

TN0132 Technical note

STM32 Serial Wire Viewer and ETM capabilities with EWARM 5.40 and MDK-ARM 3.70

Introduction

This document presents Serial Wire Viewer (SWV) and Embedded Trace Macrocell (ETM) capabilities with these toolchains in various configurations:

- RVMDK 3.70 (RealView ® Microcontroller Development Kit from Keil[™])
- EWARM 5.40 (Embedded Workbench® for ARM® from IAR Systems)

The STM32 provides a 4-bit ETM port as well as a Serial Wire Viewer port.

In general, the ETM is used to find problems using heavy duty trace debugging such as looking for difficult bugs, while the SWV is used to provide a low cost method of obtaining information from inside the MCU using ARM CoreSight[™] technology.

1 SWV feature capabilities

1.1 Introduction

Serial Wire Viewer is the ability of the ARM[™] core to send real-time trace information out via a single wire port called the Serial Wire Output (SWO). The trace information is in several familiar formats such as:

- Instrumentation Trace Macrocell (ITM) for application driven trace source that supports printf style debugging.
- Data Watchpoint and Trace (DWT) for variable monitoring and PC-sampling, which can in turn be used to periodically output the PC (sampled) or various CPU internal counters and to obtain profiling information from the target:
 - Program Counter sampling.
 - Data read and write cycles.
 - Variable and peripheral values.
 - Event counters.
 - Exception entry and return.
- Timestamps and CPU cycles are emitted relative to packets.

Our focus is on the analysis of the serial wire port's output information, in particular configurations, and to highlight its capabilities of providing information and data at the high speed that the STM32 runs at.

1.2 Context

This manual comes with a *.zip* file containing the subdirectories and files that make up the core of application examples.

These application examples are configured at 72 MHz (maximum frequency of the STM32 MCU) and highlight the following SWV features:

- Data access
- Interrupt
- Program counter sampling
- Printf

Each application example's folder contains:

- inc subfolder containing the example header files
- src subfolder containing the example source files
- project subfolder containing two projects that compile the example files:
 - EWARMv5 containing the project for the EWARM toolchain
 - ARM-MDK containing the project for the ARM-MDK toolchain

These examples are tested in the following hardware and software conditions:

- SW/HW toolchain: EWARM 5.40/JLINK v6 and ARM-MDK 3.70/ULINK2
- Target board: STM3210E-Eval Rev.A
- Office PC Pentium® 4 CPU 3.20 GHz, 504 MB of RAM, SP2
- SW clock autodetected



1.3 Program counter (PC) sample

The display of program counter values is useful for program flow change, profile analysis and determining where the CPU might be caught in an infinite loop. Profile analysis gives a helpful indication where the CPU is spending its time.

Nevertheless in some conditions, the SWO and toochains are not capable of providing every program counter value because of the high speed that the STM32 runs at. Section 1.3.1 and Section 1.3.2 illustrate the limitations detected with both ARM-MDK and EWARM toolchains with particular configurations.

1.3.1 ARM-MDK / ULINK2 toolchain example

Running the PC Sampling example using the ARM-MDK toolchain highlights that if the PC sampling prescaler is equal to 7*1024 (10044 samples per second), a hardware buffer overrun occurs. This is due to the fact that the USB cannot accept data at the speed ULINK2 is sending it (see *Figure 1*).

Type Ovf Num	Address	Data	PC	Dly	Cycles	Time[s]	
PC Sample			0800028CH		6299	0.00008749	-
PC Sample			0800028AH		12443	0.00017282	
PC Sample			0800028EH		18587	0.00025815	
PC Sample			0800028EH		24731	0.00034349	
PC Sample			0800028AH		30875	0.00042882	
PC Sample			0800028EH		37019	0.00051415	
PC Sample			0800028EH		43163	0.00059949	
PC Sample			0800028CH		49307	0.00068482	
PC Sample			0800028AH		55451	0.00077015	
PC Sample			0800028EH		61595	0.00085549	
PC Sample			0800028EH		67739	0.00094082	
PC Sample			0800028AH		73883	0.00102615	
PC Sample			0800028CH		80027	0.00111149	
PC Sample			0800028AH		86171	0.00119682	
PC Sample			0800028EH		92315	0.00128215	
PC Sample			0800028EH		98459	0.00136749	
PC Sample			0800028AH		104603	0.00145282	
PC Sample			0800028EH		110747	0.00153815	
PC Sample			0800028EH		116891	0.00162349	
PC Sample			0800028CH		123035	0.00170882	

Figure 1. ARM-MDK PC samples: Hardware buffer overrun

By decreasing the prescaler value until 3*1024 (23437 samples per second), an overflow occurs due to the fact that the SWO communication channel is not fast enough to handle that much data. Consequently, an overflow is detected as illustrated by *Figure 2*.



