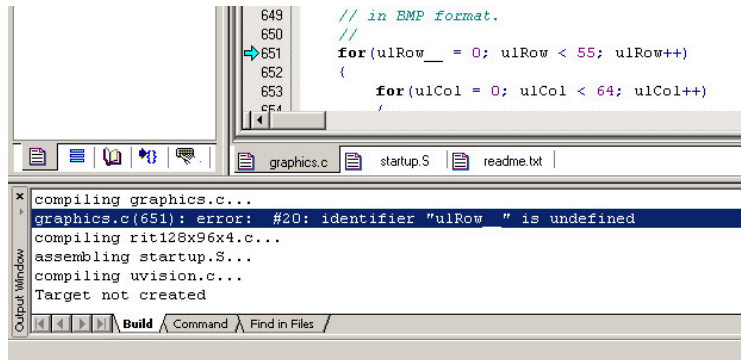


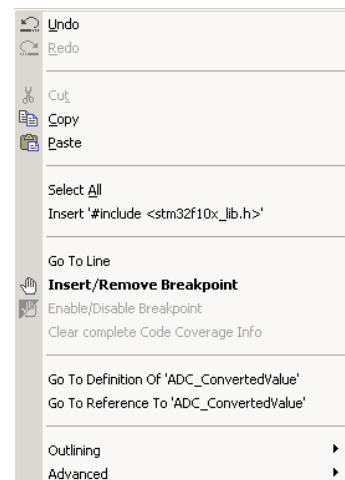
Figure 3 Error messages

Finding the Definition and the Reference of a Variable





Neat Feature: Sometimes it can be quite difficult to find the definition and reference of a variable. μ Vision makes this very easy. To find the definition or reference of a variable: right click on it and select either definition or reference in the window that opens as shown in Figure 4. You will be taken to this location.

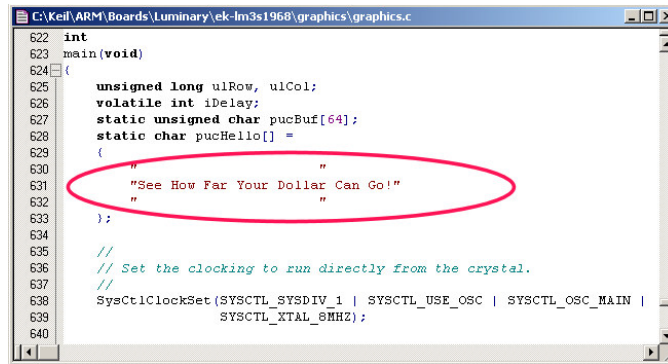
1. In `graphics.c` as shown in Figure 3, right click on the variable `ulRow`.
2. Select Definition and view the result(s) in the Browse window.
3. Click once on each entry of `ulRow` and see where they are referenced.
4. Doubleclick on an entry for `ulRow` in the Browse window go to this place in the source file.
5. It would have taken some time to find where some variables are defined without this Neat Feature.

Figure 4 Finding Definitions and References of variables.



Changing text on the OLED Display

1. In the file graphics.c scroll to around line number 631 as shown in Figure 5.
2. This is the ASCII string "See How Far Your Dollar Can Go! "
3. You can open this file if not already visible by double-clicking on it in the Project Workspace in the Source directory. This is in the upper left hand corner of μ Vision.
4. Enter your own ASCII text inside the quotes.
5. Click on Rebuild  and then program the processor flash memory. 
6. Enter the debug mode  and click on Run. 
7. Your new text will now be visible on the LCD.
8. Stop program execution and take μ Vision out of debug mode to be ready for the next exercise.



```

622 int
623 main(void)
624 {
625     unsigned long ulRow, ulCol;
626     volatile int iDelay;
627     static unsigned char pucBuf[64];
628     static char pucHello[] =
629     {
630         "
631         "See How Far Your Dollar Can Go!"
632         "
633     };
634
635     //
636     // Set the clocking to run directly from the crystal.
637     //
638     SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC | SYSCTL_OSC_MAIN |
639                   SYSCTL_XTAL_0MHz);
640

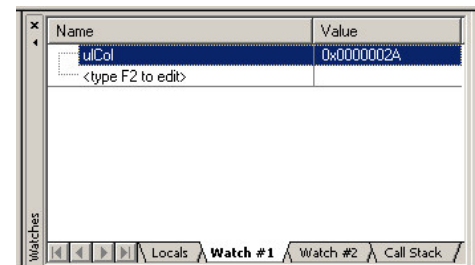
```

