Cortex

series

SIGNUM SYSTEMS CORPORATION

ETM for Cortex-M3

User
Manual

SIGNUM
SYSTEMS

SIGNUM
SYSTEMS
1211 FLYNN RD., UNIT #104
CAMARILLO, CA 93012, U.S.A
PHONE 805 • 383 • 3682
WWW.SIGNUM.COM

Table of Contents

Introduction 1

Connecting Emulator to Cortex-M3 Board 1

Overview of Cortex-M3 Trace Architecture 2

JTAGjet-Trace and Trace S/W Components 4

Demo Programs 4

Enabling Cortex-M3 Trace 6

Selecting the Trace Port Size 6

Trace Clock and GPIO Initialization 7

Trace Window Overview 7

ETM Control Window Overview 8

Conditional Tracing (pre-filtering) 9

Starting Trace at Specified Address 9

Start/Stop Trace on Specified Addresses 10

Using Complex Trace Start/Stop Events 11

Data Value Tracing (ITM Trace) 12

Data Only Tracing 14

Adding PC to ITM Tracing 15

Tracing Exceptions (Interrupts) 16

Tracing Exceptions Using ITM Events 16

Tracing Exceptions in PC Trace 17

Understanding the Timestamp 18

Application Generated ITM Trace 19

DWT and ITM Profiling Related Features 20

Tracing DWT Counters 20

Tracing CPU Clock Cycles Counter 21

ITM PC Sampling 22

ITM Timestamp 23

Appendix A – Connecting to Specific Boards25

Connecting to ST STM3210E EVAL Board 25

Connecting to Keil MCBSTM32E Board 25

Document Revision History 1

Introduction

This document describes how to use the Signum JTAGjet-Trace emulator to trace Cortex-M3 processors. It covers the configuration and usage of the Chameleon and Keil uVision3 debuggers. Note that even though the examples employ Signum's Chameleon debugger, all trace-related windows and settings are identical in the Keil µV3 driver. The differences between the two debuggers are indicated in the text.

The scope of this document is limited to Cortex-M3 specifics. The general features of the ETM Trace window and operations are described in separate documents available through the **ARM Specific Documentation** option in Chameleon's **Help** menu reproduced here for your convenience.

DOCUMENT FILE	TITLE
etm_m3_um.pdf	ETM for Cortex-M3: User Manual
etb_um.pdf	ETB Addendum to Chameleon Debugger: User Manual
etm_um.pdf	ETM Addendum to Chameleon Debugger: User Manual
JTAGjet-Trace-ETM.pdf	ETM Trace: Presentation as PDF

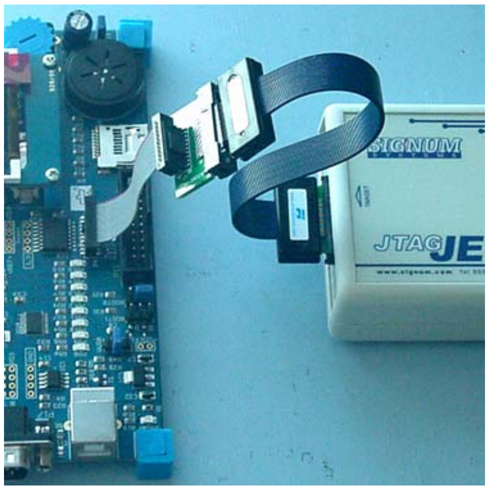
In the Keil µVision3 installation, the same set of PDF documents is available in the .\Signum\Docs subdirectory.

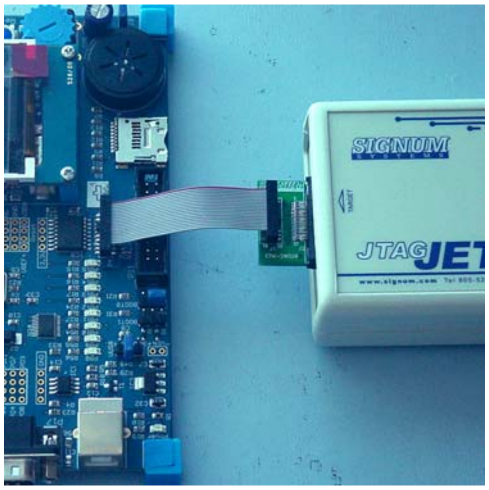
Connecting Emulator to Cortex-M3 Board

Cortex-M3 trace data can be brought out of the board by using one of two possible connectors:

- 20-pin new-style high-density (0.05 inch spacing) connector, or
- 38-pin Mictor ETM style connector (may exist on some custom-made boards).

The JTAGjet-Trace emulator can be attached to the 20-pin connector either with or without a dark-blue ETM Mictor cable, as shown in the illustrations below.





For a detailed description of all JTAGjet probes, including pin assignments, please refer to this document:

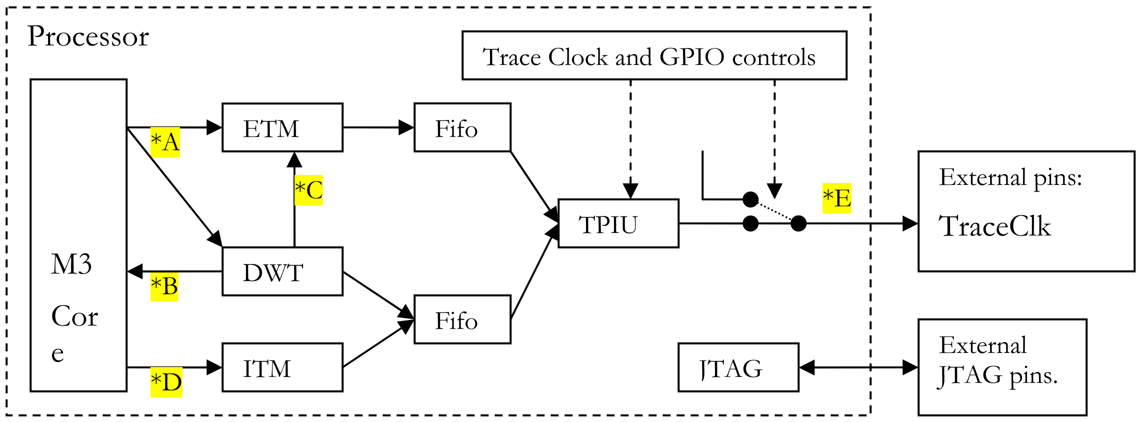
jtag_probes.pdf	JTAG Probes for Signum Emulators

Cution: Some older Cortex-M3 target boards, such as the ST STM32F EVAL and Keil MCBSTM32E, may not have keyed 20-pin connectors, in which pin 7 is removed, on them. On such boards, it may be necessary to either remove the key pin or remove the plug from the probe's key receptacle. Even if pin 7 is removed, the 20-pin Cortex ETM probe for the JTAGjet-Trace can be inserted incorrectly. Exercise caution to avoid faulty orientation of the probe or engaging only one row of pins, and use shrouded headers on your own boards. As the connector does not disconnect easily, do not pull it by its cable.

For illustrations of the proper connector orientation on popular STM32 boards, please refer to *Appendix A – Connecting to Specific Boards*.

Overview of Cortex-M3 Trace Architecture

It is essential to understand Cortex-M3 on-chip debug and trace related resources in order to use them efficiently in debugging and tracing. This section reviews the Cortex-M3 trace related modules, albeit without delving into detail.



ON-CHIP MODULES AND COMPONENTS

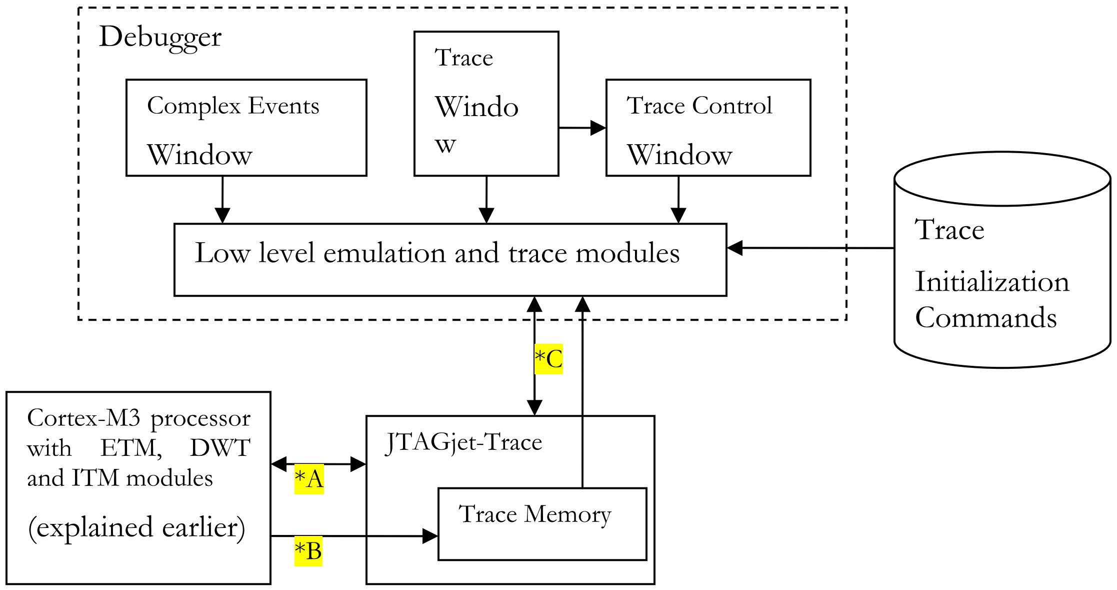
MODULE	DESCRIPTION
JTAG	Access to all modules and components through JTAG.
ETM	Embedded Trace Macrocell. Generates compressed instruction (PC flow) trace.
DWT	Data Watchpoint and Trace module. Generates ITM trace records (or ETM events) or stops the core using 4 comparators.
ITM	Instrumentation Trace Macrocell. Allows the application to insert 32-bit values to the trace stream by writing to the ITM Stimulus Port registers (located at range 0xE0000000-0xE000007C).
Fifo	FIFO buffers (different sizes for ETM and ITM).
TPIU	Trace Port Interface Unit. Combines ETM and ITM/DWT trace streams into 1, 2 or 4-bit TraceData clocked by TraceCLK signal.
Trace Clock & GPIO controls	Processor dependent modules that provide trace clock and control trace pins (possibly shared with GPIO).
Processor	All above modules, memory and peripherals (not shown).

CONTROL AND TRACE DATA FLOW

***A**	Used to observe CPU execution by the **ETM** and **DWT**.
***B**	Used to stop the code on a watchpoint detected by the **DWT**.
***C**	Used to signal events to the **ETM** (start or stop the trace).
***D**	Used to write application-generated **ITM** trace records.
***E**	External trace pins. **TraceClk** and **TraceData** (1, 2 or 4 pins).

A detailed description of Cortex-M3 debug architecture can be found in the pdf file DDI 0337G available at www.arm.com.

JTAGjet-Trace and Trace S/W Components


Debugger
Complex Events Window
Trace Window
Trace Control Window
Low level emulation and trace modules
Trace Initialization Commands
*C
Cortex-M3 processor with ETM, DWT and ITM modules (explained earlier)
*A
*B
JTAGjet-Trace
Trace Memory

The picture above shows the basic components of the trace software. Each of these components is responsible for controlling different parts of the on-chip trace modules.

CONNECTIONS AND COMPONENTS

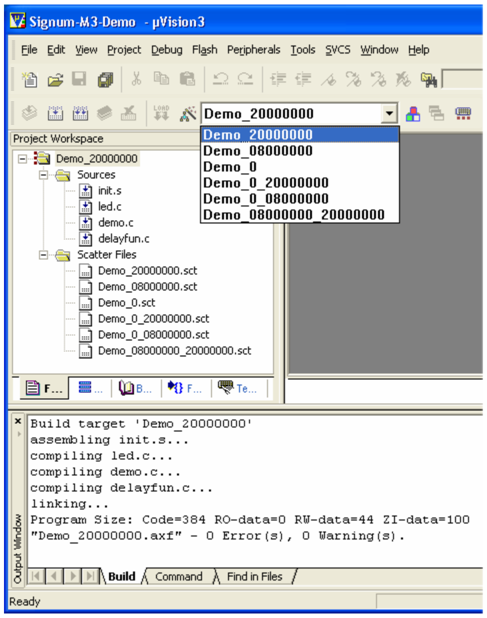
***A**	JTAG pins.
***B**	**TraceClk** and **TraceData** (raw 1/2/4 bit trace).
***C**	USB 2.0 (480MHz high speed) connection.
JTAGjet-Trace	JTAGjet-Trace emulator hardware.
Trace Memory	Trace memory. Captures raw trace samples.
Debugger	Chameleon or third-party debugger like Keil µV3.
Complex Events Window	Controls the on-chip DWT and ITM modules. The DWT may generate events for the ETM.
Trace Window	Displays decoded trace – ETM and ITM.
Trace Control Window	Controls the on-chip ETM module.
Trace Initialization Commands	Optional processor-dependent commands to enable the trace clock and configure the GPIO pins.

Demo Programs

The demo program used in this manual is located in one of the following folders.

FOLDER	CONTENTS
.\ARM\Demos\Generic\CortexM3\Demo	Chameleon sub-directory
.\Signum\Demos\Generic\CortexM3\Demo	Keil's µV3 sub-directory

The program was compiled using Keil's uVision3 MDK 3.20 with the Signum-M3-Demo.uv2 workspace file:



```
Build target 'Demo_20000000'
assembling init.s...
compiling led.c...
compiling demo.c...
compiling delayfun.c...
linking...
Program Size: Code=384 RO-data=0 RW-data=44 ZI-data=100
"Demo_20000000.axf" - 0 Error(s), 0 Warning(s).
```

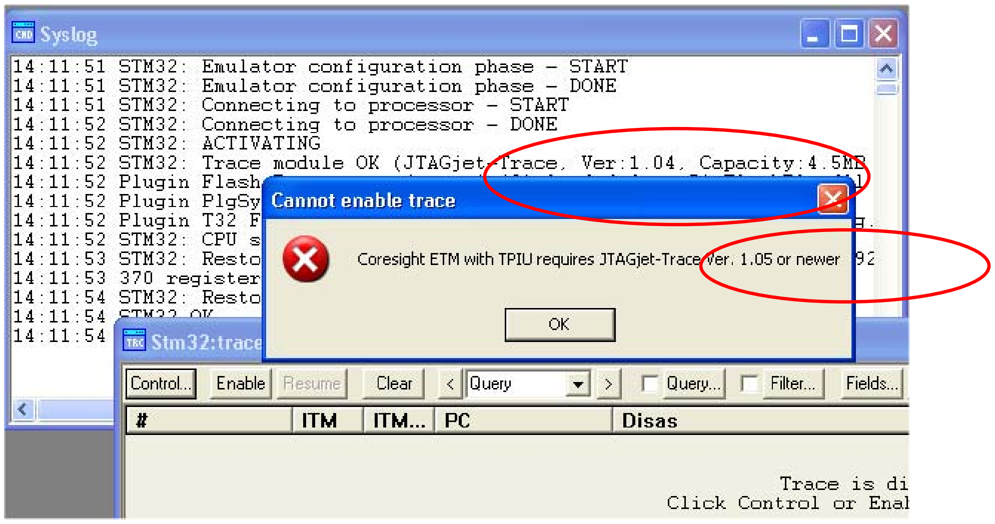
Since the address of RAM varies from one Cortex-M3 board to another, the following ELF files — also shown in the target list in the illustration above — can be generated.