Туре	Ovf	Num	Address	Data	PC	Dly	Cycles	Time[s]	
ITM		23		05C21701H			1401	0.00001946	
PC Sample	(X)				080003BEH	×	871553	0.01210490	
PC Sample	X				080003BEH	×	871553	0.01210490	
PC Sample					080003C2H	×	871553	0.01210490	
PC Sample	X				080003C2H	×	871553	0.01210490	
PC Sample	X				080003C2H	×	871553	0.01210490	
PC Sample					080003C2H	×	871553	0.01210490	
PC Sample	X				080003C2H	×	871553	0.01210490	
PC Sample	X				080003C2H	×	871553	0.01210490	
PC Sample	X				080003C2H	×	871553	0.01210490	
PC Sample					080003C0H	Х	871553	0.01210490	
PC Sample	X				080003C2H	×	871553	0.01210490	
PC Sample	X				080003C2H	×	871553	0.01210490	
PC Sample	X				080003C2H	Х	871553	0.01210490	
PC Sample					080003C0H	×	871553	0.01210490	
PC Sample	X				080003C2H	Х	871553	0.01210490	
PC Sample	X				080003C2H	X	871553	0.01210490	
PC Sample	×				080003C2H	X	871553	0.01210490	
PC Sample					080003BEH	X	871553	0.01210490	
PC Sample					080003C0H	×	871553	0.01210490	

Figure 2. ARM-MDK PC samples: Overflow

1.3.2 EWARM / J-Link toolchain example

Running the PC Sampling example using the EWARM toolchain highlights that when the Rate (the number of samples per second) is set to 86538, the SWO communication channel is not fast enough to handle that much data. Consequently, an Overflow is detected, illustrated by *Figure 3*.

θXB	्र 🥎 🖬 🕼						
Index	SW0 Packet	Cycles	Event	Value	Trace		
					while(Tim	ingDelay != 0);	
					??Delay_0:		
014413	178A030008	97911680	PC	0x0800038A	LDR	R1, [R0]	
014414	70	97911680	OVERFLOW				
					while(Tim	ingDelay != 0);	
					??Delay_0:		
014415	178Å030008	97911680	PC	0x0800038A	LDR	R1, [R0]	
014416	178E030008	97911680	PC	0x0800038E	BNE	??Delay_0	
014417	70	97911680	OVERFLOW				
014418	178C030008	97911680	PC	0x0800038C	CMP	R1, #0x0	
014419	178E030008	97911680	PC	0x0800038E	BNE	??Delay_0	
					while(Tim	ingDelay != 0);	
					??Delay_0:		5
014420	1782030008	97911680	PC	0~08000384	TDP	R1 [R0]	~
<							>

1.4 Read and write data frames

Read and write data frames can be displayed giving the address of the responsible instruction, the data value transferred, the data address and timestamps in both core cycles and seconds. *Figure 4* shows a series of data reads and writes showing these attributes.



1.4.1 ARM-MDK / ULINK2 toolchain example

Running the Data Access example using the ARM-MDK toolchain highlights that if a delay of less than 310 μ s is inserted before incrementing the j variable, a hardware buffer overrun occurs. This is due to the fact that the USB cannot accept data access at the speed the ULINK2 is sending it (see *Figure 4*).

By decreasing the delay until 155 μ s, an overflow occurs because the SWO communication channel is not fast enough to handle that much data access, for example, some values are not displayed in the trace record window (see *Figure 5*). To avoid these two problems in the Data Access example, the user should insert a delay greater than 310 μ s.

Туре	Ovf	Num	Address	Data	PC	Dly	Cycles	Time[s]	
Data Read			20000004H	0F000F07H	08000678H				
Exception ???		0							
Data Read			20000004H	00000F6EH					
Data Write			20000004H	00000F6FH	08000670H				
Data Read			20000004H	00000F6FH	08000678H				
Data Write			20000004H	00000F70H	08000670H				
Data Read			20000004H	00000F70H	08000678H				
Data Write			20000004H	00000F71H	08000670H				
Data Read			20000004H	00000F71H	08000678H				
Data Write			20000004H	00000F72H	08000670H				
Data Read			20000004H	00000F72H	08000678H				9
Data Write			20000004H	00000F73H	08000670H				
Data Read			20000004H	00000F73H	08000678H				
Data Write			20000004H	00000F74H	08000670H				
Data Read			20000004H	00000F74H	08000678H				
Data Write			20000004H	00000F75H	08000670H				
Data Read			20000004H	00000F75H	08000678H				
Data Write			20000004H	00000F76H	08000670H				
Data Read			20000004H	00000F76H	08000678H				
Data Write			20000004H	00000F77H	08000670H				

Figure 4. ARM-MDK data read access: Hardware buffer overrun

Figure 5.	ARM-MDK data read access: Overflow
i igui o oi	

Туре	Ovf	Num	Address	Data	PC	Dly	Cycles	Time[s]	
	01		Address		10		1410		
ITM		U		00H				0.00001958	
Data Read			20000008H	00000001H			1410	0.00001958	
Data Write	\cap		20000008H	00000002H		×	3937	0.00005468	
Data Write	X		20000008H	00000003H		X	11997	0.00016663	
Data Read			20000008H	00000003H		×	20677	0.00028718	
Data Write			20000008H	00000004H		X	20677	0.00028718	
Data Write	X		20000008H	00000005H		X	26877	0.00037329	
Data Read			20000008H	00000005H		X	35557	0.00049385	
Data Write			20000008H	00000006H		X	35557	0.00049385	
Data Write	X		20000008H	00000007H		×	41757	0.00057996	
Data Read			20000008H	00000007H		X	50437	0.00070051	
Data Write			20000008H	00000008H		x	50437	0.00070051	
Data Write	X		20000008H	00000009H		Ŷ	56637	0.00078663	
Data Write	- Ŷ		20000008H	0000000AH		ŝ	62217	0.00086412	
Data Read			20000008H	0000000AH		Ŷ	70897	0.00098468	
Data Write			20000008H	0000000BH		Ŷ	70897	0.00098468	
	- U					Ŷ			
Data Write	X		20000008H	0000000CH			78957	0.00109662	
Data Write	X		20000008H	0000000DH		X	84537	0.00117413	
Data Write	X		20000008H	0000000EH		Х	93217	0.00129468	
Data Write			20000008H	0000000FH		X	98797	0.00137218	



