Atmel®

SMART ARM-based Microcontrollers

Advanced Debugging with ETM for SAM V7/E7/S7 MCUs

APPLICATION NOTE

Introduction

The Embedded Trace Macrocell (ETM) is a real-time trace module providing instruction tracing of a processor. An ETM is an integral part of the ARM® debug solution, providing the user with a real-time debug solution.

The Atmel® | SMART SAM V7/E7/S7 microcontrollers implement the ETM for the instruction stream, via a 6-line Trace Port Interface Unit (TPIU).

This application note describes how to configure and use the ETM for these Atmel MCUs. It covers the following general aspects of the ETM:

- Hardware implementation
- Configuration
- Trace debug display

Table of Contents

1. Embedded Trace Macrocells (ETM)

1.1. Why Embedded Trace Macrocells?

To debug or optimize an embedded system, embedded developers have two options:

1. Conventional debug using JTAG interface—Setting breakpoints and/or watch points to halt the processing unit and, from there, use a debug connection to examine or modify register or memory and single-step to understand how the program works. While enabling developers to control the execution and debug their code, conventional debug has several disadvantages:
   – It is intrusive, as debug alters the behavior of the system.
   – The requirement to stop the processor may not be possible for some applications (e.g. hard-disk, automotive, etc.).
   – It cannot be used to debug software operating real-time.
   – It offers no visibility of software performance.
2. Real-time trace using Embedded Trace Macrocells (ETM)—When the system is running, trace macrocells collect instructions, compress this information and deliver off-chip in real-time or on-chip for post-processing.
   Trace is post-merged with source code in the development workstation for later analysis.

ETM provides a non-intrusive "real-time trace" and offers significant advantages over conventional debug tools. They enable developers to accelerate their product development and deliver high-quality optimized software by analyzing in real-time how the software operates on the platform:

- Trace the core at full speed with zero performance overhead
- Trace and debug your system executing in real-time
- Cycle-accurate trace
- Collect vital timing on program execution for performance optimizations and 'always-in-time' code verification
- Extensive sequential trigger logic & on-chip filtering conditions control which data is captured
- Capture program activity around the event you want to look at
- View trace over longer elapsed time
- More visibility requiring fewer pins or smaller trace buffers

ARM provides a comprehensive set of real-time trace macrocells:

- ARM processor ETM helps to observe how the software executes on the ARM processor(s).
- Instrumentation trace macrocells used for high level software instrumentation (print type debug, OS and application events).
- Bus trace macrocells make bus information visible that cannot be inferred from core trace.

2. ARM Cortex-M7 Processor with ETM

2.1. ARM Cortex-M7 Processor Block Diagram

The ARM® Cortex®-M7 processor is the high-end processor of the ARM Cortex-M family, providing 5 CoreMark/MHz and up to 630 DMIPS. The ARM Cortex-M7 processor is based on the ARMv7-M architecture and is binary compatible with other ARM Cortex -M processors.

The ARM Cortex-M7 is connected to the ETM which is connected to the TPIU.

Note: On Atmel | SMART SAM V7/E7/S7 microcontrollers, the ETM is implemented for instruction trace only.