FILE	DESCRIPTION
Demo_0.axf	Code and data at 0x0
Demo_20000000.axf	Code and data at 0x2000_0000 (most common)
Demo_08000000.axf	Code and data 0x 0x0800_0000 (for TMS570)
Demo_0_20000000.axf	Code at 0, data at 0x2000_0000 (for flash)
Demo_0_08000000.axf	Code at 0, data at 0x0800_0000 (for TMS570)
Demo_08000000_20000000.axf	Code at 0x0800_0000, data at 0x2000_0000 (for STM32 flash)

Our examples were loaded from within Chameleon into the STM32 processor, in which RAM is at 0x20000000. Therefore, we used the Demo_20000000.axf file. Even though you may need to load a different file, your result should be the same, since the settings in the examples do not rely on physical addresses.

Enabling Cortex-M3 Trace

Trace collection is not enabled by default when the debugger starts. To activate the feature, open the Trace window by selecting Trace from the View menu. Press the Enable button. You may be required to upgrade your JTAGjet emulator:



```
14:11:51 STM32: Emulator configuration phase - START
14:11:51 STM32: Emulator configuration phase - DONE
14:11:51 STM32: Connecting to processor - START
14:11:52 STM32: Connecting to processor - DONE
14:11:52 STM32: ACTIVATING
14:11:52 STM32: Trace module OK (JTAGjet-Trace, Ver:1.04, Capacity:4.5MB
14:11:52 Plugin Flash
14:11:52 Plugin PlgSy
14:11:52 Plugin T32 F
14:11:52 STM32: CPU s
14:11:53 STM32: Resto
14:11:53 370 register
14:11:54 STM32: Resto
14:11:54 STM32 OK
14:11:54
```

Cannot enable trace

Coresight ETM with TPIU requires JTAGjet-Trace ver. 1.05 or newer

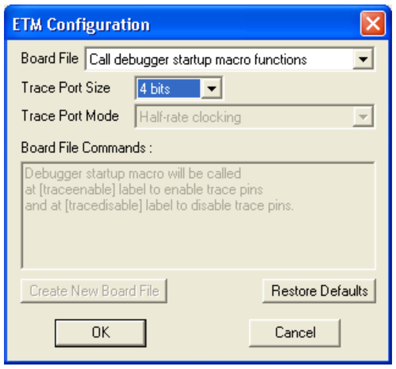
```
Trace is di
Click Control or Enal
```

If the emulator does require an update, please contact sales@signum.com. You still will be able to use the old JTAGjet-Trace hardware for basic debugging of Cortex-M3. Trace will not be shown, however.

Selecting the Trace Port Size

When enabling the trace for very first time, you will be asked to select its port size (see the figures).

In Chameleon:

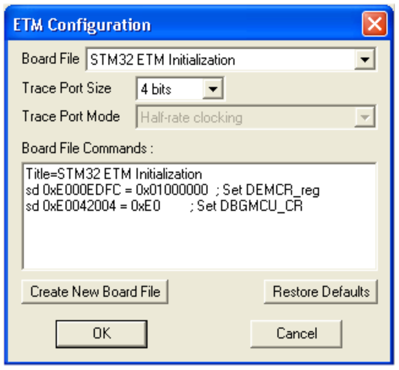


Board File	Call debugger startup macro functions
Trace Port Size	4 bits
Trace Port Mode	Half-rate clocking

Board File Commands :

Debugger startup macro will be called
at [traceenable] label to enable trace pins
and at [tracedisable] label to disable trace pins.

In Keil's µV3:



Board File	STM32 ETM Initialization
Trace Port Size	4 bits
Trace Port Mode	Half-rate clocking

Board File Commands :

```
Title=STM32 ETM Initialization
sd 0xE000EDFC = 0x01000000  ; Set DEMCR_reg
sd 0xE0042004 = 0xE0        ; Set DBGMCU_CR
```

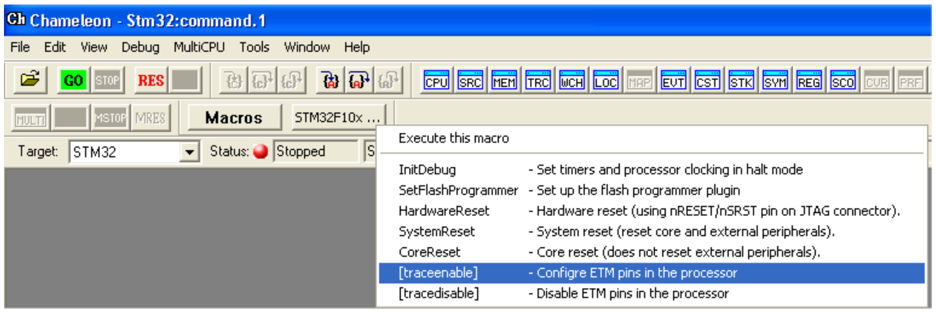
The Cortex-M3 processor is only capable of generating trace using 1, 2 or 4 TraceData pins. The current release of the JTAGjet-Trace hardware (1.05, as of this writing) supports only the widest 4-bit trace port. Any other values are ignored. As of September 2008, all the

Cortex-M3 chips we tested supported the 4-bit port. Consequently, there is no immediate need to support 1- and 2-bit tracing with their limited bandwidths.

Trace Clock and GPIO Initialization

Certain processors, such as the STM32, must be properly configured, or else the on-chip trace module will be disabled. Similarly, the appropriate GPIO pins must be configured to enable trace signal export. Since such configuration is processor dependent, it is performed by an external script.

In Signum Chameleon, the startup macro can be executed either by selecting the **[traceenable]** label or from the **Macros** toolbar:


Chameleon - Stm32:command.1
File Edit View Debug MultiCPU Tools Window Help
GO STOP RES
CPU SRC MEM TRC WCH LOC MAP EVT CST STK SYM REG SCO CUR PRF
MULTI MSTOP MRES Macros STM32F10x ...
Target: STM32
Status: Stopped

Execute this macro

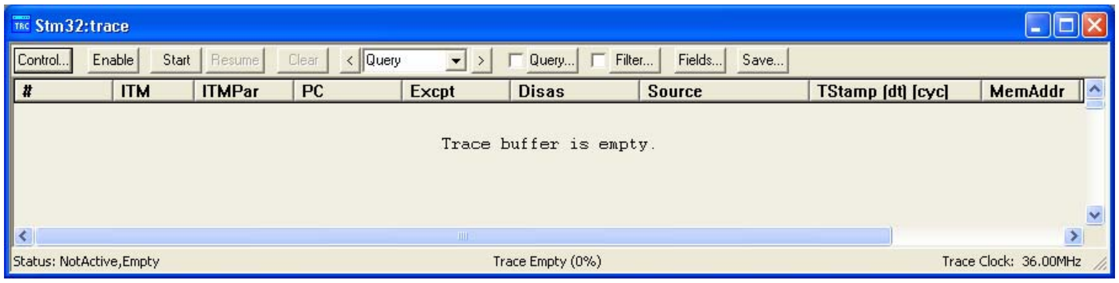
InitDebug	- Set timers and processor clocking in halt mode
SetFlashProgrammer	- Set up the flash programmer plugin
HardwareReset	- Hardware reset (using nRESET/nSRST pin on JTAG connector).
SystemReset	- System reset (reset core and external peripherals).
CoreReset	- Core reset (does not reset external peripherals).
[traceenable]	- Configre ETM pins in the processor
[tracedisable]	- Disable ETM pins in the processor

In Keil µV3, make sure that you have assigned a proper **Board File** (e.g., STM32 ETM Initialization), as explained in the previous chapter.

All internal Cortex-M3 trace related modules — ETM, ITM, DWT, and TPIU — are configured without the need for any additional CPU-specific settings.

Trace Window Overview

After successfully enabling the trace, you should see the following empty trace window:

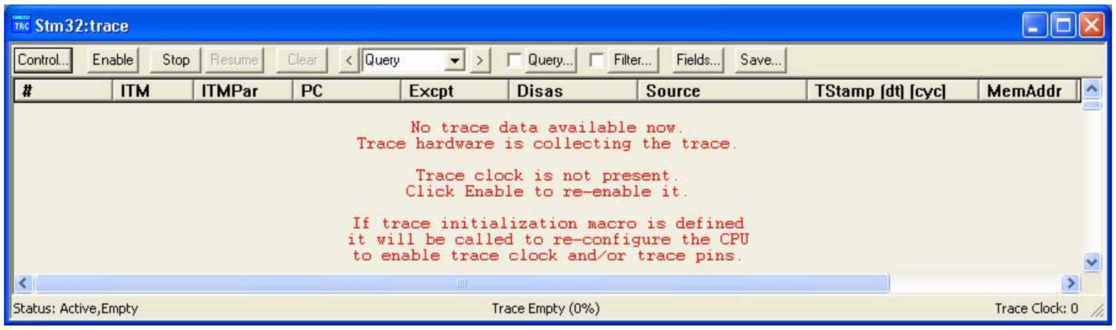

Stm32:trace
Control... Enable Start Resume Clear < Query > Query... Filter... Fields... Save...
# ITM ITMPar PC Excpt Disas Source TStamp [dt] [cyc] MemAddr

Trace buffer is empty.

Status: NotActive,Empty   Trace Empty (0%)   Trace Clock: 36.00MHz

The right side of Trace Status Bar at the bottom of the Trace window shows the Trace Clock frequency. Since Cortex-M3 supports only half-rate clocking, this frequency is always half of the current CPU frequency. In the figure above, the CPU runs at 72 MHz (2*36 MHz).

The JTAGjet-Trace hardware measures the frequency of the trace clock even when the trace is not being collected. Trace Clock with a value of 0 indicates that the trace clock is not available. Re-enable the trace by clicking the **Enable** button, as instructed by the Trace window:


Stm32:trace
Control... Enable Stop Resume Clear < Query > Query... Filter... Fields... Save...
# ITM ITMPar PC Excpt Disas Source TStamp [dt] [cyc] MemAddr

```
No trace data available now.
Trace hardware is collecting the trace.

Trace clock is not present.
Click Enable to re-enable it.

If trace initialization macro is defined
it will be called to re-configure the CPU
to enable trace clock and/or trace pins.
```

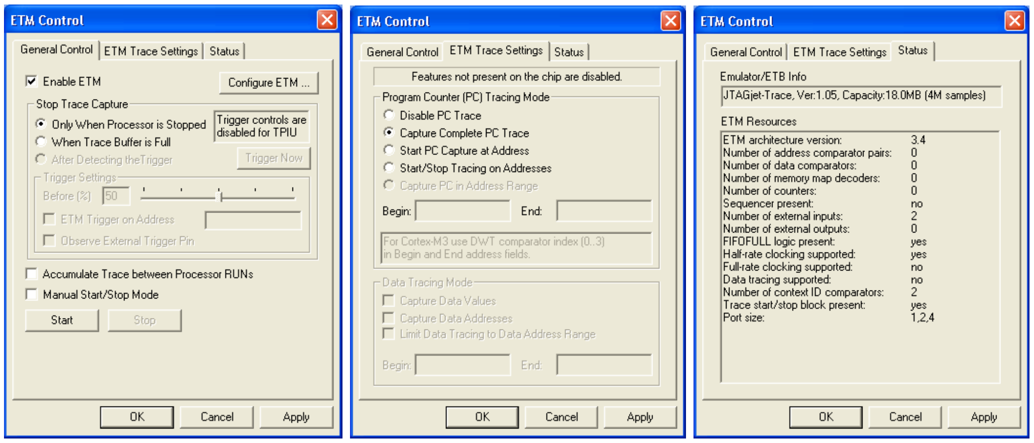
Status: Active,Empty   Trace Empty (0%)   Trace Clock: 0

To troubleshoot any possible difficulties, make sure that:

- the correct trace initialization macro is selected,
- Trace Clock is visible (executing the macro manually), and
- your application is not disabling the ETM pins.

ETM Control Window Overview

To display the **ETM Control** window, click the **Control** button in the left corner of the **Trace** window. The ETM Control window has 3 tabs, the default setting of which is shown in the figure below.



General Control:
- [✓] Enable ETM
- Stop Trace Capture: [● Only When Processor is Stopped] / When Trace Buffer is Full / After Detecting theTrigger
- Trigger controls are disabled for TPIU
- Trigger Settings: Before (%) 50; ETM Trigger on Address; Observe External Trigger Pin
- [ ] Accumulate Trace between Processor RUNs
- [ ] Manual Start/Stop Mode

ETM Trace Settings:
- Features not present on the chip are disabled.
- Program Counter (PC) Tracing Mode: Disable PC Trace / [● Capture Complete PC Trace] / Start PC Capture at Address / Start/Stop Tracing on Addresses / Capture PC in Address Range
- Begin: End:
- For Cortex-M3 use DWT comparator index (0..3) in Begin and End address fields.
- Data Tracing Mode: [ ] Capture Data Values; [ ] Capture Data Addresses; [ ] Limit Data Tracing to Data Address Range; Begin: End:

Status:

Emulator/ETB Info: JTAGjet-Trace, Ver:1.05, Capacity:18.0MB (4M samples)

ETM Resources

ETM architecture version:	3.4
Number of address comparator pairs:	0
Number of data comparators:	0
Number of memory map decoders:	0
Number of counters:	0
Sequencer present:	no
Number of external inputs:	2
Number of external outputs:	0
FIFOFULL logic present:	yes
Half-rate clocking supported:	yes
Full-rate clocking supported:	no
Data tracing supported:	no
Number of context ID comparators:	2
Trace start/stop block present:	yes
Port size:	1,2,4

The **General Control** tab allows you to control the processes of starting and stopping trace capture. These settings are Cortex-M3 independent. See the etm_um.pdf file for details.

The **ETM Trace Settings** tab lets you decide what will be captured in the trace buffer. These settings are specific to Cortex-M3. Since the Cortex-M3 ETM does not have any address comparators for conditional tracing, the DWT comparators and **Complex Events** windows must be configured to control what is emitted on the ETM pins and consequently captured by the JTAGjet trace memory.

The **Status** tab shows the JTAGjet-Trace hardware capabilities and the on-chip ETM resources of the processor in use. These capabilities are determined by the device manufacturer and the ETM architecture. The version of the ETM Architecture for Cortex-M3 is 3.4. The available resources are quite limited compared to the ETM implementations for other ARM cores and processors. Certain controls, like **Trigger Settings** and **Data Tracing Mode**, are not applicable to Cortex-M3, and thus grayed out.

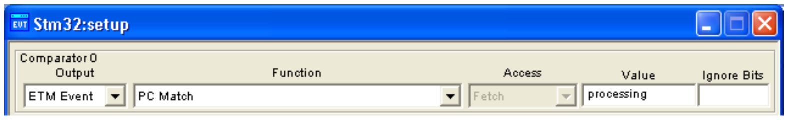
Conditional Tracing (pre-filtering)

The **ETM Trace Settings** tab in the **ETM Control** window must specify the DWT comparator index to be used as the trace start/stop condition. This chapter concerns itself with tracing that starts at a specified code address.

Starting Trace at Specified Address

Since no trace stop event is defined, you should select **Stop Trace Capture/When Trace Buffer is Full** in the **General Control** tab. Otherwise, the trace buffer will overflow and the starting address in the trace will get overwritten.

The DWT comparator 0 is defined in the **Complex Events** window as a **PC Match** on a specified address. The definition is assigned to the function **processing** whose name is entered in the Value field. The field allows you to enter either the function's C/C++ name or address. The Output of the DWT comparator is set to **ETM Event**, which ensures that it will be used by the ETM module.