1.4.2 EWARM / J-Link toolchain example

Similar behavior is detected with the EWARM toolchain. In fact, to be able to perform a read access followed by a write access on all j values in the Data Log window, a delay of about 38 µs must be inserted before incrementing j. If not, only the first and last values are detected (see *Figure 6*).

Figure 6. E	WARM data read	access : Overflow
-------------	----------------	-------------------

1ts	☑ Data Log @ j [Read/Write] [0×20000408 - 0×2001	0040B]		
Breakpoints					
Bre	Debug Log Build Breakpoints Fi	nd In SWO Trace			
×	Time	Program Counter	j	Address	
	0.000us	0x08000176	R 0x0000000	@ 0x20000408+?	
	0.000us	0x0800017A	-	@ 0x20000408+?	
	0.000us				
		0x0800017A	-	@ 0x20000408+?	
		Overflow	(W. O. 000000000)	@ 0 00000400.0	
	0.000us 0.000us	 Overflow	(W 0x0000028)	@ 0x20000408+?	
	0.000us 0.000us	Overflow			
	0.000us				
	0.000us				
alog	0.000us	Overflow			
	Data Log Data Log Summary				
Rea	ady				



1.5 Printf

The Printf software example writes some data to a specific ITM address and CoreSight automatically sends this data to the SWO port. This data can be displayed on the **Serial Wire Viewer** window. This method is marginally intrusive to the user program and referred to as printf "debugging".

1.5.1 ARM-MDK / ULINK2 toolchain example

When running the example using the ARM-MDK toolchain, an overload is detected if we send simultaneously 10 * 202 through the ITM port 0. Data is skipped from iterations 5 to 9 (see *Figure 7*). This is not an SWO related limitation but is due to the fact that the USB cannot accept data access at the speed the ULINK2 is sending it.

Figure 7. ARM-MDK Printf: Hardware buffer overrun

 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789
 123456789

1.5.2 EWARM / J-Link toolchain example

With the EWARM toolchain, all values sent via the SWO are captured by EWARM (no overload is detected). The **Trace record** window can display the latest fifty lines of submitted data (see *Figure 8*). The user can consult all submitted data stored in the log file.

Figure 8. EWARM Printf: Data display

123456789 123456



1.6 Exception trace dialog

The interrupt example aims to determine the number of times that the interrupt was entered using 2 different methods:

- 1. Using the interrupt window: the SWV captures Systick exceptions' return and exit. These are timestamped and the exception number is then displayed.
- 2. Using a variable Tick incremented in the Systick interrupt handler.

1.6.1 ARM-MDK / ULINK2 toolchain example

After running the example using the ARM-MDK toolchain, it is easy to see that the Tick variable's value and the Systick exception value in the output window are similar only when the interrupt periodicity is greater than or equal to 430 μ s (see *Figure 9*). Otherwise some interrupts are missed due to the hardware buffer overrun. By decreasing the interrupt periodicity value until 160 μ s, an Overflow is also detected (see *Figure 10*). To avoid these two problems in the Exception example, the user in this case should insert a delay of greater than 430 μ s.

igui	<u>00. A</u>		К Ехсер				, i u i i			
Num	Name	Count	Total Time	Min Time In	Max Time In	Min Time Out	Max Time Out	First Time [s]	Last Time [s]	^
3	HardFault	0	242.403 us							
4	MemManage	0	0 s							
5	BusFault	0	0 s							1
6	UsageFault	0	0 s							_
11	SVCall	0	0 s							
12	DbgMon	0	0 s							
14	PendSV	0	0 s							
15	SysTick	21268	18.639 ms	55.556 ns	142.222 us	0 s	1.333 s	0.00024240	4.69187292	
16	ExtIRQ 0	0	0 s							
17	ExtIRQ 1	0	0 s							
18	ExtIRQ 2	0	0 s							
19	ExtIRQ 3	0	0 s							
20	ExtIRQ 4	0	0 s							
21	ExtIRQ 5	0	0 s							
22	ExtIRQ 6	0	0 s							
23	ExtIRQ 7	0	0 s							~
<					100					
eady				Trace: HW B	uffer Overrun)	ULII	VK Cortex Debugge	er t1: 6.367	516