Figure 5 Changing the Text on the LCD

Watch Window in Real-Time

1. Make sure μ Vision is in Edit mode.
2. In file graphics.c find line 625 `unsigned long ulRow, ulCol;`
3. This is the definition of these two local variables. We will make them static so the Serial Wire Viewer can see them.
4. Add static so this line reads `static unsigned long ulRow, ulCol;`
5. Rebuild the project and reload to the Luminary board.
6. Enter debug mode and click on Run. You can configure the Watch window while the program is running.
7. If the Watch window shown in Figure 6 is not open in the bottom area of your screen, open it with View/Watch & Call Stack Window. Click on the Watch #1 tab.
8. Click once on "type F2 to edit" and press F2. Enter `ulCol`
9. This will display the values of the static variable ulCol. This is the column selected on the OLED display.
10. If ??????? is displayed instead of numbers – either stop and start the program or enter the variable in the fully qualified format as shown here: `\graphics\main\ulCol`.
11. **Neat Feature:** Carefully click once or twice on the Value until it is blocked over.
12. Enter a new value (0 will work) and press Enter. This new value will be injected into this memory location in real-time. The pattern moving across the screen will be momentarily shifted as the new value of ulCol is accepted.
13. Stop program execution and take μ Vision out of debug mode.

Figure 6 Watch window



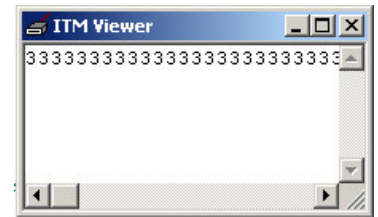
How It Works: μ Vision uses the Serial Wire Debug Port (as opposed to the JTAG port) to read or write memory locations without stealing any CPU cycles. This is nearly always non-intrusive and therefore does not impact the program execution timings. The only time this will be intrusive is in the unlikely event the program running on the CPU and μ Vision read or write to the same memory location at exactly the same. Then the CPU will be stalled for only one clock cycle. In practice, this cycle stealing effectively never happens.

ITM Viewer: Instrumentation Trace Macrocell

Neat Feature: The ITM Viewer window is written to by customer code and uses the ITM channel Port 0. This is really easy to do. In this example, it writes an ASCII value to the ITM Viewer window. This is called serial debug or printf debugging. In the code below, writing the value ch will take 1 CPU cycles. No cycles are used to transfer the data through the ITM out to μ Vision.

```
#define ITM_Port8(n)    (*(volatile unsigned char *) (0xE0000000+4*n))
ITM_Port8(0) = ch;    //data sent to  $\mu$ Vision
```

1. Add these two lines just inside the main function:
#define ITM_Port8(n) (*(volatile unsigned char *) (0xE0000000+4*n))
char test = 0x33;
2. Near the end of graphics.c add this line: ITM_Port8(0) = test; (I did before if (ulCol > 53 near line 693.)
3. Rebuild, Reload and enter debug mode.
4. Open the ITM Viewer window by clicking on View/Serial Window/ITM Viewer.
5. Run the program and the ITM Viewer will fill up with multiple 3s as shown here.
6. 0x33 is the ASCII value for the number 3.
7. You can easily format the data better by adding appropriate CR and LF chars.
8. The ITM Viewer uses the Serial Wire Debug (SW-DB) of the Cortex-M3 to get the data outside the processor in real-time.
9. These writes to the ITM will also be displayed in the Trace Records window.



PC Sample Example

Super Neat Feature: The PC Sample is an output of the Serial Wire Viewer (SWV) and is useful for Performance or Profile Analysis (where does your program spend most of its time) and determining the PC (program counter) when your program has “gone into the weeds” and is caught in an infinite loop. The SWV is a very good tool for these purposes.