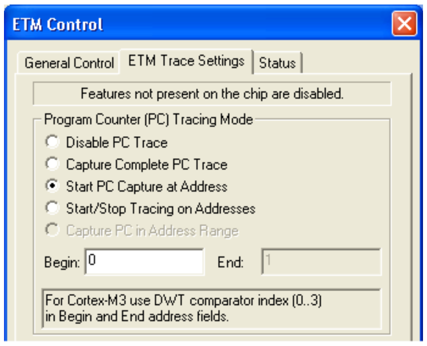


Stm32:setup

Comparator 0

Output	Function	Access	Value	Ignore Bits
ETM Event	PC Match	Fetch	processing	

The DWT comparator index 0 in the **Begin** field makes the trace start when the ETM event generated by DWT Comparator 0 occurs, that is, when the PC reaches the **processing** function.



ETM Control

General Control | ETM Trace Settings | Status

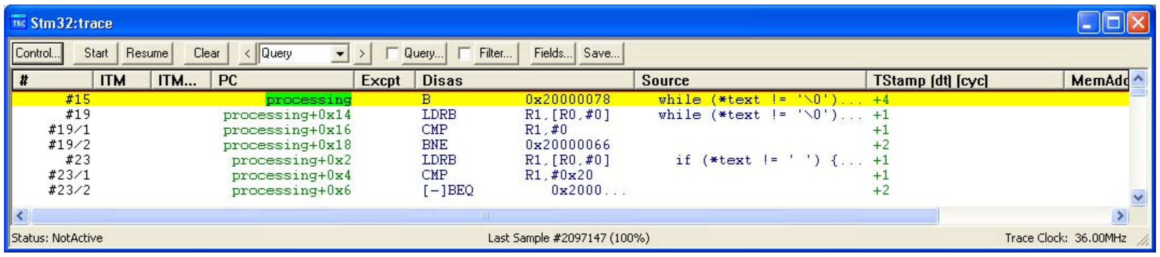
Features not present on the chip are disabled.

Program Counter (PC) Tracing Mode
- Disable PC Trace
- Capture Complete PC Trace
- [✓ Start PC Capture at Address]
- Start/Stop Tracing on Addresses
- Capture PC in Address Range

Begin: 0 End: 1

For Cortex-M3 use DWT comparator index (0..3) in Begin and End address fields.

As the function **processing** is the triggering event, its name appears in the PC field on the first line in the **Trace** window. The PC field displays symbols in lieu of addresses.



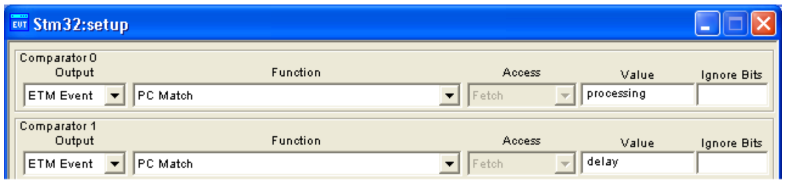
Stm32:trace

```
#        ITM  ITM...  PC                  Excpt  Disas                      Source                      TStamp [dt] [cyc]  MemAdd
   #15                       processing          B       0x20000078         while (*text != '\0')...  +4
   #19               processing+0x14             LDRB    R1,[R0,#0]         while (*text != '\0')...  +1
 #19/1               processing+0x16             CMP     R1,#0                                        +1
 #19/2               processing+0x18             BNE     0x20000066                                   +2
   #23                processing+0x2             LDRB    R1,[R0,#0]           if (*text != ' ') {...  +1
 #23/1                processing+0x4             CMP     R1,#0x20                                     +1
 #23/2                processing+0x6             [-]BEQ     0x2000...                                 +2
```

Status: NotActive   Last Sample #2097147 (100%)   Trace Clock: 36.00MHz

Start/Stop Trace on Specified Addresses

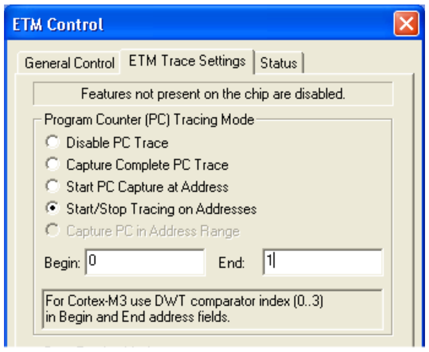
On many occasions, there is no need to trace the entire program. Instead, it suffices to limit the trace to a certain part of the code. To accomplish this, you should define two DWT comparators as shown.



Stm32:setup

	Output	Function	Access	Value	Ignore Bits
Comparator 0	ETM Event	PC Match	Fetch	processing	
Comparator 1	ETM Event	PC Match	Fetch	delay	

Then assign the DWT comparator indexes 0 and 1, respectively, to the **Begin** and **End** of the capture address range.



ETM Control

General Control | ETM Trace Settings | Status

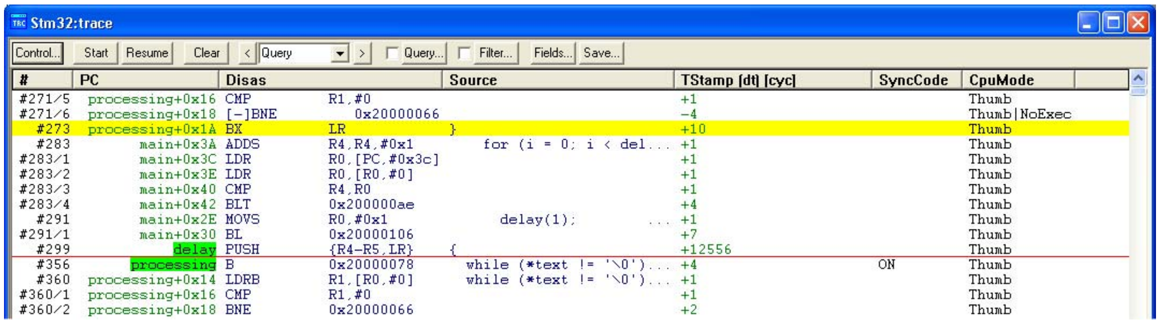
Features not present on the chip are disabled.

Program Counter (PC) Tracing Mode
- ( ) Disable PC Trace
- ( ) Capture Complete PC Trace
- ( ) Start PC Capture at Address
- (●) Start/Stop Tracing on Addresses
- ( ) Capture PC in Address Range

Begin: 0   End: 1

For Cortex-M3 use DWT comparator index (0..3) in Begin and End address fields.

Program tracing will start and stop on these two DWT-generated ETM events, as shown below.



Stm32:trace

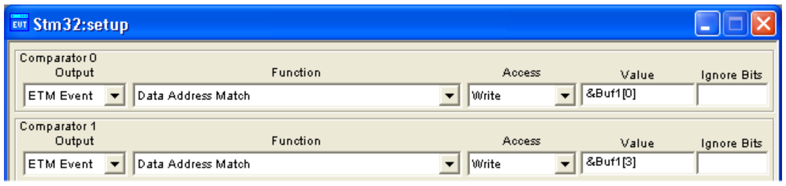
```
#       PC               Disas                      Source                       TStamp [dt] [cyc]  SyncCode  CpuMode
#271/5  processing+0x16  CMP     R1,#0                                           +1                           Thumb
#271/6  processing+0x18  [-]BNE     0x20000066                                   -4                           Thumb|NoExec
  #273  processing+0x1A  BX      LR                 }                            +10                          Thumb
  #283        main+0x3A  ADDS    R4,R4,#0x1            for (i = 0; i < del...    +1                           Thumb
#283/1        main+0x3C  LDR     R0,[PC,#0x3c]                                   +1                           Thumb
#283/2        main+0x3E  LDR     R0,[R0,#0]                                      +1                           Thumb
#283/3        main+0x40  CMP     R4,R0                                           +1                           Thumb
#283/4        main+0x42  BLT     0x200000ae                                      +4                           Thumb
  #291        main+0x2E  MOVS    R0,#0x1                delay(1);          ...   +1                           Thumb
#291/1        main+0x30  BL      0x20000106                                      +7                           Thumb
  #299            delay  PUSH    {R4-R5,LR}         {                            +12556                       Thumb
  #356       processing  B       0x20000078           while (*text != '\0')...   +4                 ON        Thumb
  #360  processing+0x14  LDRB    R1,[R0,#0]           while (*text != '\0')...   +1                           Thumb
#360/1  processing+0x16  CMP     R1,#0                                           +1                           Thumb
#360/2  processing+0x18  BNE     0x20000066                                      +2                           Thumb
```

The red line is a trace discontinuity marker. It indicates the place where trace was not collected. The timestamp **TStamp** field shows in delta mode **[dt]** that the trace skipped 12556 trace clocks. The **ON** symbol in **SyncCode** field denotes the reason for the trace restart, in this case, the trace ON condition. The yellow trace cursor is located on the return from the previous call to the **processing** function, which happens to be the **BX LR** instruction. After the return, the code executes some instructions in the **main** function (main+0x…), after which the **delay** function is called. The first instruction of the **delay** function appears in sample 299. Trace collection stops right after since DWT comparator 1 detected a PC value match.

Note: As tracing starts on one event and stops on another, all function, library calls and interrupts in between are traced. Unlike with the on-chip ETM modules that have built-in address matches (ARM7, ARM9 and ARM11), Cortex-M3 does not provide for tracing exclusively within a specific continuous PC range.

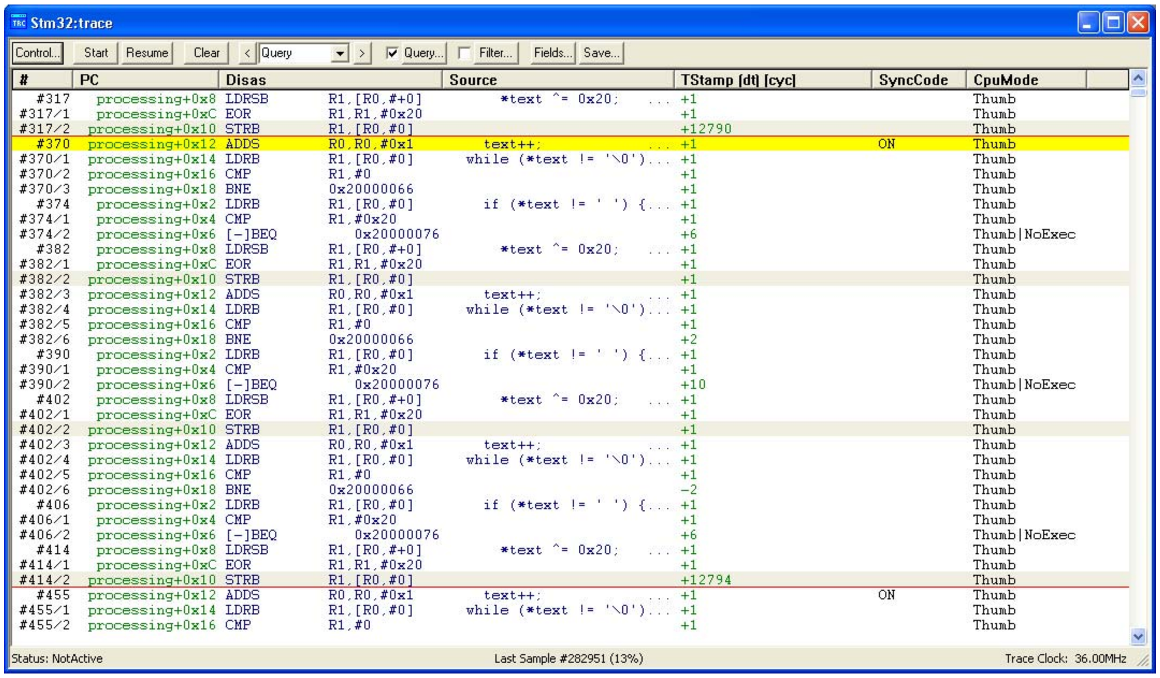
Using Complex Trace Start/Stop Events

The examples in the previous chapters used **PC Match** to generate the ETM start and stop events. More complex events that trigger on data values can also be applied. The following ETM Events settings enables you to trace only the code that writes four elements to the array **Buf1**, starting on write to **Buf1[0]** and stopping on write to **Buf1[3]**.


Stm32:setup

	Output	Function	Access	Value	Ignore Bits
Comparator 0	ETM Event	Data Address Match	Write	&Buf1[0]	
Comparator 1	ETM Event	Data Address Match	Write	&Buf1[3]	

The corresponding trace may look as follows.



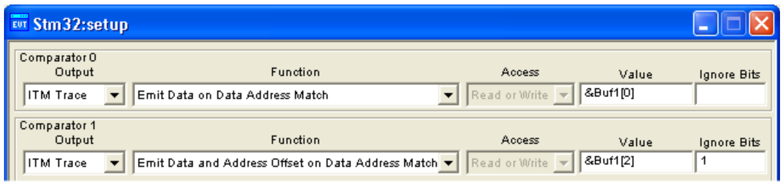
```
#        PC                 Disas                         Source                       TStamp [dt] [cyc]   SyncCode   CpuMode
  #317    processing+0x8  LDRSB      R1,[R0,#+0]          *text ^= 0x20;      ...  +1                             Thumb
#317/1    processing+0xC  EOR        R1,R1,#0x20                                   +1                             Thumb
#317/2   processing+0x10  STRB       R1,[R0,#0]                                    +12790                         Thumb
  #370   processing+0x12  ADDS       R0,R0,#0x1         text++;               ...  +1                  ON         Thumb
#370/1   processing+0x14  LDRB       R1,[R0,#0]       while (*text != '\0')...     +1                             Thumb
#370/2   processing+0x16  CMP        R1,#0                                         +1                             Thumb
#370/3   processing+0x18  BNE        0x20000066                                    +1                             Thumb
  #374    processing+0x2  LDRB       R1,[R0,#0]         if (*text != ' ') {...     +1                             Thumb
#374/1    processing+0x4  CMP        R1,#0x20                                      +1                             Thumb
#374/2    processing+0x6  [-]BEQ        0x20000076                                 +6                             Thumb|NoExec
  #382    processing+0x8  LDRSB      R1,[R0,#+0]          *text ^= 0x20;      ...  +1                             Thumb
#382/1    processing+0xC  EOR        R1,R1,#0x20                                   +1                             Thumb
#382/2   processing+0x10  STRB       R1,[R0,#0]                                    +1                             Thumb
#382/3   processing+0x12  ADDS       R0,R0,#0x1         text++;               ...  +1                             Thumb
#382/4   processing+0x14  LDRB       R1,[R0,#0]       while (*text != '\0')...     +1                             Thumb
#382/5   processing+0x16  CMP        R1,#0                                         +1                             Thumb
#382/6   processing+0x18  BNE        0x20000066                                    +2                             Thumb
  #390    processing+0x2  LDRB       R1,[R0,#0]         if (*text != ' ') {...     +1                             Thumb
#390/1    processing+0x4  CMP        R1,#0x20                                      +1                             Thumb
#390/2    processing+0x6  [-]BEQ        0x20000076                                 +10                            Thumb|NoExec
  #402    processing+0x8  LDRSB      R1,[R0,#+0]          *text ^= 0x20;      ...  +1                             Thumb
#402/1    processing+0xC  EOR        R1,R1,#0x20                                   +1                             Thumb
#402/2   processing+0x10  STRB       R1,[R0,#0]                                    +1                             Thumb
#402/3   processing+0x12  ADDS       R0,R0,#0x1         text++;               ...  +1                             Thumb
#402/4   processing+0x14  LDRB       R1,[R0,#0]       while (*text != '\0')...     +1                             Thumb
#402/5   processing+0x16  CMP        R1,#0                                         +1                             Thumb
#402/6   processing+0x18  BNE        0x20000066                                    -2                             Thumb
  #406    processing+0x2  LDRB       R1,[R0,#0]         if (*text != ' ') {...     +1                             Thumb
#406/1    processing+0x4  CMP        R1,#0x20                                      +1                             Thumb
#406/2    processing+0x6  [-]BEQ        0x20000076                                 +6                             Thumb|NoExec
  #414    processing+0x8  LDRSB      R1,[R0,#+0]          *text ^= 0x20;      ...  +1                             Thumb
#414/1    processing+0xC  EOR        R1,R1,#0x20                                   +1                             Thumb
#414/2   processing+0x10  STRB       R1,[R0,#0]                                    +12794                         Thumb
  #455   processing+0x12  ADDS       R0,R0,#0x1         text++;               ...  +1                  ON         Thumb
#455/1   processing+0x14  LDRB       R1,[R0,#0]       while (*text != '\0')...     +1                             Thumb
#455/2   processing+0x16  CMP        R1,#0                                         +1                             Thumb

Status: NotActive                    Last Sample #282951 (13%)                    Trace Clock:  36.00MHz
```

Let's take a closer look at this trace display. The tagged, gray lines show **STRB** instructions, which actually write to the **Buf1** array. The tagging of an address is accomplished by double-clicking an address in the **PC** column. The first captured instruction after the event (sample 370) is not **STRB**, but rather **ADDS**, which immediately follows it. This is because a data address match is detected when **STRB** instruction is already in the execution phase. The last instruction traced appears in sample 412/2, which correctly shows the trace stop ETM Event.