Figure 9. ARM-MDK Exception: Hardware buffer overrun

Figure 10. ARM-MDK Exception: Overflow

			-						
Num	Name	Count	Total Time	Min Time In	Max Time In	Min Time Out	Max Time Out	First Time [s]	Las
2	NMI	0	Os						
3	HardFault	0	0 s						
4	MemManage	0	0 s						
5	BusFault	0	0 s						
6	UsageFault		Os						
	SVCall	-							
	DbgMon	0							
		0							
		(4479)		416.667 ns	416.667 ns	501.917 us	502.028 us	0.00058815	2.25
		0 \							
	•								
	•								
23	ExtIRQ 7	0	\ 0s						
			<u> </u>						
			1						
_									
((() () () () () () () () ()	Tick,0x0A		(4509)						
I	type F2 to edit>								
	2 3 4 5 6 11 12 14 15 16 17 18 19 20 21 22 23	2 NMI 3 HardFault 4 MemManage 5 BusFault 6 UsageFault 11 SVCall 12 DbgMon 14 PendSV 15 SysTick 16 ExtIRQ 0 17 ExtIRQ 2 19 ExtIRQ 3 20 ExtIRQ 4 21 ExtIRQ 5 22 ExtIRQ 6	2 NMI 0 3 HardFault 0 4 MemManage 0 5 BusFault 0 6 UsageFault 0 11 SVCall 0 12 DbgMon 0 14 PendSV 0 15 SysTick 4479 16 ExtIRQ<0 0 17 ExtIRQ<1 0 18 ExtIRQ<2 0 20 ExtIRQ<3 0 20 ExtIRQ<5 0 22 ExtIRQ 7 0	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 4479 1.866 ms 16 ExtIRQ 0 0 0 s 17 ExtIRQ 1 0 0 s 18 ExtIRQ 2 0 0 s 20 ExtIRQ 3 0 0 s 21 ExtIRQ 5 0 0 s 22 ExtIRQ 7 0 0 s 23 ExtIRQ 7 0 0 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 4479 1.866 ms 416.667 ns 16 ExtIRQ 0 0 0 s 1 17 ExtIRQ 1 0 0 s 1 18 ExtIRQ 2 0 0 s 2 20 ExtIRQ 3 0 0 s 2 21 ExtIRQ 5 0 0 s 2 22 ExtIRQ 7 0 0 s 2	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 4479 1.866 ms 416.667 ns 16 ExtIRQ 0 0 0 s 1 17 ExtIRQ 1 0 0 s 1 18 ExtIRQ 2 0 0 s 2 20 ExtIRQ 3 0 0 s 2 21 ExtIRQ 5 0 0 s 2 22 ExtIRQ 7 0 0 s 2 23 ExtIRQ 7 0 0 s 2	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 (4479) 1.866 ms 416.667 ns 416.667 ns 501.917 us 16 ExtIRQ 0 0 0 s 1 17 ExtIRQ 1 0 0 s 17 ExtIRQ 2 0 0 s 1 18 ExtIRQ 3 0 0 s 20 ExtIRQ 4 0 0 s 2 22 ExtIRQ 7 0 0 s 22 ExtIRQ 7 0 0 s 2 5 3 23 ExtIRQ 7 0 0 s 2 5 5	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 4479 1.866 ms 416.667 ns 501.917 us 502.028 us 16 ExtIRQ 0 0 0 s 1 502.028 us 502.028 us 16 ExtIRQ 1 0 0 s 1 502.028 us 502.028 us 17 ExtIRQ 1 0 0 s 1 1 502.028 us 20 ExtIRQ 3 0 0 s 2 2 ExtIRQ 4 0 0 s 21 ExtIRQ 5 0 0 s 2 2 ExtIRQ 7 0 0 s 22 ExtIRQ 7 0 <th>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 13 HArdFault 0 0 s 14 PendSV 0 0 s 15 SysTick 4479 1.866 ms 416.667 ns 501.917 us 502.028 us 0.00058815 16 ExtIRQ 0 0 0 s 1 1 502.028 us 0.00058815 16 ExtIRQ 1 0 0 s 1 1 1 0.00058815 16 ExtIRQ 1 0 0 s 1 1 1 1 0.00058815 17 ExtIRQ 1 0 0 s 2 2 1 1 1 0 0 s 20 ExtIRQ 5 0 0 s 2 2 2 ExtIRQ 7 0 0 s</th>	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 13 HArdFault 0 0 s 14 PendSV 0 0 s 15 SysTick 4479 1.866 ms 416.667 ns 501.917 us 502.028 us 0.00058815 16 ExtIRQ 0 0 0 s 1 1 502.028 us 0.00058815 16 ExtIRQ 1 0 0 s 1 1 1 0.00058815 16 ExtIRQ 1 0 0 s 1 1 1 1 0.00058815 17 ExtIRQ 1 0 0 s 2 2 1 1 1 0 0 s 20 ExtIRQ 5 0 0 s 2 2 2 ExtIRQ 7 0 0 s



1.6.2 EWARM / J-Link toolchain example

Similar behavior is detected with the EWARM toolchain. In fact, a delay of 17.7 μ s should be inserted to guarantee that the SysTick exception value displayed in the **Exception trace** window is the same as the Tick variable. If this condition is not verified, some exceptions are missed, as illustrated by *Figure 11*.