Figure 2-1 ARM Cortex-M7 Processor Block Diagram

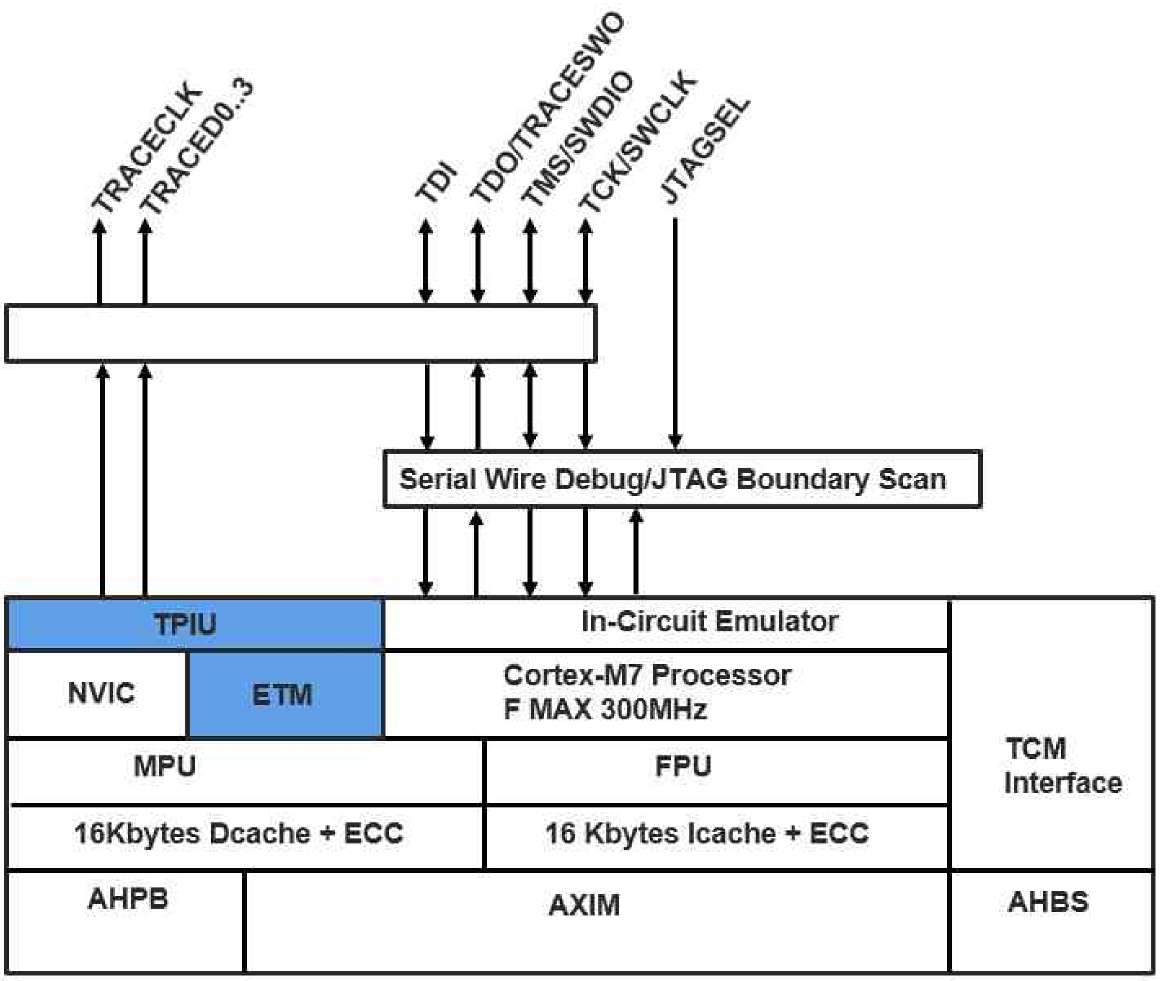
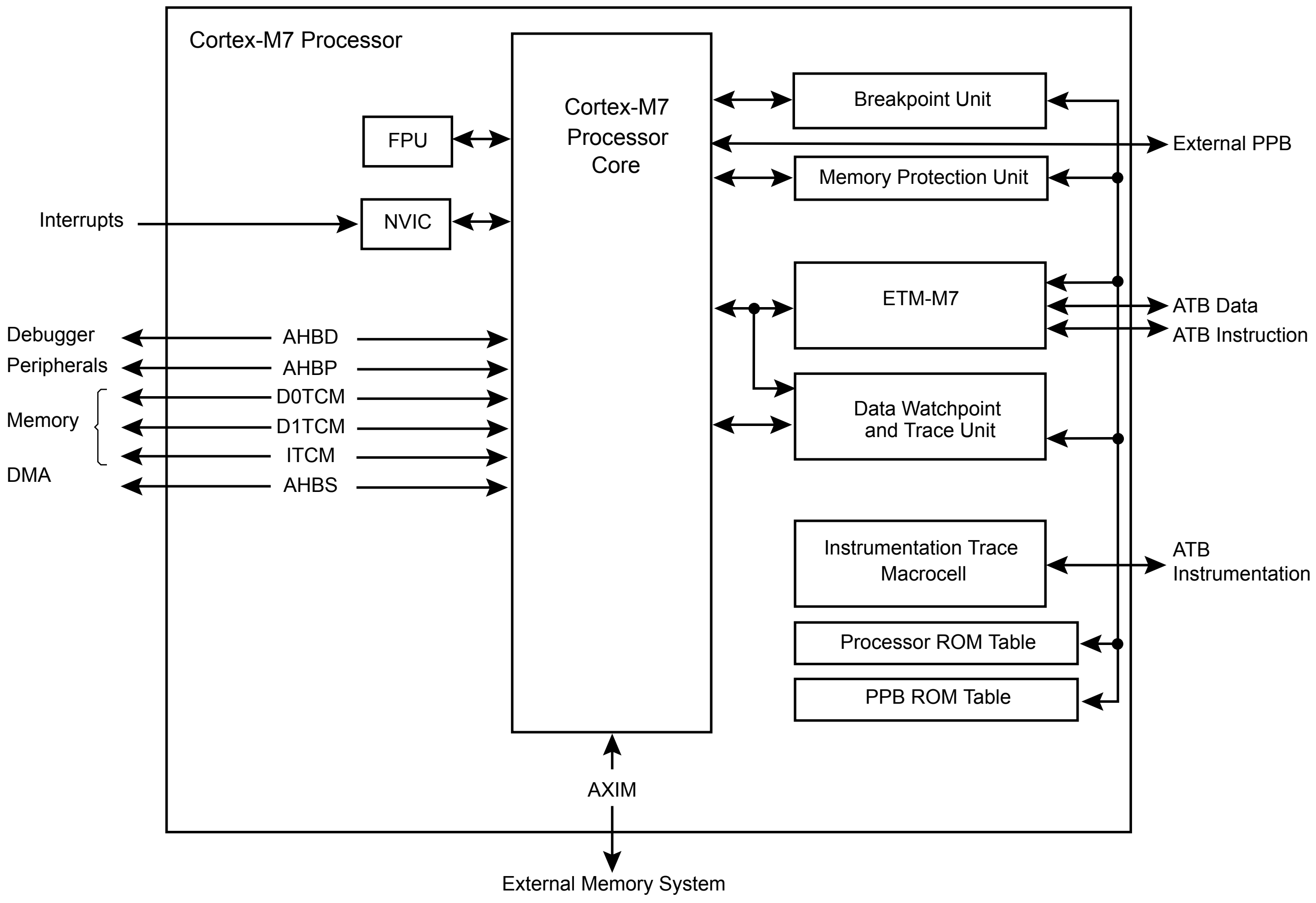