The time stamp in the **TStamp** column displays timing differences. The prevailing values of +1 are interspersed with +2, +6 or +10. The duration of trace discontinuity is shown as +12790 and +12794. Such small timestamp inaccuracies are unavoidable and are discussed in Section *Understanding the Timestamp* further in the text.

Data Value Tracing (ITM Trace)

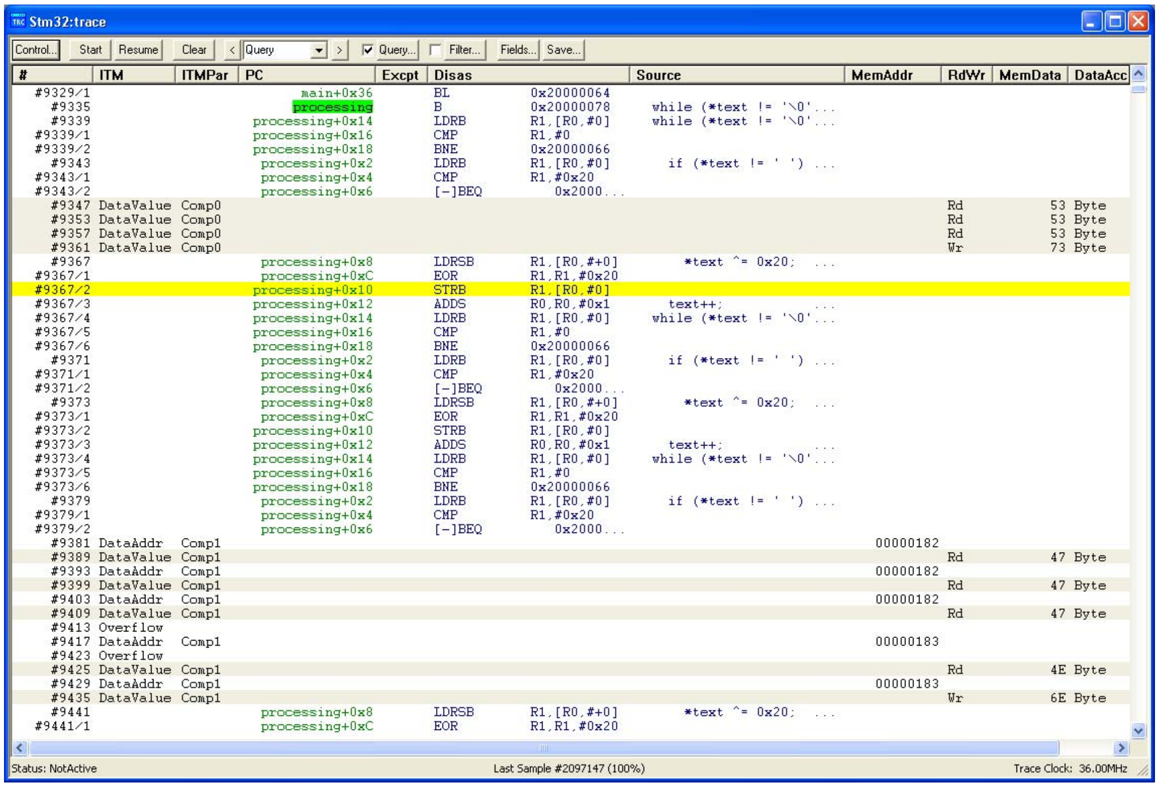
Cortex-M3 ETM itself does not have data tracing capability. However, Data values and/or addresses can be inserted into the trace stream by controlling the DWT and ITM on-chip modules. The following **Complex Events** settings



Stm32:setup

	Output	Function	Access	Value	Ignore Bits
Comparator 0	ITM Trace	Emit Data on Data Address Match	Read or Write	&Buf1[0]	
Comparator 1	ITM Trace	Emit Data and Address Offset on Data Address Match	Read or Write	&Buf1[2]	1

will result in the following trace.



#	ITM	ITMPar	PC	Excpt	Disas	Source	MemAddr	RdWr	MemData	DataAcc
#9329/1			main+0x36		BL 0x20000064					
#9335			processing		B 0x20000078	while (*text != '\0'...				
#9339			processing+0x14		LDRB R1,[R0,#0]	while (*text != '\0'...				
#9339/1			processing+0x16		CMP R1,#0					
#9339/2			processing+0x18		BNE 0x20000066					
#9343			processing+0x2		LDRB R1,[R0,#0]	if (*text != ' ') ...				
#9343/1			processing+0x4		CMP R1,#0x20					
#9343/2			processing+0x6		[-]BEQ 0x2000...					
#9347	DataValue	Comp0						Rd	53	Byte
#9353	DataValue	Comp0						Rd	53	Byte
#9357	DataValue	Comp0						Rd	53	Byte
#9361	DataValue	Comp0						Wr	73	Byte
#9367			processing+0x8		LDRSB R1,[R0,#+0]	*text ^= 0x20; ...				
#9367/1			processing+0xC		EOR R1,R1,#0x20					
#9367/2			processing+0x10		STRB R1,[R0,#0]					
#9367/3			processing+0x12		ADDS R0,R0,#0x1	text++; ...				
#9367/4			processing+0x14		LDRB R1,[R0,#0]	while (*text != '\0'...				
#9367/5			processing+0x16		CMP R1,#0					
#9367/6			processing+0x18		BNE 0x20000066					
#9371			processing+0x2		LDRB R1,[R0,#0]	if (*text != ' ') ...				
#9371/1			processing+0x4		CMP R1,#0x20					
#9371/2			processing+0x6		[-]BEQ 0x2000...					
#9373			processing+0x8		LDRSB R1,[R0,#+0]	*text ^= 0x20; ...				
#9373/1			processing+0xC		EOR R1,R1,#0x20					
#9373/2			processing+0x10		STRB R1,[R0,#0]					
#9373/3			processing+0x12		ADDS R0,R0,#0x1	text++; ...				
#9373/4			processing+0x14		LDRB R1,[R0,#0]	while (*text != '\0'...				
#9373/5			processing+0x16		CMP R1,#0					
#9373/6			processing+0x18		BNE 0x20000066					
#9379			processing+0x2		LDRB R1,[R0,#0]	if (*text != ' ') ...				
#9379/1			processing+0x4		CMP R1,#0x20					
#9379/2			processing+0x6		[-]BEQ 0x2000...					
#9381	DataAddr	Comp1					00000182			
#9389	DataValue	Comp1						Rd	47	Byte
#9393	DataAddr	Comp1					00000182			
#9399	DataValue	Comp1						Rd	47	Byte
#9403	DataAddr	Comp1					00000182			
#9409	DataValue	Comp1						Rd	47	Byte
#9413	Overflow									
#9417	DataAddr	Comp1					00000183			
#9423	Overflow									
#9425	DataValue	Comp1						Rd	4E	Byte
#9429	DataAddr	Comp1					00000183			
#9435	DataValue	Comp1						Wr	6E	Byte
#9441			processing+0x8		LDRSB R1,[R0,#+0]	*text ^= 0x20; ...				
#9441/1			processing+0xC		EOR R1,R1,#0x20					

Status: NotActive   Last Sample #2097147 (100%)   Trace Clock: 36.00MHz

In addition to the usual PC trace in which complete PC tracing takes place, you can see the **DataValue** and **DataAddr** trace records in the **ITM** column of the Trace window. Note the gray tagging of the **DataValue** samples.

The field **IMPPar** is an additional ITM-dependent parameter that displays the comparator index, **Comp0** or **Comp1**, generating the particular data trace record. The **MemData** field shows the data values recorded during the write or read operations. The **RdWr** field specifies the access type, **Rd** or **Wr**, while the column **DataAcc** specifies the access width: **Byte, Word** or **Dword**.

As of this writing, the **MemAddr** filed merely shows an offset to the specified address value, the real address of **Buf1[2]** being 0x20000182. Future versions of the program will display real addresses, as opposed to offsets.

One bit in comparator 1 is set as "ignore bits," which allows us to trace the address range 0x20000182-0x20000183 using only one comparator.

Our trace warrants a few additional comments:

As you can see, several ITM records are adjacent to each other. This is because the ITM has its own FIFO and several records are buffered before they are exported by the TPIU and joined with the ETM trace.

An analysis of the code in the trace buffer reveals that the first **STRB** instruction, indicated by the yellow trace cursor on sample 9367/2, is actually displayed after an ITM write record (sample 9361) which shows the value written by this instruction to the memory. This is also related to FIFO related latency.

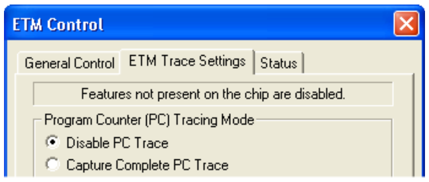
Samples 9413 and 9423 are marked as **Overflow**. These are ITM FIFO overflows. These do not affect PC tracing, since the ETM trace has a priority over the ITM trace.

It is not possible to trace only reads or only writes. The trace will include both types of access.

The code traced in this example does not seem to be optimized – it reads the same value 3 times before writing it. The trace allows you to see how compilation options may change the timing of your code.

Data Only Tracing

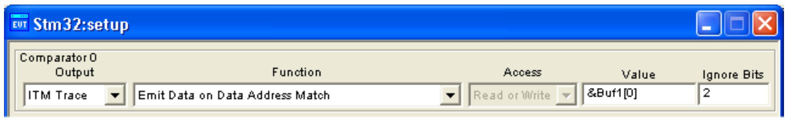
You can disable PC tracing by using the following settings:



Features not present on the chip are disabled.

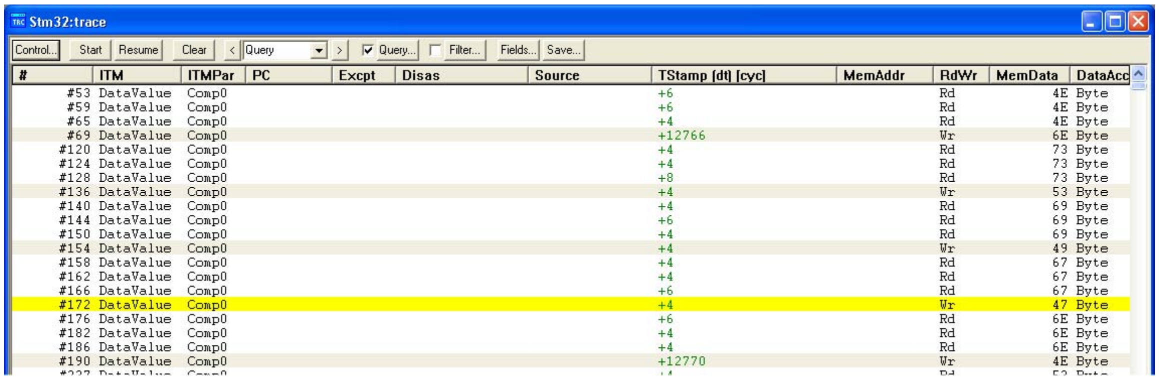
Program Counter (PC) Tracing Mode
[✓ Disable PC Trace]
Capture Complete PC Trace

These DWT settings — 2 address bits are ignored — you can trace a 4 byte array:



Comparator 0 Output	Function	Access	Value	Ignore Bits
ITM Trace	Emit Data on Data Address Match	Read or Write	&Buf1[0]	2

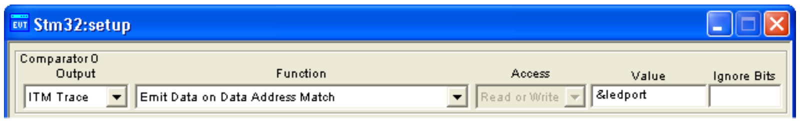
These settings may produce the following trace display. Note the tagged writes:



#	ITM	ITMPar	PC	Excpt	Disas	Source	TStamp [dt] [cyc]	MemAddr	RdWr	MemData	DataAcc
#53	DataValue	Comp0					+6		Rd	4E	Byte
#59	DataValue	Comp0					+6		Rd	4E	Byte
#65	DataValue	Comp0					+4		Rd	4E	Byte
#69	DataValue	Comp0					+12766		Wr	6E	Byte
#120	DataValue	Comp0					+4		Rd	73	Byte
#124	DataValue	Comp0					+4		Rd	73	Byte
#128	DataValue	Comp0					+8		Rd	73	Byte
#136	DataValue	Comp0					+4		Wr	53	Byte
#140	DataValue	Comp0					+4		Rd	69	Byte
#144	DataValue	Comp0					+6		Rd	69	Byte
#150	DataValue	Comp0					+4		Rd	69	Byte
#154	DataValue	Comp0					+4		Wr	49	Byte
#158	DataValue	Comp0					+4		Rd	67	Byte
#162	DataValue	Comp0					+4		Rd	67	Byte
#166	DataValue	Comp0					+6		Rd	67	Byte
#172	DataValue	Comp0					+4		Wr	47	Byte
#176	DataValue	Comp0					+6		Rd	6E	Byte
#182	DataValue	Comp0					+4		Rd	6E	Byte
#186	DataValue	Comp0					+4		Rd	6E	Byte
#190	DataValue	Comp0					+12770		Wr	4E	Byte

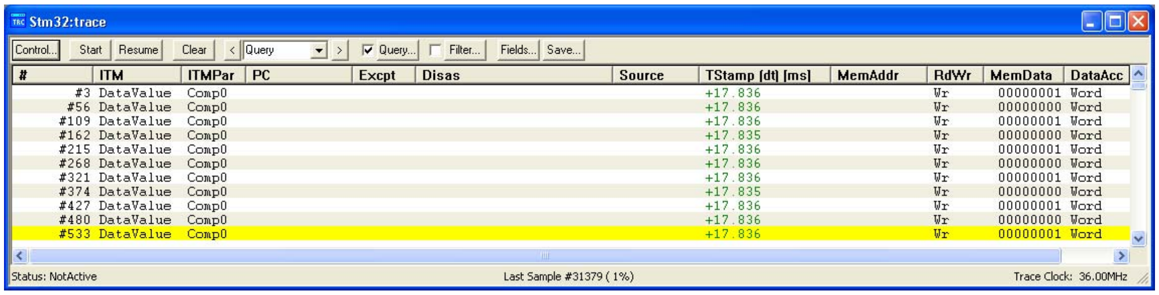
In our example, **Buf1** has been accessed frequently, however, if data only tracing traces some location that happens to be accessed rarely, the trace may accumulate slowly. You may be able to see the **Last Sample** index and the % of the trace buffer used changing in the Trace window status bar.