SysTick 2422 211161.111us 618970.6 0.472us 339.97 Approximative time count: 4840 Overflow count: 3386 Overflow count: 3386 Interrupt Log Summary Interrupt Log Summary Interrupt Log Interrupt Log Summary Interrupt Log Summary Interrupt Log Summary X Image: Summary Image: Summary Image: Summary Image: Summary X Image: Summary Image: Summary Image: Summary Image: Summary Index SWO Packet Cycles Event Value Trace 012587 0E0030 59768217 Return to Exception 0 012588 F0BF70 59782616 Packet and Timestamp 14399 012589 0E0F10 59782616 Packet and Timestamp 14399 012590 70 59782616 OVERFLOW 0 012591 0E0030 59782616 Return to Exception 0		Expression	i Value	Location	Туре					
× Interrupt Count First Time Total Time Fastest Slo SysTick 2422 211161.111us 618970.6 0.472us 339.97 Approximative time count: 4840 Overflow count: 3386 0 0 0 0 Interrupt Log Interrupt Log Summary		Tick	(3387)	0×20000408	vu32					
× Interrupt Count First Time Total Time Fastest Slo SysTick 2422 211161.111us 618970.6 0.472us 339.97 Approximative time count 4840 Overflow count 3386 0 0 9 Overflow count 3386 Interrupt Log Summary										
× Interrupt Count First Time Total Time Fastest Slo SysTick 2422 211161.111us 618970.6 0.472us 339.97 Approximative time count 4840 Overflow count 3386 0 0 9 Interrupt Log Interrupt Log Summary Interrupt Log Summary Interrupt Log Summary Interrupt Log Summary × ● ● ● ● ● Index SWO Packet Cycles Event Value Trace 012587 0E0030 59768217 Return to Exception 0 0 012588 F0BF70 59782616 Packet and Timestamp 14399 0 012589 0E0F10 59782616 Enter Exception Number 15 0 0 012590 70 59782616 Return to Exception 0 0 0 012591 0E0030 59782616 Return to Exception 0 0										
× Interrupt Count First Time Total Time Fastest Slo SysTick 2422 211161.111us 618970.6 0.472us 339.97 Approximative time count: 4840 Overflow count: 3386 0 0 0 0 Interrupt Log Interrupt Log Summary										
Interrupt Count First Time Total Time Fastest Slo SysTick 2422 211161.111us 618970.6 0.472us 339.97 Approximative time count: 4840 Overflow count: 3386 Overflow count: 3386 0 0 Interrupt Log Interrupt Log Summary Interrupt Log Summary Interrupt Log Summary Interrupt Log Summary × Index SWO Packet Cycles Event Value Trace 012587 0E0030 59768217 Return to Exception 0 0 012588 F0BF70 59782616 Packet and Timestamp 14399 012589 0E0F10 59782616 Enter Exception Number 15 012591 0E0030 59782616 Return to Exception 0 012591 0E0030 59782616 Return to Exception 0										
Interrupt Count First Time Total Time Fastest Slo SysTick 2422 211161.111us 618970.6 0.472us 339.97 Approximative time count: 4840 Overflow count: 3386 Overflow count: 3386 0 0 Interrupt Log Interrupt Log Summary Interrupt Log Summary Interrupt Log Summary Interrupt Log Summary × Index SWO Packet Cycles Event Value Trace 012587 0E0030 59768217 Return to Exception 0 0 012588 F0BF70 59782616 Packet and Timestamp 14399 012589 0E0F10 59782616 Enter Exception Number 15 012591 0E0030 59782616 Return to Exception 0 012591 0E0030 59782616 Return to Exception 0										
Interrupt Count First Time Total Time Fastest Slo SysTick 2422 211161.111us 618970.6 0.472us 339.97 Approximative time count: 4840 Overflow count: 3386 Overflow count: 3386 0 0 Interrupt Log Interrupt Log Summary Interrupt Log Summary Interrupt Log Summary Interrupt Log Summary × Index SWO Packet Cycles Event Value Trace 012587 0E0030 59768217 Return to Exception 0 0 012588 F0BF70 59782616 Packet and Timestamp 14399 012589 0E0F10 59782616 Enter Exception Number 15 012591 0E0030 59782616 Return to Exception 0 012591 0E0030 59782616 Return to Exception 0	멅									
Interrupt Count First Time Total Time Fastest Slo SysTick 2422 211161.111us 618970.6 0.472us 339.97 Approximative time count: 4840 Overflow count: 3386 Overflow count: 3386 0 0 Interrupt Log Interrupt Log Summary Interrupt Log Summary Interrupt Log Summary Interrupt Log Summary × Index SWO Packet Cycles Event Value Trace 012587 0E0030 59768217 Return to Exception 0 0 012588 F0BF70 59782616 Packet and Timestamp 14399 012589 0E0F10 59782616 Enter Exception Number 15 012591 0E0030 59782616 Return to Exception 0 012591 0E0030 59782616 Return to Exception 0	Š.									
Interrupt Count First Time Total Time Pastest Sto Approximative time count 4840 Overflow count 3386 0.472us 339.97 Interrupt Log Interrupt Log Summary	Š,	Debug Log Bi	uild Live Watch Liv	e Watch						