- 1) Click on STOP to halt program execution. Leave μ Vision in Debug mode.
- 2) Open Peripherals/Target Settings and select the Trace tab to open the Cortex Target Driver Setup.
- 3) Configure it as shown below in Figure 7. Make sure the Periodic in the PC Sampling area is checked. This turns on the PC Sampling feature. Ignore the ITM fields. Uncheck EXTRC.
- 4) Click on OK. Click Yes if asked to take new values.

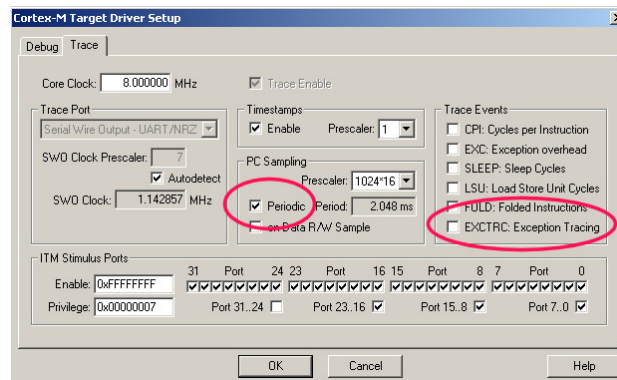


Figure 7 Trace Configuration to Sample the PC

- 5) Make sure the Trace Records window is open. Select Peripherals/Trace/Records if it is not.
- 6) Right click in the Trace Records window and ensure PC Samples is checked.
- 7) Double click in the Trace Records window to clear it if necessary, even when the program is running.
- 8) Click on RUN. The Trace Records window will immediately fill up with PC values as shown in Figure 8.

How it works: Clearly the PC is in a tight loop around addresses 184H to 188H and sometimes executes instructions near 754H. This example is easy to find without a trace but if the loop was bigger and more complicated the track of the PC will be captured allowing detailed analysis. If the program is stopped due to a breakpoint, the instruction(s) that caused the problem will also be captured. Complicated “off into the weeds” problems are nearly impossible to find without a trace.

- 9) Stop program execution and open the Disassembly window in the View menu item. Note the program usually stops in the loop at 184H to 188H. Note not all PC values can be show due to the high speed of the processor.
- 10) If you still have the ITM Viewer working from the last example – these will be displayed here also as shown.
- 11) Note you can filter in or out types of records by right-clicking on the Trace Records window.

Type	Qvf	Num	Address	Data	PC	Dly	Cycles	Time[s]
PC Sample					00000186H		12459394709	1557.42433863
PC Sample					00000186H		12459411093	1557.42638662
PC Sample					00000186H		12459427477	1557.42843462
PC Sample					00000186H		12459443861	1557.43048263
PC Sample					00000186H		12459460245	1557.43253063
PC Sample					00000186H		12459476629	1557.43457863
PC Sample					00000186H		12459493013	1557.43662662
PC Sample					00000186H		12459509397	1557.43867463
ITM		0		33H			12459522430	1557.44030375
PC Sample					00000758H		12459525781	1557.44072263
PC Sample					0000075CH		12459542165	1557.44277062
PC Sample					00000332H		12459558549	1557.44481862
PC Sample					00000758H		12459574933	1557.44686663
PC Sample					00000184H		12459591317	1557.44891463
PC Sample					00000184H		12459607701	1557.45096262
PC Sample					00000184H		12459624085	1557.45301062
PC Sample					00000184H		12459640469	1557.45505863
PC Sample					00000184H		12459656853	1557.45710663
PC Sample					00000184H		12459673237	1557.45915462
PC Sample					00000184H		12459689621	1557.46120262

Figure 8 Trace Records showing PC Sampling with Timestamps

Logic Analyzer (LA)

We will now graph the static variable ulCol in real-time using the Serial Wire Viewer. We changed this to a static variable in the Watch window exercise. The Serial Wire Viewer is unable to see local variables because its scope keeps changing.