Consider the following settings that trace accesses to the **ledport** global variable.



Comparator 0 Output	Function	Access	Value	Ignore Bits
ITM Trace	Emit Data on Data Address Match	Read or Write	&ledport	

Here is the resulting trace:

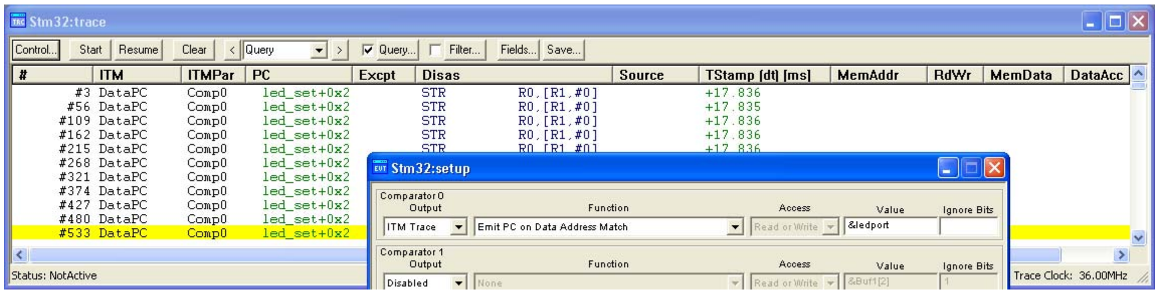


```
Stm32:trace
Control...  Start  Resume  Clear  < Query > ✓ Query...  Filter...  Fields...  Save...
#     ITM        ITMPar  PC   Excpt  Disas   Source  TStamp [dt] [ms]  MemAddr  RdWr  MemData   DataAcc
   #3 DataValue  Comp0                               +17.836                    Wr    00000001  Word
  #56 DataValue  Comp0                               +17.836                    Wr    00000000  Word
 #109 DataValue  Comp0                               +17.836                    Wr    00000001  Word
 #162 DataValue  Comp0                               +17.835                    Wr    00000000  Word
 #215 DataValue  Comp0                               +17.836                    Wr    00000001  Word
 #268 DataValue  Comp0                               +17.836                    Wr    00000000  Word
 #321 DataValue  Comp0                               +17.836                    Wr    00000001  Word
 #374 DataValue  Comp0                               +17.835                    Wr    00000000  Word
 #427 DataValue  Comp0                               +17.836                    Wr    00000001  Word
 #480 DataValue  Comp0                               +17.836                    Wr    00000000  Word
 #533 DataValue  Comp0                               +17.836                    Wr    00000001  Word
Status: NotActive                 Last Sample #31379 ( 1%)                Trace Clock:  36.00MHz
```

During the first 20 seconds of the program run, the trace filled only 1% of the total trace buffer depth! The trace shows that **ledset** is written to every 17.836 ms.

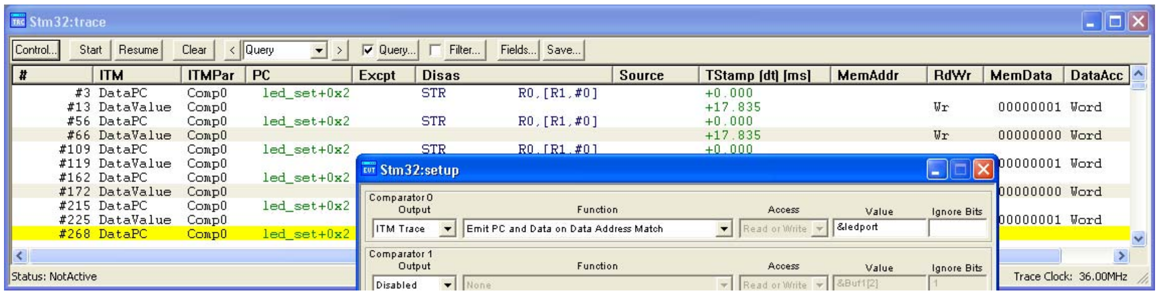
Adding PC to ITM Tracing

Data-only tracing cannot show from which location in your code the particular reads or writes originated. While at times this is acceptable, illegal accesses to certain locations may require you find out which function is responsible. To this end, it is possible to configure the DWT to output the PC where access took place. You can either setup the DWT to produce the PC without any data values by selecting **Emit PC as Data Address Match**, as follows.



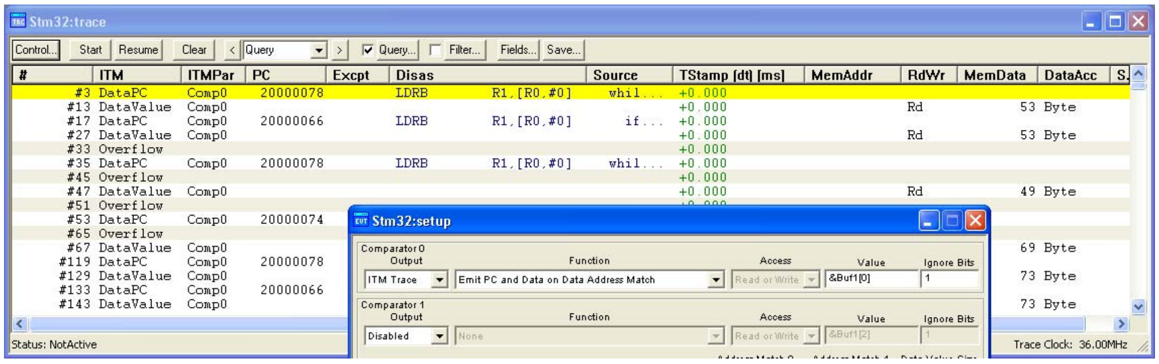
```
Stm32:trace
Control...  Start  Resume  Clear  < Query > ✓ Query...  Filter...  Fields...  Save...
#     ITM      ITMPar  PC            Excpt  Disas               Source  TStamp [dt] [ms]  MemAddr  RdWr  MemData  DataAcc
   #3 DataPC   Comp0   led_set+0x2          STR     R0,[R1,#0]          +17.836
  #56 DataPC   Comp0   led_set+0x2          STR     R0,[R1,#0]          +17.835
 #109 DataPC   Comp0   led_set+0x2          STR     R0,[R1,#0]          +17.836
 #162 DataPC   Comp0   led_set+0x2          STR     R0,[R1,#0]          +17.836
 #215 DataPC   Comp0   led_set+0x2          STR     R0,[R1,#0]          +17.836
 #268 DataPC   Comp0   led_set+0x2
 #321 DataPC   Comp0   led_set+0x2
 #374 DataPC   Comp0   led_set+0x2
 #427 DataPC   Comp0   led_set+0x2
 #480 DataPC   Comp0   led_set+0x2
 #533 DataPC   Comp0   led_set+0x2
Status: NotActive                                                       Trace Clock:  36.00MHz

Stm32:setup
Comparator 0
 Output      Function                          Access         Value      Ignore Bits
 ITM Trace   Emit PC on Data Address Match     Read or Write  &ledport
Comparator 1
 Output      Function                          Access         Value      Ignore Bits
 Disabled    None                              Read or Write  &Buf1[2]   1
```

Alternately, select **Emit PC and Data as Data Address Match** to make the DWT yield both the PC and the data value.



```
Stm32:trace
Control...  Start  Resume  Clear  < Query > ✓ Query...  Filter...  Fields...  Save...
#     ITM        ITMPar  PC            Excpt  Disas               Source  TStamp [dt] [ms]  MemAddr  RdWr  MemData   DataAcc
   #3 DataPC     Comp0   led_set+0x2          STR     R0,[R1,#0]          +0.000
  #13 DataValue  Comp0                                                    +17.835                    Wr    00000001  Word
  #56 DataPC     Comp0   led_set+0x2          STR     R0,[R1,#0]          +0.000
  #66 DataValue  Comp0                                                    +17.835                    Wr    00000000  Word
 #109 DataPC     Comp0   led_set+0x2          STR     R0,[R1,#0]          +0.000
 #119 DataValue  Comp0                                                                                     00000001  Word
 #162 DataPC     Comp0   led_set+0x2
 #172 DataValue  Comp0                                                                                     00000000  Word
 #215 DataPC     Comp0   led_set+0x2
 #225 DataValue  Comp0                                                                                     00000001  Word
 #268 DataPC     Comp0   led_set+0x2
Status: NotActive                                                         Trace Clock:  36.00MHz

Stm32:setup
Comparator 0
 Output      Function                                Access         Value      Ignore Bits
 ITM Trace   Emit PC and Data on Data Address Match  Read or Write  &ledport
Comparator 1
 Output      Function                                Access         Value      Ignore Bits
 Disabled    None                                    Read or Write  &Buf1[2]   1
```

Caution is advised, however. Using the 32-bit PC in the ITM may easily make the ITM FIFO to overflow, as illustrated in the example below.



Stm32:trace

Status: NotActive — Trace Clock: 36.00MHz

#	ITM	ITMPar	PC	Excpt	Disas	Source	TStamp [dt] [ms]	MemAddr	RdWr	MemData	DataAcc
#3	DataPC	Comp0	20000078		LDRB R1,[R0,#0]	whil...	+0.000				
#13	DataValue	Comp0					+0.000		Rd	53	Byte
#17	DataPC	Comp0	20000066		LDRB R1,[R0,#0]	if...	+0.000				
#27	DataValue	Comp0					+0.000		Rd	53	Byte
#33	Overflow						+0.000				
#35	DataPC	Comp0	20000078		LDRB R1,[R0,#0]	whil...	+0.000				
#45	Overflow						+0.000				
#47	DataValue	Comp0					+0.000		Rd	49	Byte
#51	Overflow										
#53	DataPC	Comp0	20000074								
#65	Overflow										
#67	DataValue	Comp0								69	Byte
#119	DataPC	Comp0	20000078								
#129	DataValue	Comp0								73	Byte
#133	DataPC	Comp0	20000066								
#143	DataValue	Comp0								73	Byte

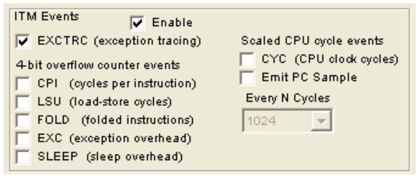
Stm32:setup

	Output	Function	Access	Value	Ignore Bits
Comparator 0	ITM Trace	Emit PC and Data on Data Address Match	Read or Write	&Buf1[0]	1
Comparator 1	Disabled	None	Read or Write	&Buf1[2]	1

Tracing Exceptions (Interrupts)

Tracing Exceptions Using ITM Events

The DWT/ITM module may also be configured to trace exceptions using the **Complex Events** window.



ITM Events [✓ Enable]

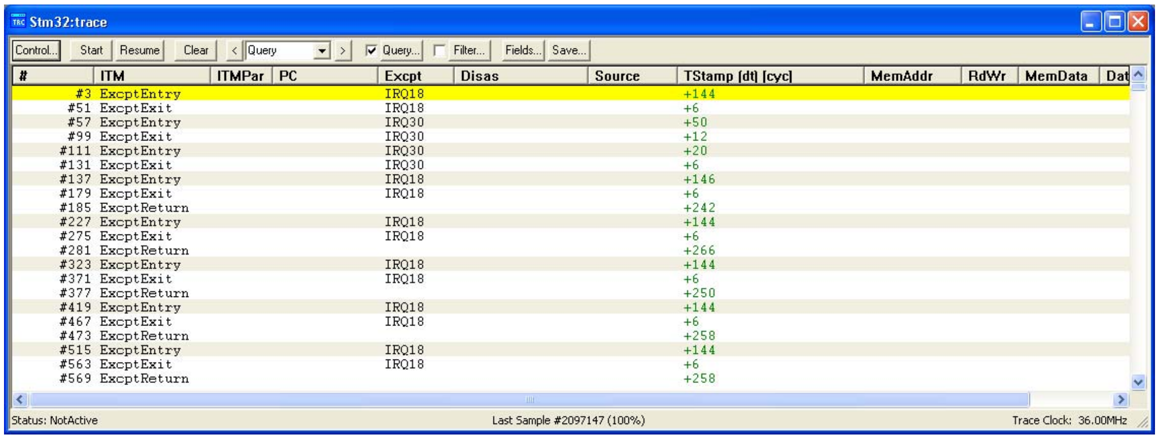
[✓] EXCTRC (exception tracing)

4-bit overflow counter events
- [ ] CPI (cycles per instruction)
- [ ] LSU (load-store cycles)
- [ ] FOLD (folded instructions)
- [ ] EXC (exception overhead)
- [ ] SLEEP (sleep overhead)

Scaled CPU cycle events
- [ ] CYC (CPU clock cycles)
- [ ] Emit PC Sample

Every N Cycles: 1024

The following trace shows several interrupts running on the system with the **Blinky** application running on Keil's MCBSTM32E board. The Interrupt **IRQ18** is generated by a DMA channel used by the A/D converter. The exception **IRQ30** is a timer interrupt.



Stm32:trace

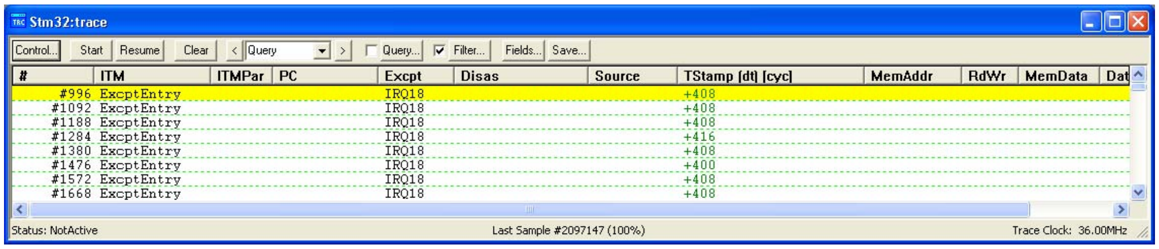
Status: NotActive — Last Sample #2097147 (100%) — Trace Clock: 36.00MHz

#	ITM	ITMPar	PC	Excpt	Disas	Source	TStamp [dt] [cyc]	MemAddr	RdWr	MemData
#3	ExcptEntry			IRQ18			+144			
#51	ExcptExit			IRQ18			+6			
#57	ExcptEntry			IRQ30			+50			
#99	ExcptExit			IRQ30			+12			
#111	ExcptEntry			IRQ30			+20			
#131	ExcptExit			IRQ30			+6			
#137	ExcptEntry			IRQ18			+146			
#179	ExcptExit			IRQ18			+6			
#185	ExcptReturn						+242			
#227	ExcptEntry			IRQ18			+144			
#275	ExcptExit			IRQ18			+6			
#281	ExcptReturn						+266			
#323	ExcptEntry			IRQ18			+144			
#371	ExcptExit			IRQ18			+6			
#377	ExcptReturn						+250			
#419	ExcptEntry			IRQ18			+144			
#467	ExcptExit			IRQ18			+6			
#473	ExcptReturn						+258			
#515	ExcptEntry			IRQ18			+144			
#563	ExcptExit			IRQ18			+6			
#569	ExcptReturn						+258			

The **ExcptEntry** ITM packets are generated on entry to an exception and are displayed as tagged. Each **ExcptExit** packet is generated upon the exit from an exception. The presence of the **ExcptReturn** packet signifies that code returned from all pending exceptions to the main application code.