SysTick 2422 211161.111us 618970.6 0.472us 339.97 Approximative time count: 4840 Overflow count: 3386 0 0 0 0 Interrupt Log Interrupt Log Summary Interrupt Log Summary Interrupt Log Summary 1	×	Interrupt			Count	F	First Time	Total Time	Fastest	Slowes
Noterflow count: 3386 Interrupt Log Interrupt Log Summary X					(2422)	21116	1.111us	618970.6	0.472us	339.972us
Noterflow count: 3386 Interrupt Log Interrupt Log Summary X										
Interrupt Log Interrupt Log Summary × •				0						
× O ➤ ► Index SWO Packet Cycles Event Value Trace 012587 0E0030 59768217 Return to Exception 0 012588 F0BF70 59782616 Packet and Timestamp 14399 012589 0E0F10 59782616 Enter Exception Number 15 012590 70 59782616 OVERFLOW 0 012591 0E0030 59782616 Return to Exception 0	nar	Overflow co	ount: 3386							
× O ➤ ► Index SWO Packet Cycles Event Value Trace 012587 0E0030 59768217 Return to Exception 0 012588 F0BF70 59782616 Packet and Timestamp 14399 012589 0E0F10 59782616 Enter Exception Number 15 012590 70 59782616 OVERFLOW 0 012591 0E0030 59782616 Return to Exception 0	١.									
× O ➤ ► Index SWO Packet Cycles Event Value Trace 012587 0E0030 59768217 Return to Exception 0 012588 F0BF70 59782616 Packet and Timestamp 14399 012589 0E0F10 59782616 Enter Exception Number 15 012590 70 59782616 OVERFLOW 0 012591 0E0030 59782616 Return to Exception 0	Sec.									
× O ➤ ► Index SWO Packet Cycles Event Value Trace 012587 0E0030 59768217 Return to Exception 0 012588 F0BF70 59782616 Packet and Timestamp 14399 012589 0E0F10 59782616 Enter Exception Number 15 012590 70 59782616 OVERFLOW 0 012591 0E0030 59782616 Return to Exception 0	- ăl									
× O ➤ ► Index SWO Packet Cycles Event Value Trace 012587 0E0030 59768217 Return to Exception 0 012588 F0BF70 59782616 Packet and Timestamp 14399 012589 0E0F10 59782616 Enter Exception Number 15 012590 70 59782616 OVERFLOW 0 012591 0E0030 59782616 Return to Exception 0	pt Lo									
Index SWO Packet Cycles Event Value Trace 012587 0E0030 59768217 Return to Exception 0 012588 F0BF70 59782616 Packet and Timestamp 14399 012589 0E0F10 59782616 Enter Exception Number 15 012590 70 59782616 OVERFLOW 0 012591 0E0030 59782616 Return to Exception 0	errupt Log									
012587 0E0030 59768217 Return to Exception 0 012588 F0BF70 59782616 Packet and Timestamp 14399 012589 0E0F10 59782616 Enter Exception Number 15 012590 70 59782616 OVERFLOW 012591 0E0030 59782616 Return to Exception 0	Interrupt Log	Interrupt Log	Interrupt Log Summar	у						
012588 F0BF70 59782616 Packet and Timestamp 14399 012589 0E0F10 59782616 Enter Exception Number 15 012590 70 59782616 OVERFLOW 012591 0E0030 59782616 Return to Exception 0				Y						
012589 0E0F10 59782616 Enter Exception Number 15 012590 70 59782616 OVERFLOW 012591 0E0030 59782616 Return to Exception 0	×	ტ×ি	9 🍾 🔛 8=		Event	Valu	16	Trace		
012590 70 59782616 OVERFLOW 012591 0E0030 59782616 Return to Exception 0	×	O X ⊡ Index	୍ତ୍ତ 🍾 🔚 🕅 SWO Packet 🛛 (Cycles			91	Trace		
012591 0E0030 59782616 Return to Exception 0	×		SWO Packet (0E0030 5		Return to Except	ion 0		Trace		
	×		SWO Packet 0 0E0030 5 F0BF70 5	Cycles	Return to Except Packet and Times	ion 0 stamp 143		Trace		
012502 E0EE70 E070701E Backet and Timestann 14200	×		SWO Packet (0E0030 5 F0BF70 5 0E0F10 5	Cycles	Return to Except Packet and Times Enter Exception OVERFLOW	ion 0 stamp 143 Number 15		Trace		
	×		SWO Packet 0 0E0030 9 F0BF70 9 0E0F10 9 70 9	Cycles	Return to Except Packet and Times Enter Exception OVERFLOW	ion 0 stamp 143 Number 15		Trace		
C012593 C0FF887A 61797014 Timestamp synch 1999999	ace		SWO Packet 0 0E0030 9 F0BF70 9 0E0F10 9 70 9 0E0030 9	Cycles 59768217 59782616 59782616 59782616 59782616 59782616	Return to Except Packet and Times Enter Exception OVERFLOW	ion 0 stamp 143 Number 15 sion 0 stamp 143	99	Trace		
	ace		SWO Packet 0 0E0030 9 F0BF70 9 0E0F10 9 70 9 0E0030 9 F0BF70 9 70 9 0E0030 9 F0BF70 9 0E0030 9 F0BF70 9	Cycles 59768217 59782616 59782616 59782616 59782616 59782616 59797015	Return to Except Packet and Times Enter Exception OVERFLOW Return to Except Packet and Times	ion 0 stamp 143 Number 15 sion 0 stamp 143	99	Trace		
Ready	O Trace		SWO Packet 0 0E0030 9 F0BF70 9 0E0F10 9 70 9 0E0030 9 F0BF70 9 70 9 0E0030 9 F0BF70 9 0E0030 9 F0BF70 9	Cycles 59768217 59782616 59782616 59782616 59782616 59782616 59797015	Return to Except Packet and Times Enter Exception OVERFLOW Return to Except Packet and Times	ion 0 stamp 143 Number 15 sion 0 stamp 143	99			