1. μ Vision must be in Debug mode and stopped. μ Vision must be configured as in Appendix A.
2. Open the Trace configuration window and deselect PC Sample. Click on OK. Close the Trace Records window.
3. Open the Logic Analyzer window by View/Logic Analyzer Window.
4. Click on Setup... the Setup Logic Analyzer window in Figure 9 opens up.
5. Click in the area under Current Logic Analyzer Signals and press the PC **Insert** key.
6. Enter **ulCol** and press Enter key. Error ? See Step 7. If not, set MAX to 0xFF and MIN to 0x0.
7. If you get an error – this means variable ulCol is out of scope. Either close this window, run the program to bring it into scope and try to reenter ulCol or enter the fully qualified variable as shown in Figure 9. \graphics\main\ulCol
8. Click on CLOSE. Click on **OUT** to set the range to 20 seconds.
9. Click on Run.
10. Put the scroll bar to the right side if needed.
11. The Logic Analyzer as shown in Figure 10 will display the changing values of ulCol in real-time. No CPU cycles are stolen to do this. In addition, any read or writes to ulCol will also be sent to the Trace Records window.

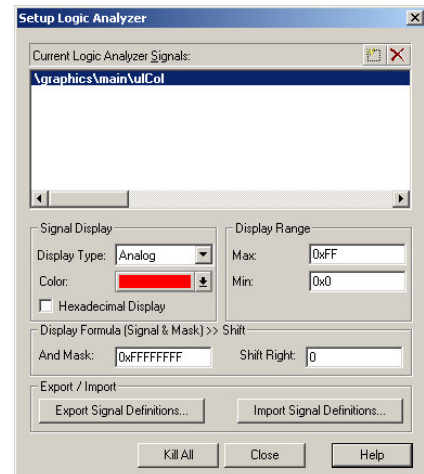


Figure 9 LA Configuration

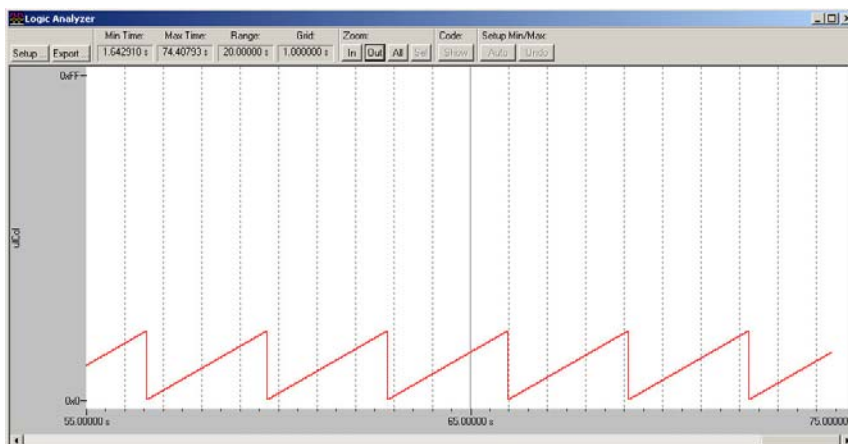


Figure 10 Logic Analyzer Window

Logic Analyzer (LA) continued...the Data Reads and Writes in Trace Records...

1. You can do the next steps while the program is still running.
2. Open the Trace Records window by selecting Peripherals/Trace/Records.
3. Figure 11 opens up. Clear this window by double-clicking on it and it will fill up again.

Notes:

1. There are ITM, Data Read and Data Write values as shown in the Trace Records window.
2. 20000004H is the physical address of the variable ulCol. This information is available in the Symbol table.
3. The data written in the Data column is increasing as it is in the Logic Analyzer.
4. If you select *on Data R/W Sample* in the Trace configuration menu, the address of the instruction is added.
5. If the Watch window is still configured with ulCol, it will also be updating at this time.
6. Note the data value of 0x33 from the previous ITM example.
7. You can right click on this window and filter record types in and out.