There can be several **ExcptEntry/ExcptExit** pairs (without **ExcptReturn**) at the beginning of the trace. This is because the application was stopped with interrupts enabled, and some IRQs got registered as pending and were handled as soon as the code was started. After that, the exception **IRQ18** is generated with constant intervals. The **TStamp** field is in the delta mode, enabling us to determine that the code spends 258 clocks from **ExcptReturn** to **ExcptEntry** in a background task. Similarly, it handles IRQ18 in the 150 clocks, comprising the 144 clocks from **ExcptEntry** to **ExcptExit** and 6 from **ExcptExit** to **ExcptReturn**.

A trace post-filtered so that it shows only the **ExcptEntry** samples, might look like this.

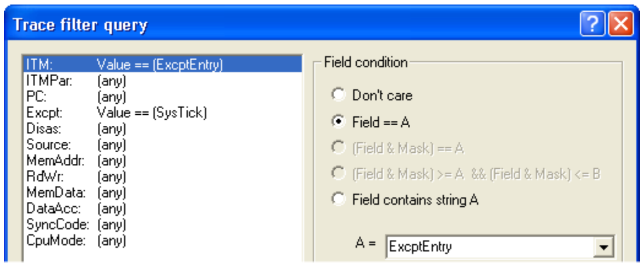


#	ITM	ITMPar	PC	Excpt	Disas	Source	TStamp [dt] [cyc]	MemAddr	RdWr	MemData	Dat
#996	ExcptEntry			IRQ18			+408				
#1092	ExcptEntry			IRQ18			+408				
#1188	ExcptEntry			IRQ18			+408				
#1284	ExcptEntry			IRQ18			+416				
#1380	ExcptEntry			IRQ18			+408				
#1476	ExcptEntry			IRQ18			+400				
#1572	ExcptEntry			IRQ18			+408				
#1668	ExcptEntry			IRQ18			+408				

Status: NotActive   Last Sample #2097147 (100%)   Trace Clock: 36.00MHz

The delta mode timestamp shows **IRQ18** being generated every 408 cycles.

Consider a more complex post-filtering.

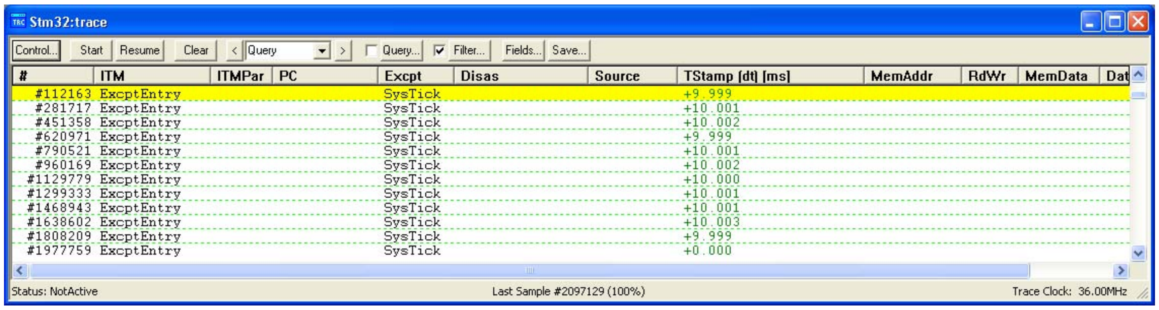


ITM:	Value == (ExcptEntry)
ITMPar:	(any)
PC:	(any)
Excpt:	Value == (SysTick)
Disas:	(any)
Source:	(any)
MemAddr:	(any)
RdWr:	(any)
MemData:	(any)
DataAcc:	(any)
SyncCode:	(any)
CpuMode:	(any)

Field condition: [✓ Field == A]

A = ExcptEntry

With the above settings, we can find all **SysTick** exceptions recorded in the trace buffer.



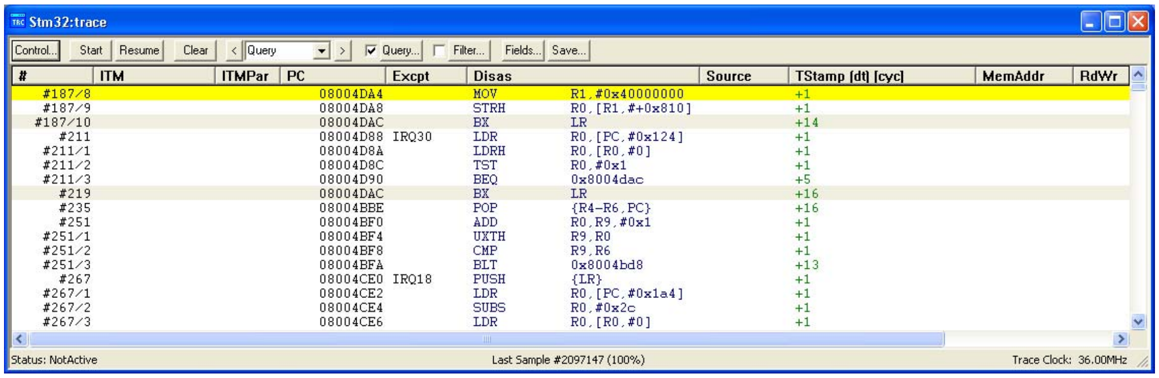
#	ITM	ITMPar	PC	Excpt	Disas	Source	TStamp [dt] [ms]	MemAddr	RdWr	MemData	Dat
#112163	ExcptEntry			SysTick			+9.999				
#281717	ExcptEntry			SysTick			+10.001				
#451358	ExcptEntry			SysTick			+10.002				
#620971	ExcptEntry			SysTick			+9.999				
#790521	ExcptEntry			SysTick			+10.001				
#960169	ExcptEntry			SysTick			+10.002				
#1129779	ExcptEntry			SysTick			+10.000				
#1299333	ExcptEntry			SysTick			+10.001				
#1468943	ExcptEntry			SysTick			+10.001				
#1638602	ExcptEntry			SysTick			+10.003				
#1808209	ExcptEntry			SysTick			+9.999				
#1977759	ExcptEntry			SysTick			+0.000				

Status: NotActive   Last Sample #2097129 (100%)   Trace Clock: 36.00MHz

Here the **SysTick** exception is generated about every 10ms, as indicated by the timestamp.

Tracing Exceptions in PC Trace

Exceptions are also shown in the **Excpt** field in the course of "normal" PC tracing without any extra settings.


Stm32:trace

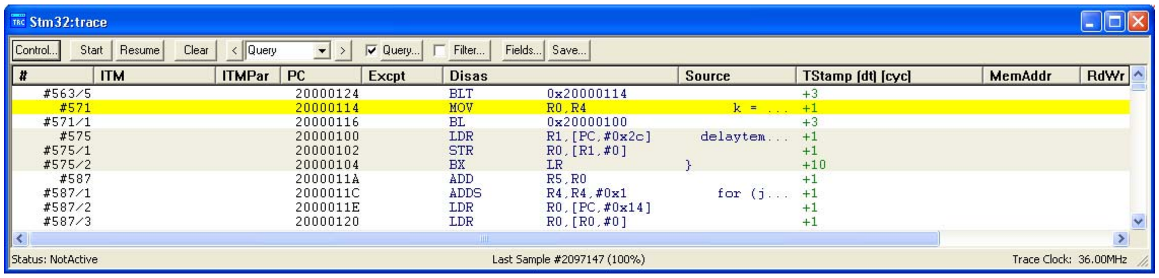
#	ITM	ITMPar	PC	Excpt	Disas		Source	TStamp (dt) [cyc]	MemAddr	RdWr
#187/8			08004DA4		MOV	R1,#0x40000000		+1		
#187/9			08004DA8		STRH	R0,[R1,#+0x810]		+1		
#187/10			08004DAC		BX	LR		+14		
#211			08004D88	IRQ30	LDR	R0,[PC,#0x124]		+1		
#211/1			08004D8A		LDRH	R0,[R0,#0]		+1		
#211/2			08004D8C		TST	R0,#0x1		+1		
#211/3			08004D90		BEQ	0x8004dac		+5		
#219			08004DAC		BX	LR		+16		
#235			08004BBE		POP	{R4-R6,PC}		+16		
#251			08004BF0		ADD	R0,R9,#0x1		+1		
#251/1			08004BF4		UXTH	R9,R0		+1		
#251/2			08004BF8		CMP	R9,R6		+1		
#251/3			08004BFA		BLT	0x8004bd8		+13		
#267			08004CE0	IRQ18	PUSH	{LR}		+1		
#267/1			08004CE2		LDR	R0,[PC,#0x1a4]		+1		
#267/2			08004CE4		SUBS	R0,#0x2c		+1		
#267/3			08004CE6		LDR	R0,[R0,#0]		+1		

Status: NotActive Last Sample #2097147 (100%) Trace Clock: 36.00MHz

It is noteworthy that only the exception entry is identified in the **Except** field. Instructions which exit exceptions handlers are simply functions returns, shown in gray. This particular trace exposes an unusually frequent sequence of exceptions, as the processor was stopped and many exceptions where pending. Timer **IRQ30** terminates in sample **187/10** and it is immediately followed by another timer **IRQ30** exception. This interrupt handler is very short, taking only 5 instructions. It reads one register and lets the program return to the main application in sample **235**. After just 5 instructions executed from the main program (samples **235** through **251/3**), the handling of the DMA exception **IRQ18** starts in sample **267**.

Understanding the Timestamp

Timestamp recorded by the JTAGjet-Trace is stored for every ETM sample. JTAGjet-Trace records the time for every trace sample that is arrives at the ETM connector. Cortex-M3, which contains version 3.4 of the ETM module, is capable of sending one trace sample to trace up to 16 instructions. Therefore, the timing of these instructions cannot be precisely known, and the timestamp for each individual instruction in the trace packet must be estimated.

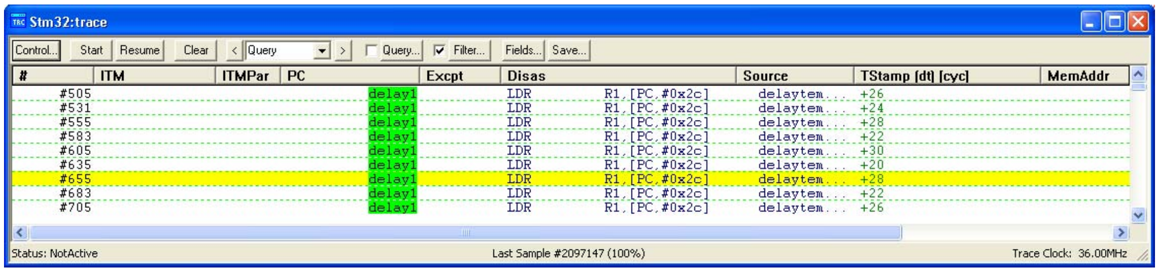

Stm32:trace

#	ITM	ITMPar	PC	Excpt	Disas		Source	TStamp (dt) [cyc]	MemAddr	RdWr
#563/5			20000124		BLT	0x20000114		+3		
#571			20000114		MOV	R0,R4	k = ...	+1		
#571/1			20000116		BL	0x20000100		+3		
#575			20000100		LDR	R1,[PC,#0x2c]	delaytem...	+1		
#575/1			20000102		STR	R0,[R1,#0]		+1		
#575/2			20000104		BX	LR	}	+10		
#587			2000011A		ADD	R5,R0		+1		
#587/1			2000011C		ADDS	R4,R4,#0x1	for (j...	+1		
#587/2			2000011E		LDR	R0,[PC,#0x14]		+1		
#587/3			20000120		LDR	R0,[R0,#0]		+1		

Status: NotActive Last Sample #2097147 (100%) Trace Clock: 36.00MHz

In the illustration above, all) instructions tagged with gray were generated from a single physical sample **575**. The timestamp estimate indicates that the first two instructions, **LDR** and **STR**, were executed in 1 cycle each, while the third instruction, **BL**, took 10 cycles. In reality, this may not be exactly the case. All we know for sure is that these three instructions were executed in the total of 12 cycles.

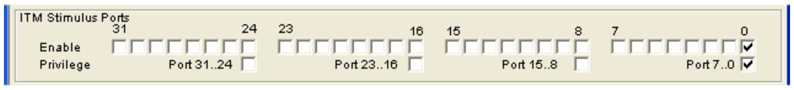
In general, the timestamp cannot be relied upon for short time measurements. This is why the timing between calls to the **delay1** function in a loop, as shown in the figure below, contains values ranging from **+20** to **+30 cycles**. For much longer function runs, the accuracy of +/-16 cycles will not be that important.



```
Stm32:trace
Control...  Start  Resume  Clear  < Query > [ ] Query...  [✓] Filter...  Fields...  Save...
#        ITM      ITMPar  PC              Excpt  Disas                          Source         TStamp [dt] [cyc]   MemAddr
    #505                                  delay1        LDR        R1,[PC,#0x2c]   delaytem...  +26
    #531                                  delay1        LDR        R1,[PC,#0x2c]   delaytem...  +24
    #555                                  delay1        LDR        R1,[PC,#0x2c]   delaytem...  +28
    #583                                  delay1        LDR        R1,[PC,#0x2c]   delaytem...  +22
    #605                                  delay1        LDR        R1,[PC,#0x2c]   delaytem...  +30
    #635                                  delay1        LDR        R1,[PC,#0x2c]   delaytem...  +20
    #655                                  delay1        LDR        R1,[PC,#0x2c]   delaytem...  +28
    #683                                  delay1        LDR        R1,[PC,#0x2c]   delaytem...  +22
    #705                                  delay1        LDR        R1,[PC,#0x2c]   delaytem...  +26
Status: NotActive                 Last Sample #2097147 (100%)                  Trace Clock: 36.00MHz
```

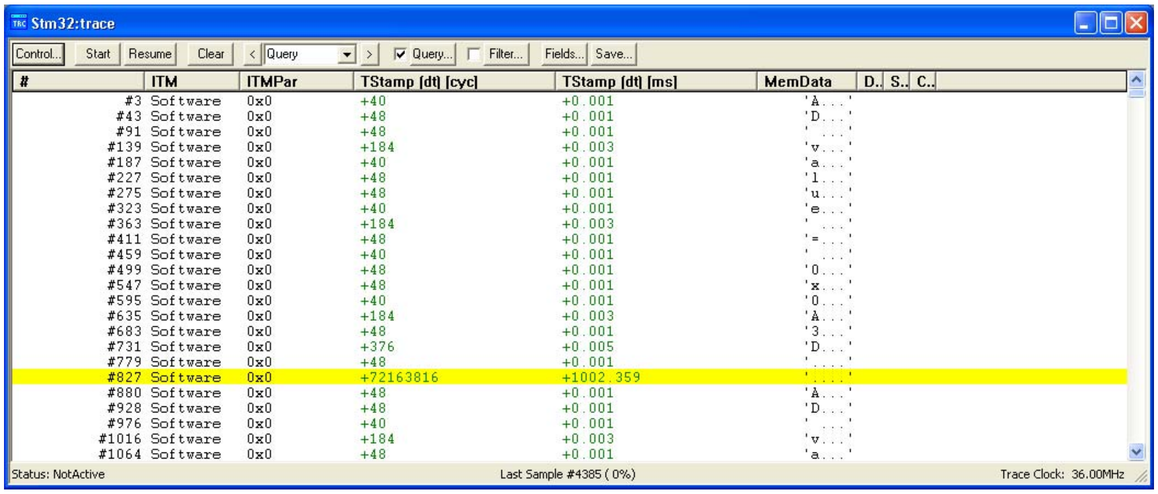
Application Generated ITM Trace

The Cortex-M3 ITM module includes 32 ITM Stimulus Port Registers at addresses **0xE0000000 – 0xE000007C**. By writing to these registers from application, it is possible to generate custom trace records.



```
ITM Stimulus Ports
           31          24  23          16  15           8  7            0
Enable     [ ][ ][ ][ ][ ][ ][ ][ ]  [ ][ ][ ][ ][ ][ ][ ][ ]  [ ][ ][ ][ ][ ][ ][ ][ ]  [ ][ ][ ][ ][ ][ ][ ][✓]
Privilege        Port 31..24 [ ]        Port 23..16 [ ]        Port 15..8 [ ]        Port 7..0 [✓]
```