Figure 11. EWARM Exception: Overflow



2 ETM feature capabilities

2.1 Introduction

An embedded trace macrocell (ETM) is a real-time trace module providing program flow tracing.

For the STM32, the ETM unit provides a high bandwidth instruction trace over a dedicated 4-bit high-speed trace bus using a special hardware probe such as an IAR J-Trace for a Cortex-M3 or Signum JTAGjet-Trace.

This section describes the ETM features implemented by:

- ARM-MDK/JTAGjet-Trace
- EWARM/J-Trace CM3

The focus is on its ability to provide program flow information at the high speed that the STM32 runs at.

2.2 Context

This user guide comes with a zip file containing the subdirectories and files that makes up the core of application examples.

These application examples are configured at 72 MHz (maximum frequency of the STM32 MCU) and highlight the following trace features:

- Instruction timing
- Data tracing
- Function profiler

Each application example's folder contains:

- *inc* subfolder containing the example header files
- src subfolder containing the example source files
- project subfolder containing two projects that compile the example files:
 - EWARMv5 containing the project for the EWARM toolchain
 - ARM-MDK containing the project for the ARM-MDK toolchain

These examples are tested in the following hardware and software conditions:

- SW/HW toolchain: EWARM 5.40/J-Trace CM3 and ARM-MDK 3.70/JTAGjet-Trace
- Target board: STM3210E-Eval Rev.A
- Office PC Pentium® 4 CPU 3.20 GHz, 504 MB of RAM, SP2

These examples use the following ETM options:

- Stall processor on FIFO full
- Trace buffer size: 0x400000
- Trace port mode: Normal, half-rate clocking.



2.3 Instruction timing

The ETM allows reconstruction of program execution which is useful for debugging and especially for detecting rare bugs source.

J-Trace CM3 and JTAGjet-Trace have a 4 MB buffer for trace data storage. Since for Cortex-M3 one trace frame corresponds to approximately 1 byte, the buffer overflows after approximately 4 million trace frames.

The instruction timing may be inferred from the timestamp of the transmitted frame.

2.3.1 EWARM / J-Trace toolchain example

This feature is not supported in EWARM 5.40/J-Trace CM3.

2.3.2 ARM-MDK / JTAGjet-Trace toolchain example

The trace window contains the timestamp TStamp field. It shows useful information on the traced code such as the execution time (in cycles) for each instruction.

After running the Instruction Timing example, where every physical sample contains the same 6 instructions, logically, the same execution time should be found for each sample. However, this is not the case, see *Figure 12*.

Generally, the timestamp cannot be relied upon for very short times, but the error of +/- 16 cycles has no real effect once longer times are being observed.

Control Enable Start	Resume Clear < Query	• >	Cuery Filter Fields Save		
# ITM	ITM PC Excpt	Disas	Source	TStamp (dt) [cyc]	MemAdd
#2711409/6	0800060C	BCC	0x8000600	+6	
#2711421	08000600	LDR	R1,[R4,#0xc]	+1	
#2711421/1	08000602	CMP	R1, R0	+1	
#2711421/2	08000604			+1	
#2711421/3	08000606	LDR	R1,[R4,#0xc]	+1 ≻ 8 cycles	
#2711421/4	08000608	ADD	R1, R1, #0×1	+1	
#2711421/5	0800060A	STR	R1,[R4,#0xc]	+1	
#2711421/6	0800060C	BCC	0x8000600	+2	
#2711429	08000600	LDR	R1,[R4,#0xc]	+1)	
#2711429/1	08000602	CMP	R1, R0	+1	
#2711429/2	08000604			+1	
#2711429/3	08000606	LDR	R1,[R4,#0xc]	+1 > 12 cycles	
#2711429/4	08000608	ADD	R1,R1,#0x1	+1	
#2711429/5	0800060A	STR	R1,[R4,#0xc]	+1	
#2711429/6	0800060C	BCC	0x8000600	+6.2	
#2711441	08000600	LDR	R1,[R4,#0xc]	+1)	
#2711441/1	08000602	CMP	R1, R0	+1	
#2711441/2	08000604			+1	
#2711441/3	08000606	LDR	R1,[R4,#0xc]	+1 > 12 cycles	
#2711441/4	08000608	ADD	R1, R1, #0x1	+1	
#2711441/5	0800060A	STR	R1,[R4,#0xc]	+1	
#2711441/6	0800060C	BCC	0x8000600	+6)	
#2711453	08000600	LDR	R1,[R4,#0xc]	+1	
#2711453/1	08000602	CMP	R1, R0	+1	
#2711453/2	08000604			+1	
#2711453/3	08000606	LDR	R1,[R4,#0xc]	+1 > 16 cycles	
#2711453/4	08000608	ADD	R1, R1, #0x1	+1	
#2711453/5	0800060A	STR	R1,[R4,#0xc]	+1	
#2711453/6	0800060C	BCC	0x8000600	+10/	
<					>
Status: NotActive,Full			Trace Full (100%)	Tra	ace Clock: 36.00MHz

Figure 12. ETM Trace window in ARM-MDK: Timestamps



2.4 Data tracing

Data tracing is a useful feature that allows the debugger to trace variable values or addresses during program execution in addition to the program flow. The STM32 ETM does not have data tracing capability. Nevertheless, data values and/or addresses can be traced by controlling the DWT and ITM on-chip modules.