Type	Ovf	Num	Address	Data	PC	Dly	Cycles	Time[s]
ITM		0		33H			595677370	74.45967125
Data Read			20000004H	0000026H		X	595677651	74.45970637
Data Write	X		20000004H	0000027H		X	595677651	74.45970637
ITM		0		33H			596141828	74.51772850
Data Read			20000004H	0000027H		X	596142115	74.51776437
Data Write	X		20000004H	0000028H		X	596142115	74.51776437
ITM		0		33H			596606286	74.57578575
Data Read			20000004H	0000028H		X	596606572	74.57582150
Data Write	X		20000004H	0000029H		X	596606572	74.57582150
ITM		0		33H			597070724	74.63381050
Data Read			20000004H	0000029H		X	597071008	74.63387600
Data Write	X		20000004H	000002AH		X	597071008	74.63387600
ITM		0		33H			597535146	74.69189325
Data Read			20000004H	000002AH		X	597535430	74.69192875
Data Write	X		20000004H	000002BH		X	597535430	74.69192875
ITM		0		33H			597999555	74.74994437
Data Read			20000004H	000002BH		X	597999838	74.74997975
Data Write	X		20000004H	000002CH		X	597999838	74.74997975
ITM		0		33H			598463951	74.80799387
Data Read			20000004H	000002CH		X	598464232	74.80802900

Figure 11 Trace Records window showing Data Reads and Writes to ulCol

1. Stop program execution.
2. Scroll to the last entry in the Trace Records window.
3. The last Data Write will have the same value as the Watch window and the LA window.

Interrupts

We can also view interrupts in the Trace Records and Interrupt Trace windows.

1. If necessary, stop execution and exit debug mode.
2. Open the project c:\Keil\ARMboards\ek-lm3s1968\interrupts\interrupts.Uv2
3. Configure μ Vision exactly as in Appendix A.
4. Rebuild, Reload the project and enter Debug mode.
5. Open the Trace Records and Exception Trace windows in the Peripherals/Trace menu. Position as appropriate.
6. Run the program. You can clear either window by right clicking in it.
7. Note the exceptions fill in both windows with timestamps, counts and various other times in Figures 12 and 13.
8. This is the end of the exercises.

Type	Ovf	Num	Address	Data	PC	Dly	Cycles	Time[s]
Exception Entry		18	413047443				51.63093037	
Exception Exit		18	428998991				53.62482387	
Exception Entry		17	428998995				53.62482438	
Exception Exit		17	444998993				55.62482362	
Exception Entry		16	444998993				55.62482413	
Exception Exit		16	460998984				57.62482300	
Exception Return		0	460998992				57.62482400	
Exception Entry		18	477047344				59.63091800	
Exception Exit		18	492998989				61.62482362	
Exception Entry		17	492998993				61.62482413	
Exception Exit		17	508998987				63.62482337	
Exception Entry		16	508998991				63.62482387	
Exception Exit		16	524998601				65.62482513	
Exception Return		0	524998601				65.62482513	
Exception Entry		18	541047330				67.63091625	
Exception Entry		17	541067331				67.63341637	
Exception Entry		16	541097352				67.63591900	
Exception Exit		16	556998983				69.62482288	
Exception Return		17	556998991				69.62482388	
Exception Exit		17	572998991				71.62482388	

Figure 12 Interrupts listed by occurrence.


Num	Name	Count	Total Time	Min Time In	Max Time In	Min Time Out	Max Time Out	First Time [s]	Last Time [s]
2	NMI	0	0 s						
3	HardFault	0	0 s						
4	MemManage	0	0 s						
5	BusFault	0	0 s						
6	UsageFault	0	0 s						
11	SVCALL	0	0 s						
12	DbgMon	0	0 s						
14	PendSV	0	0 s						
15	SysTick	0	0 s						
16	ExitRQ 0	3	5.989 s	1.989 s	2.000 s	2.011 s	6.000 s	55.62482413	67.63591900
17	ExitRQ 1	3	6.003 s	2.000 s	2.003 s	4.009 s	6.000 s	53.62482438	67.63341637
18	ExitRQ 2	3	5.990 s	1.994 s	2.002 s	6.006 s	6.006 s	51.63093037	67.63091625
19	ExitRQ 3	0	0 s						
20	ExitRQ 4	0	0 s						
21	ExitRQ 5	0	0 s						
22	ExitRQ 6	0	0 s						
23	ExitRQ 7	0	0 s						


Figure 13 Interrupts listed by number.

Appendix A: Configuring the Serial Wire Viewer (SWV):

The configuration of the SWV needs to be done for each project you use. It will be saved when you close a project or close μ Vision. There are four parts to this setup:

- 1) Select the Serial Wire Debug Port instead of the JTAG port (Figure 15).
- 2) Configure the Serial Wire Output (Figure 14).
- 3) Set the Trace. (Figure 16)
- 4) Configure the FLASH programmer. (Figure 17)

μ Vision must not be in debugging mode for these steps. If it is, click on the debugger icon.  The menus below will be grayed out if μ Vision is in debugger mode.