If the ITM module is set by the user to enable the **ITM Stimulus Ports** with **Port 0** enabled as shown in the figure above, the trace may look like this.



```
Stm32:trace
Control...  Start  Resume  Clear  < Query > [✓] Query...  [ ] Filter...  Fields...  Save...
#            ITM       ITMPar  TStamp [dt] [cyc]  TStamp [dt] [ms]  MemData  D.. S.. C..
       #3 Software  0x0     +40              +0.001            'A...'
      #43 Software  0x0     +48              +0.001            'D...'
      #91 Software  0x0     +48              +0.001            ' ...'
     #139 Software  0x0     +184             +0.003            'v...'
     #187 Software  0x0     +40              +0.001            'a...'
     #227 Software  0x0     +48              +0.001            'l...'
     #275 Software  0x0     +48              +0.001            'u...'
     #323 Software  0x0     +40              +0.001            'e...'
     #363 Software  0x0     +184             +0.003            ' ...'
     #411 Software  0x0     +48              +0.001            '=...'
     #459 Software  0x0     +40              +0.001            ' ...'
     #499 Software  0x0     +48              +0.001            '0...'
     #547 Software  0x0     +48              +0.001            'x...'
     #595 Software  0x0     +40              +0.001            '0...'
     #635 Software  0x0     +184             +0.003            'A...'
     #683 Software  0x0     +48              +0.001            '3...'
     #731 Software  0x0     +376             +0.005            'D...'
     #779 Software  0x0     +48              +0.001            '....'
     #827 Software  0x0     +72163816        +1002.359         '....'
     #880 Software  0x0     +48              +0.001            'A...'
     #928 Software  0x0     +48              +0.001            'D...'
     #976 Software  0x0     +40              +0.001            ' ...'
    #1016 Software  0x0     +184             +0.003            'v...'
    #1064 Software  0x0     +48              +0.001            'a...'
Status: NotActive                 Last Sample #4385 ( 0%)                  Trace Clock: 36.00MHz
```

The **Software** entry in the **ITM** field denotes a trace record generated by a software write to the Stimulus Port. The **ITMPar** shows the stimulus port index. The trace displays the text **"AD value = 0xABCD"** every second. The **TStamp** column displays timestamps both in cycles and ms.

The code to generate such trace, based on an example from Keil's µV3 with extra comments added to it, looks as follows.

```
#define ITM_Port8(n)    (*((volatile unsigned char *)(0xE0000000+4*(n))))
#define ITM_Port32(n)   (*((volatile unsigned long *)(0xE0000000+4*(n))))
#define DEMCR           (*((volatile unsigned long *)(0xE000EDFC)))
#define TRCENA          0x01000000
int SendChar (int ch)  {       /* Write character via ITM Stimulus Port 0 */
    if (DEMCR & TRCENA) {      /* If TRACE enabled. Otherwise, nothing */
        while (ITM_Port32(0) == 0);  /* Wait till ITM sends prev. data */
        ITM_Port8(0) = ch;         /* Write char to ITM Stimulus Port 0 */
    }
    return (ch);
}
```

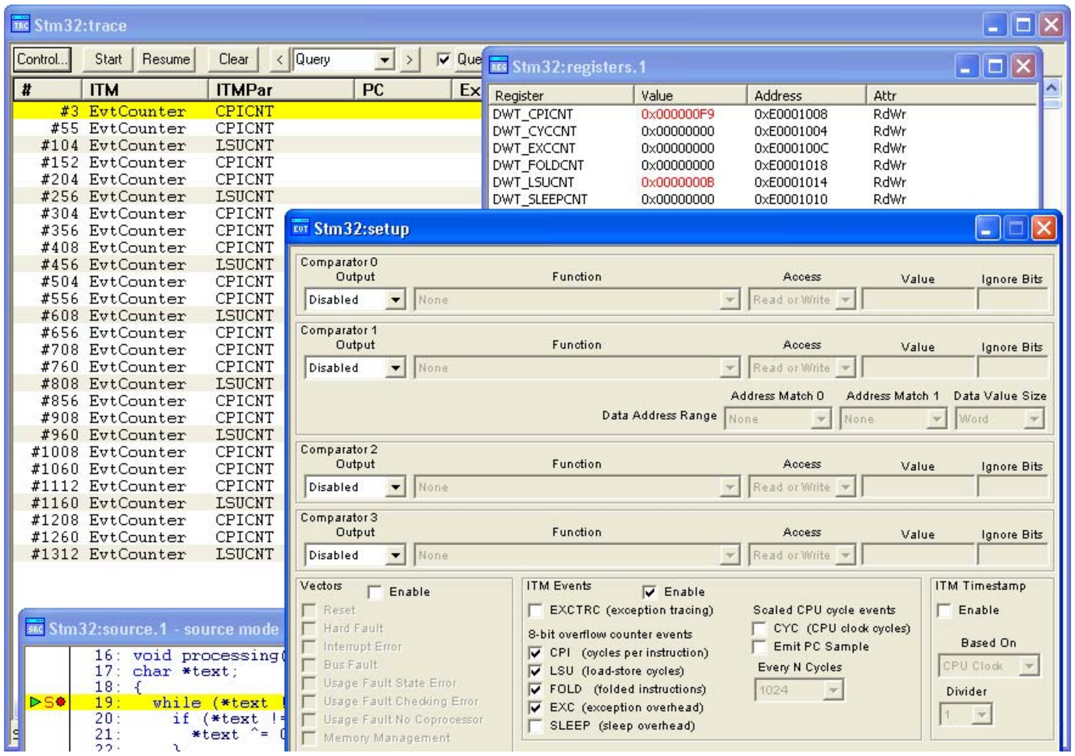
Note: The current release of the JTAGjet-Trace software does not provide a convenient way of extracting data from different ITM Stimulus Ports into separate files. Also, Port 0 is used for printing, while text should be shown in a standard console window, not in a single column. Currently, the program only allows you to save the trace to a text file and extract all ITM data from there if needed.

DWT and ITM Profiling Related Features

This chapter describes some additional DWT and ITM trace features designed to aid code profiling. Future versions of the JTAGjet-Trace software will provide improved display and post-processing of these trace records. Note that JTAGjet-Trace's internal timestamp can be used to profile the code.

Tracing DWT Counters

The Cortex-M3 DWT module has several internal 8-bit counters to count various instruction-execution related system events occurring when the CPU is running. If such an 8-bit counter overflows, it may generate a DWT trace record. The figure below shows such counters being collected in the trace. Since PC Tracing is turned off, only the DWT trace counters are shown.



Stm32:trace

#	ITM	ITMPar	PC	Ex
#3	EvtCounter	CPICNT		
#55	EvtCounter	CPICNT		
#104	EvtCounter	LSUCNT		
#152	EvtCounter	CPICNT		
#204	EvtCounter	CPICNT		
#256	EvtCounter	LSUCNT		
#304	EvtCounter	CPICNT		
#356	EvtCounter	CPICNT		
#408	EvtCounter	CPICNT		
#456	EvtCounter	LSUCNT		
#504	EvtCounter	CPICNT		
#556	EvtCounter	CPICNT		
#608	EvtCounter	LSUCNT		
#656	EvtCounter	CPICNT		
#708	EvtCounter	CPICNT		
#760	EvtCounter	CPICNT		
#808	EvtCounter	LSUCNT		
#856	EvtCounter	CPICNT		
#908	EvtCounter	CPICNT		
#960	EvtCounter	LSUCNT		
#1008	EvtCounter	CPICNT		
#1060	EvtCounter	CPICNT		
#1112	EvtCounter	CPICNT		
#1160	EvtCounter	LSUCNT		
#1208	EvtCounter	CPICNT		
#1260	EvtCounter	CPICNT		
#1312	EvtCounter	LSUCNT		

Stm32:registers.1

Register	Value	Address	Attr
DWT_CPICNT	0x000000F9	0xE0001008	RdWr
DWT_CYCCNT	0x00000000	0xE0001004	RdWr
DWT_EXCCNT	0x00000000	0xE000100C	RdWr
DWT_FOLDCNT	0x00000000	0xE0001018	RdWr
DWT_LSUCNT	0x0000000B	0xE0001014	RdWr
DWT_SLEEPCNT	0x00000000	0xE0001010	RdWr

Stm32:source.1 - source mode

```
void processing(
char *text;
{
  while (*text !
    if (*text !=
      *text ^= (
    }
```

Stm32:setup

Comparator	Output	Function	Access	Value	Ignore Bits
Comparator 0	Disabled	None	Read or Write		
Comparator 1	Disabled	None	Read or Write		
Comparator 2	Disabled	None	Read or Write		
Comparator 3	Disabled	None	Read or Write		

Data Address Range: Address Match 0: None; Address Match 1: None; Data Value Size: Word

Vectors: ☐ Enable
☐ Reset
☐ Hard Fault
☐ Interrupt Error
☐ Bus Fault
☐ Usage Fault State Error
☐ Usage Fault Checking Error
☐ Usage Fault No Coprocessor
☐ Memory Management

ITM Events: ☑ Enable
☐ EXCTRC (exception tracing)
8-bit overflow counter events
☑ CPI (cycles per instruction)
☑ LSU (load-store cycles)
☑ FOLD (folded instructions)
☑ EXC (exception overhead)
☐ SLEEP (sleep overhead)
Scaled CPU cycle events
☐ CYC (CPU clock cycles)
☐ Emit PC Sample
Every N Cycles: 1024

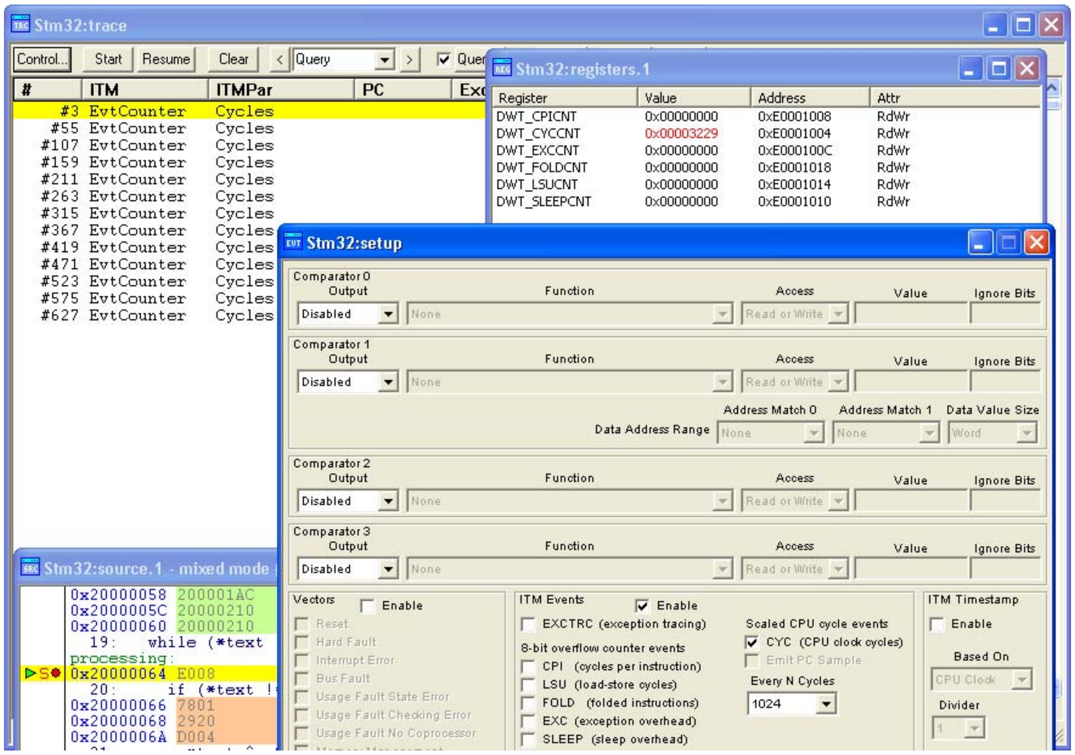
ITM Timestamp: ☐ Enable
Based On: CPU Clock
Divider: 1

The code was executing from breakpoint to breakpoint in the **processing** function. As you can see, there are 8 LSUCNT trace records tagged in the trace with gray color. The current value of the DWT_LSUCNT register is 0xB, which tells us that during the execution of this code, there were 8*256 + 0xB = 2059 LSU cycles, the additional cycles caused by the load-store instructions. Similar calculations can be made for the CPI events. The meaning of the DWT overflow counter events is as follows.

DWT COUNTER	DESCRIPTION
LSUCNT	Additional cycles required by load/store instructions.
CPICNT	Additional cycles requires by all other instructions.
FOLDCNT	Number of folded instructions. (0 cycles required.)
EXCCNT	Additional cycles required due to exception entry and exit.
SLEEPCNT	Number of CPU cycles when CPU is in sleep mode.

Tracing CPU Clock Cycles Counter

The setup needed for tracing the CPU clock cycles counter looks like this.