2.4.1 EWARM / J-Trace toolchain example

This feature is not supported in EWARM 5.40/J-Trace CM3.

2.4.2 ARM-MDK / JTAGjet-Trace toolchain example

Using JTAGjet, it is possible to insert ITM trace data in the trace stream since it shares the same trace port with the ETM.

In the Data Tracing example, the ITM unit is configured to send data values when accessing the Counter variable. Meanwhile, the two trace sources (ITM and ETM) may get out of phase due to FIFO latency. The analysis of the code in the ETM trace window, see *Figure 13*, shows that the STR instruction is displayed after an ITM write record.

Figure 13. ETM Trace window in ARM-MDK

Control.	. Enable	Start Resume	Clear	Query 💌	>	Query	Filter Fields	Save			
#		ITM	ITMPar	PC	E	Disas			R	MemData	Source
	#4194235		Comp0 Comp0 Comp0	080006A4 080006A6 080006A8 080006A8 080006A2 080006AE 080006AE		[-]BCC LDR CMP LDR ADD (STR)	0x80005 R0,[R4,#0x R0,#0 R0,[R4,#0x R0,R0,#0x1 R0,[R4,#0x	10]	Rd Rd Wr	000000000000000000000000000000000000000	_
<	¥4194261⁄6			080006B2		B	0x80006b2				while (1);
Status: N	lotActive,Full					Trace Fu	(100%)				Trace C

2.5 Function profiler

The function profiler helps find the functions where most time is spent during application execution, and the number of calls of each function.

The following example aims to determine the number of times that a function was called using 2 different ways:

- 1. Using the function profiler.
- 2. Using a variable Tick incremented in the function interrupt handler

2.5.1 ARM-MDK / JTAGjet-Trace toolchain example

This feature is not supported in ARM-MDK 3.70/JTAGjet-Trace.



2.5.2 EWARM / J-Trace toolchain example

Using EWARM 5.40, the function profiler is available using SWV with J-Link and ETM trace with J-Trace. If the ETM is used, it shows also the number of calls of each function in addition to the time spent inside it.

Figure 14 shows the watch window with the Tick variable which is incremented every SysTickHandler() entry, and the function profiler window of the Function Profiler example. SysTickHandler() is the handler of the systick exception which occurs periodically (every 1 ms).

As illustrated by *Figure 14*, the number of SysTickHandler() calls is missed. This is due to the fact that the function profiler is based on the last ETM trace data collected, which does not contain all the function entries.

×	Expression	Value	Location	Туре					
÷	Tick (200	0x2000040C	∨u32					
Wat									
×	0 🕤 🖬 📻								
	Function			Calls	Flat Ti	me	Flat Time (%)	Acc. Time	Acc. Time (%)
	SPI2_IRQHandler()			0	0		0.00	0	0.00
	SPI3_IRQF	0	0		0.00	0	0.00		
	SVCHand	0	0		0.00	0	0.00		
ς	E SysTickHa	andler()		116	580		0.01	928	0.02
file		ounterCmd(u32)		0	0		0.00	0	0.00
Æ	SysTick_I1	0	0		0.00	0	0.00		
<u>s</u>	SysTick_S	etReload(u32)		0	0		0.00	0	0.00
Fund	Debug Log Build Function Profiler Breakpoints ETM Trace								

Figure 14. Function profiler window in EWARM

If the SysTick exception occurs more frequently (every 10 μ s for example), the difference is be more noticeable as shown in *Figure 15*.

Figure 15. Function profiler window in EWARM

×	Expression	Value	Location	Туре				
÷	Tick (23094)	0x2000040C	∨u32				
Wat								
×	0 1 8							
	Function			Calls	Flat Tin	ne Flat Time (%)	Acc. Time	Acc. Time (%)
	SPI2_IRQHandler()			0	0	0.00	0	0.00
	SPI3_IRQH	0	0	0.00	0	0.00		
	SVCHandl	0	0	0.00	0	0.00		
-	🖽 SysTickHa	andler()		(11522)	57610	1.56	92176	2.49
file	SysTick_C	counterCmd(u32)		0	0	0.00	0	0.00
Pro		Config(Function	alState)	0	0	0.00	0	0.00
<u>io</u>	SysTick_S	etReload(u32)		0	0	0.00	0	0.00
Fund	Pebug Log Build Function Profiler Breakpoints ETM Trace							

To avoid this discordance, the user should use a pre-filtered trace using the Trace start and Trace stop breakpoints to limit the amount of trace data.



3 Conclusion

Regarding the SWV feature, clearly the single serial wire port is not able to provide all trace information, because of the high speed of the STM32. Nevertheless, a statistical sampling indication of performance analysis is possible with both EWARM and ARM-MDK toolchains which is sufficiently adequate for profile analysis for the SWV feature.

Concerning the ETM feature, the buffer size of the trace capture device limits the instruction trace capability, nonetheless the STM32 ETM is able to provide an efficient debug and instruction trace especially when it is combined with the ITM trace to obtain a complete debug solution.



4 Revision history

Table 1.Document revision history

Date	Revision	Changes
27-Jan-2010	1	Initial release.



Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST's terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST'S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST REPRESENTATIVE, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER'S OWN RISK.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2010 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan -Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com

Doc ID 16243 Rev 1