- 1) The correct project should be selected. (Project/Select Project).
- 2) Open the Debug setup menu by clicking on Options For Target icon on the main μ Vision toolbar: 
- 3) The Target tab will be selected. Set the CPU speed to 8 Mhz.
- 4) Select the Debug tab.
- 5) Select the ULINK Cortex Debugger.

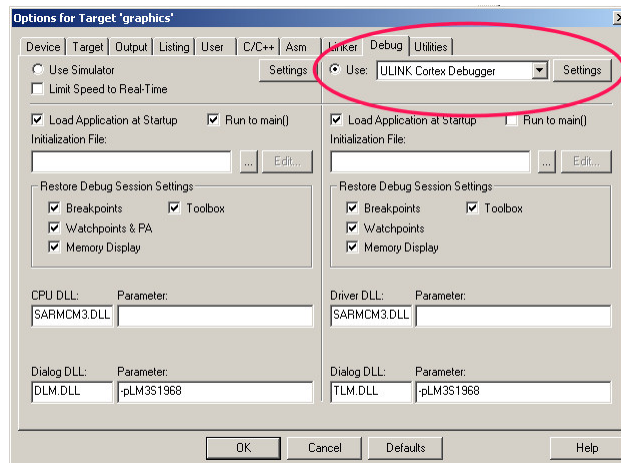


Figure 14

- 6) Click on Settings to configure the ULINK2. Figure 15 below opens up.

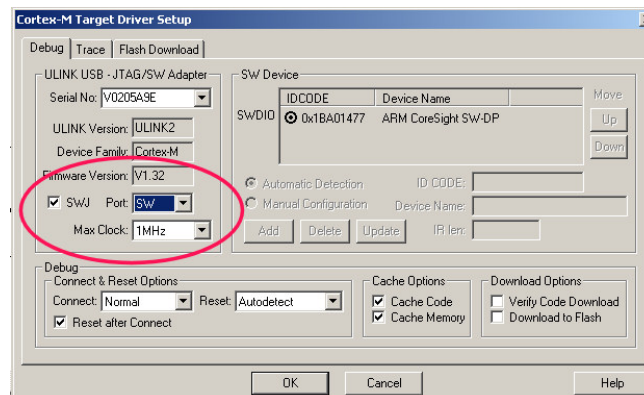


Figure 15

- 7) Select the SWJ box and set Port: to SW as shown. This must not be set to JTAG. SWV operates only through the Serial Wire debug port (SW). Max clock @ 1 MHz is correct. Up to 10 MHz will work. A Device Name similar to that shown in the SW Device box must be shown. If not, make sure both the ULINK and the board are powered.
- 8) Select the Trace tab and Figure 16 opens up to configure the SWV trace:

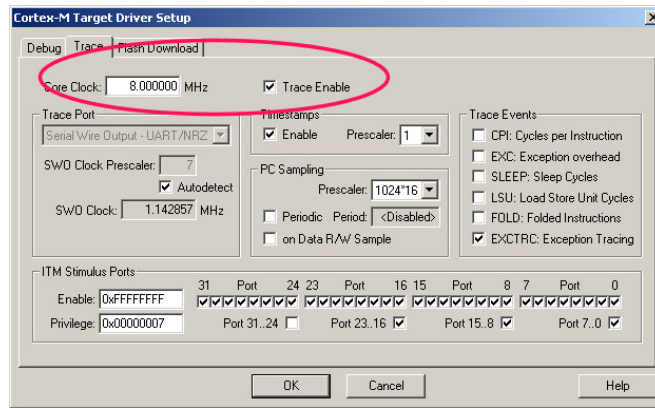




Figure 16

9) **Important Step:** Set the Core Clock to 8 MHz and check the Trace Enable box. Ensure in the ITM Stimulus Ports that at least Port 0 and Port 7..0 are selected. The rest are Don't Care for this exercise.