Stm32:trace

Control... Start Resume Clear < Query > ☑ Quer

```
#       ITM          ITMPar     PC     Exc
   #3  EvtCounter   Cycles
  #55  EvtCounter   Cycles
 #107  EvtCounter   Cycles
 #159  EvtCounter   Cycles
 #211  EvtCounter   Cycles
 #263  EvtCounter   Cycles
 #315  EvtCounter   Cycles
 #367  EvtCounter   Cycles
 #419  EvtCounter   Cycles
 #471  EvtCounter   Cycles
 #523  EvtCounter   Cycles
 #575  EvtCounter   Cycles
 #627  EvtCounter   Cycles
```

Stm32:registers.1

Register	Value	Address	Attr
DWT_CPICNT	0x00000000	0xE0001008	RdWr
DWT_CYCCNT	0x00003229	0xE0001004	RdWr
DWT_EXCCNT	0x00000000	0xE000100C	RdWr
DWT_FOLDCNT	0x00000000	0xE0001018	RdWr
DWT_LSUCNT	0x00000000	0xE0001014	RdWr
DWT_SLEEPCNT	0x00000000	0xE0001010	RdWr

Stm32:source.1 - mixed mode

```
0x20000058 200001AC
0x2000005C 20000210
0x20000060 20000210
    19:    while (*text
processing:
0x20000064 E008
    20:      if (*text !
0x20000066 7801
0x20000068 2920
0x2000006A D004
```

Stm32:setup

	Output	Function	Access	Value	Ignore Bits
Comparator 0	Disabled	None	Read or Write		
Comparator 1	Disabled	None	Read or Write		
Comparator 2	Disabled	None	Read or Write		
Comparator 3	Disabled	None	Read or Write		

Comparator 1: Data Address Range — Address Match 0: None; Address Match 1: None; Data Value Size: Word

Vectors: ☐ Enable
- ☐ Reset
- ☐ Hard Fault
- ☐ Interrupt Error
- ☐ Bus Fault
- ☐ Usage Fault State Error
- ☐ Usage Fault Checking Error
- ☐ Usage Fault No Coprocessor

ITM Events: ☑ Enable
- ☐ EXCTRC (exception tracing)
- 8-bit overflow counter events
- ☐ CPI (cycles per instruction)
- ☐ LSU (load-store cycles)
- ☐ FOLD (folded instructions)
- ☐ EXC (exception overhead)
- ☐ SLEEP (sleep overhead)

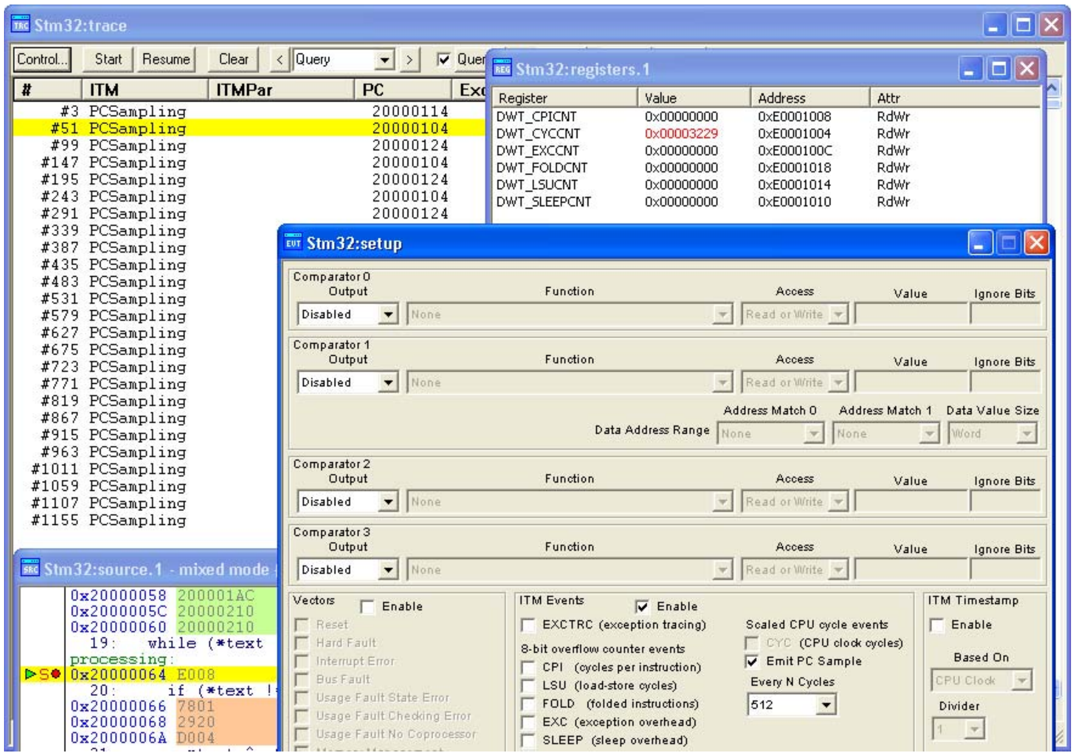
Scaled CPU cycle events
- ☑ CYC (CPU clock cycles)
- ☐ Emit PC Sample
- Every N Cycles: 1024

ITM Timestamp: ☐ Enable
- Based On: CPU Clock
- Divider: 1

The 32-bit DWT_CYCNT register holds the number of the CPU cycles. An ITM Trace record can be generated on every N CPU cycles, 1024 in our example. It is possible to setup the DWT to generate a CPU cycle counter ITM record every 64 to 16384 clocks by changing the **Every N Cycles** field in the **Complex Events** window. Since the JTAGjet-Trace emulator has an cycle accurate timestamp built-in, the CPU cycle counter is of little use in ETM tracing.

ITM PC Sampling

You may choose to generate a PC sampling record by selecting **Emit PC Sample** in the **Complex Events** window, as shown below.



Stm32:trace

Control... | Start | Resume | Clear | < | Query | > | Quer

#	ITM	ITMPar	PC	Exc
#3	PCSampling		20000114	
#51	PCSampling		20000104	
#99	PCSampling		20000124	
#147	PCSampling		20000104	
#195	PCSampling		20000124	
#243	PCSampling		20000104	
#291	PCSampling		20000124	
#339	PCSampling			
#387	PCSampling			
#435	PCSampling			
#483	PCSampling			
#531	PCSampling			
#579	PCSampling			
#627	PCSampling			
#675	PCSampling			
#723	PCSampling			
#771	PCSampling			
#819	PCSampling			
#867	PCSampling			
#915	PCSampling			
#963	PCSampling			
#1011	PCSampling			
#1059	PCSampling			
#1107	PCSampling			
#1155	PCSampling			

Stm32:registers.1

Register	Value	Address	Attr
DWT_CPICNT	0x00000000	0xE0001008	RdWr
DWT_CYCCNT	0x00003229	0xE0001004	RdWr
DWT_EXCCNT	0x00000000	0xE000100C	RdWr
DWT_FOLDCNT	0x00000000	0xE0001018	RdWr
DWT_LSUCNT	0x00000000	0xE0001014	RdWr
DWT_SLEEPCNT	0x00000000	0xE0001010	RdWr

Stm32:source.1 - mixed mode

```
0x20000058 200001AC
0x2000005C 20000210
0x20000060 20000210
  19:    while (*text
processing:
0x20000064 E008
  20:      if (*text !
0x20000066 7801
0x20000068 2920
0x2000006A D004
```

Stm32:setup

	Output	Function	Access	Value	Ignore Bits
Comparator 0	Disabled	None	Read or Write		
Comparator 1	Disabled	None	Read or Write		
Comparator 2	Disabled	None	Read or Write		
Comparator 3	Disabled	None	Read or Write		

Data Address Range: Address Match 0: None; Address Match 1: None; Data Value Size: Word

Vectors: Enable ☐
- ☐ Reset
- ☐ Hard Fault
- ☐ Interrupt Error
- ☐ Bus Fault
- ☐ Usage Fault State Error
- ☐ Usage Fault Checking Error
- ☐ Usage Fault No Coprocessor

ITM Events: Enable ☑
- ☐ EXCTRC (exception tracing)
- 8-bit overflow counter events
- ☐ CPI (cycles per instruction)
- ☐ LSU (load-store cycles)
- ☐ FOLD (folded instructions)
- ☐ EXC (exception overhead)
- ☐ SLEEP (sleep overhead)

Scaled CPU cycle events
- ☐ CYC (CPU clock cycles)
- ☑ Emit PC Sample
- Every N Cycles: 512

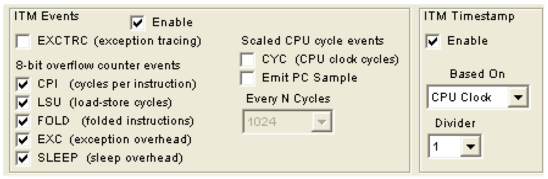
ITM Timestamp
- ☐ Enable
- Based On: CPU Clock
- Divider: 1

These settings will generate PC sample records in the ETM trace every specified number of CPU cycles, here 512. PC sampling may be used to aid statistical code profiling. **Code Coverage** and **Performance** profiling functionality is available in Chameleon only for conventional emulators that replace the CPU. In future implementations, both the **Code Coverage** module and **Performance** module will be enhanced to take advantage of the data available in the ETM trace.

The **Code Scope** window description found in the debugger contains useful PC statistics generated without the use of the ETM trace.

ITM Timestamp

The ITM Timestamp feature can be turned on from within the **Complex Events** window.

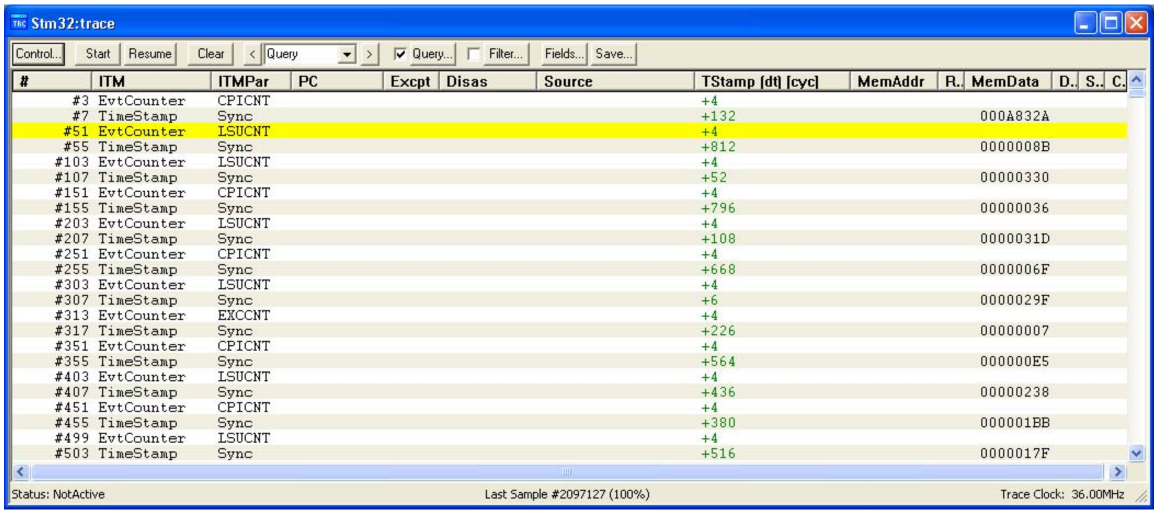


ITM Events: Enable ☑
- ☐ EXCTRC (exception tracing)
- 8-bit overflow counter events
- ☑ CPI (cycles per instruction)
- ☑ LSU (load-store cycles)
- ☑ FOLD (folded instructions)
- ☑ EXC (exception overhead)
- ☑ SLEEP (sleep overhead)

Scaled CPU cycle events
- ☐ CYC (CPU clock cycles)
- ☐ Emit PC Sample
- Every N Cycles: 1024

ITM Timestamp
- ☑ Enable
- Based On: CPU Clock
- Divider: 1

The ITM Timestamp option instructs the ITM to add a timestamp to every ITM record transmitted to the trace:



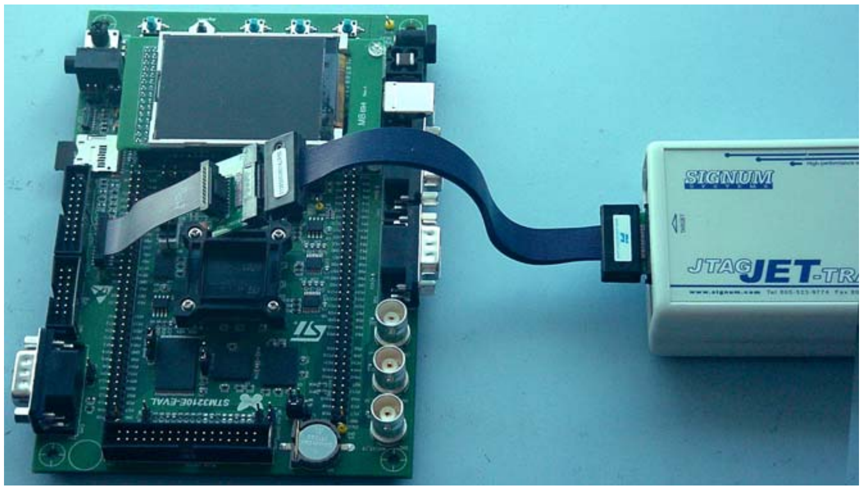
#	ITM	ITMPar	PC	Excpt	Disas	Source	TStamp (dt) [cyc]	MemAddr	R..	MemData	D..	S..	C..
#3	EvtCounter	CPICNT					+4						
#7	TimeStamp	Sync					+132			000A832A			
#51	EvtCounter	LSUCNT					+4						
#55	TimeStamp	Sync					+812			0000008B			
#103	EvtCounter	LSUCNT					+4						
#107	TimeStamp	Sync					+52			00000330			
#151	EvtCounter	CPICNT					+4						
#155	TimeStamp	Sync					+796			00000036			
#203	EvtCounter	LSUCNT					+4						
#207	TimeStamp	Sync					+108			0000031D			
#251	EvtCounter	CPICNT					+4						
#255	TimeStamp	Sync					+668			0000006F			
#303	EvtCounter	LSUCNT					+4						
#307	TimeStamp	Sync					+6			0000029F			
#313	EvtCounter	EXCCNT					+4						
#317	TimeStamp	Sync					+226			00000007			
#351	EvtCounter	CPICNT					+4						
#355	TimeStamp	Sync					+564			000000E5			
#403	EvtCounter	LSUCNT					+4						
#407	TimeStamp	Sync					+436			00000238			
#451	EvtCounter	CPICNT					+4						
#455	TimeStamp	Sync					+380			000001BB			
#499	EvtCounter	LSUCNT					+4						
#503	TimeStamp	Sync					+516			0000017F			

Status: NotActive   Last Sample #2097127 (100%)   Trace Clock: 36.00MHz

ITM **TimeStamp** records (tagged) with a timestamp show the precise number of cycles taken by an event. The feature allows a profiling program to accurately locate events in time. Since the JTAGjet-Trace provides independent hardware based cycle-accurate timestamp for every sample, the ITM based timestamp does not provide any additional benefits.

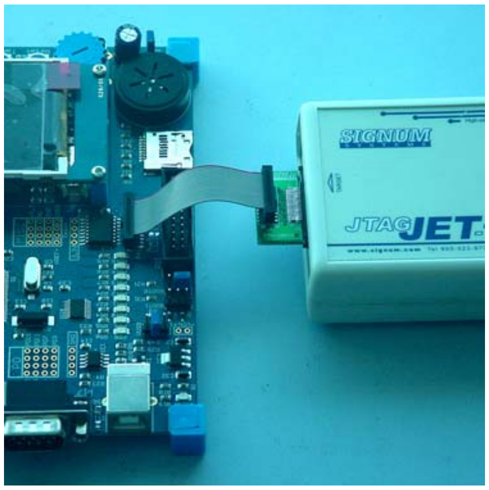
Appendix A – Connecting to Specific Boards

Connecting to ST STM3210E EVAL Board

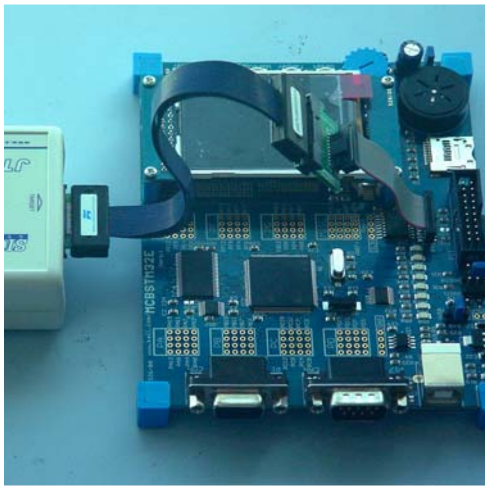


Connecting to Keil MCBSTM32E Board

Note: There are two versions of Keil's MCBSTM32E board designated as Vers. 1 and Vers. 2. These orientation of the 20-pin ETM connector on these boards is different, as shown in the illustrations.



Vers. 1



Vers. 2

Document Revision History

VERSION	DATE	COMMENTS
1.00	Oct 27 2008	Initial Release

❑

UM-Cortex-M3-ETM 11.10.08.17.35 770