Note: if the Trace Enable box is grayed out – this means you are in Debug mode. You are not allowed to enable/disable Trace in debug mode. Close this window, leave debug mode (click on the debug icon ) and click on the Target Options icon  to enter the correct window. Now you will be able to set the Trace Enable box.

- 10) Select the Flash Download tab. Figure 17 opens up.
- 11) Click on the entry shown and remove it with the Remove box.
- 12) Click on OK to leave this window. Click again on Settings (again) and select Flash Download (again).
- 13) Click on Add and reselect the algorithm LM3Sxxx 256kB Flash file. Click on Add and it will be entered.
- 14) Click on OK twice. The Serial Wire Viewer Trace is now configured and ready to use !
- 15) Click on File/Save to save these settings and return to the last example you left from.

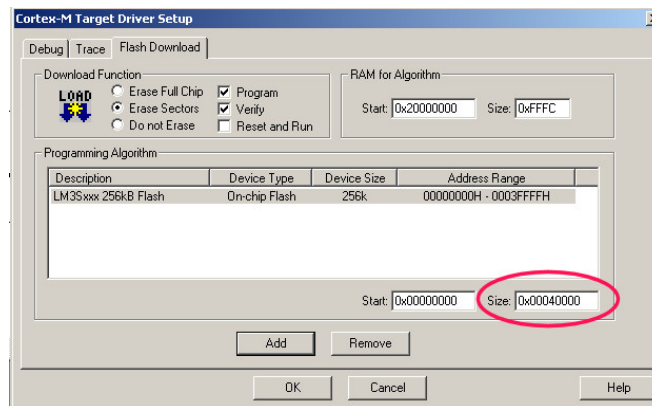


Figure 17

Trouble ?

- If you have trouble getting or maintaining communication with your evaluation board – please check these settings. Pay particular attention to the JTAG or SW setting.
- Make sure the ULINK and Luminary board are both properly connected to a USB connector on your PC.
- If these windows are not correctly configured, especially with those items circled – the Serial Wire Viewer will not function. **No trace data will be displayed. If the Core Frequency is incorrect, the data will be sporadic and erroneous or totally missing.**
- If you are unable to Stop program execution or μ Vision gets locked up – try disconnecting temporarily the ULINK2 USB cable and/or the Luminary board. Usually this restores communication in a controlled fashion.

Appendix B: Serial Wire Viewer: What is Trace good for ?

The ARM Cortex-M3 core has a new low cost version of trace called SWV and is accessed through the same JTAG connector via the ULINK2. SWV is a superset of regular debugging via the JTAG module. No special trace hardware is required. The standard Keil ULINK2 will perform both the JTAG and SWV functions at top speed by plugging into the standard JTAG connector on your target board. There is no extra cost to support SWV !

The SWV trace has these features:

- 1) Data tracing. Read/write with instruction address and timestamps.
- 2) Hardware Breakpoints and Watchpoints.
- 3) ETM trigger (if the microcontroller is so equipped).
- 4) Program Counter or Data address sampler registers (Four 32 bit registers). (not all these can be sampled).
- 5) Instrumentation Trace: Through the ITM and is also called printf debugging.

In addition, the SWV trace module has counters for:

1. # of CPU clock cycles (32 bits).
2. # of folded instructions count (8 bits).
3. # of instruction cycles (8 bit).
4. # of cycles processor is sleeping (8 bit).
5. Total cycles spent in interrupt processing (8 bit).
6. Total cycles spent in load/store operations.

Each time one of these counters rolls over an event is generated and fed out to μ Vision. These counters can be used for various events such as measuring total elapsed time and number of events during a time period.

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). 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), corrupted stack.
- 4) Out of bounds data. Uninitialized variables and arrays.
- 5) Stack overflows. What causes the stack to grow bigger than it should ?
- 6) Runaway programs: your program has gone off into the weeds and you need to know what instruction caused this.
- 7) Communication protocol and timing issues. System timing problems.
- 8) Profile analyzer.

What kinds of data can the Serial Wire Viewer see ?

1. Global variables.
2. Static variables.
3. Structures.
4. Peripheral registers – just read or write to them. Same is true for memory locations.
5. Can't see local variables. (just make them global or static)
6. Can't see DMA transfers. (because by definition these bypass the CPU and SWV can only see CPU actions)