Figure 2-2 ARM Cortex-M7 Processor Top-Level Diagram



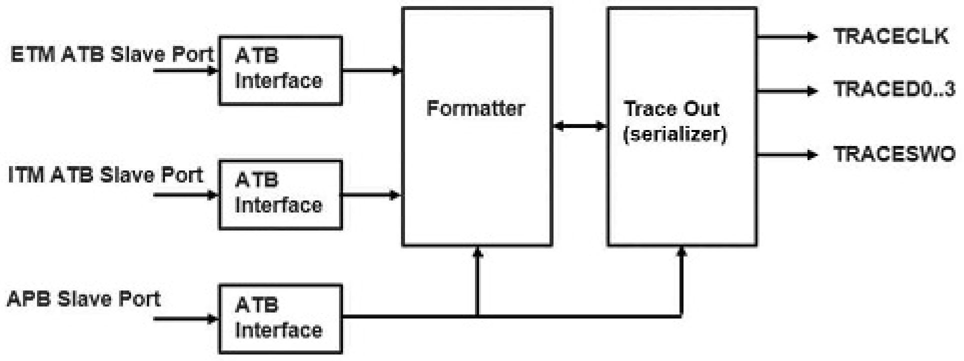
2.2. Trace Port Interface (TPIU)

The Trace Port Interface Unit (TPIU) acts as a bridge between the on-chip trace data from the ITM, the ETM and the external trace capture device.

The TPIU formats the data to be sent, and serializes the data to the different pins. The TPIU must be configured to use the ETM debug trace. The TPIU outputs trace data in a Serial Wire Output (SWO) format.

The ATB is a trace output bus used for debugging.

Figure 2-3 Trace Port Interface Unit (TPIU) Block Diagram



2.3. ETM Pins

The ETM uses the TPIU to export data out of the system.

When the ETM is configured to be a parallel trace source, the pins TRACECLK and TRACEDx must be configured to their alternate trace functions. There are four pins for data trace and one pin for the clock that need to be programmed:

- TRACECLK—always exported to enable synchronization back with the data. The TRACECLK is connected to PCK3 internally in Atmel | SMART SAM V7/E7/S7. The maximum frequency is 150 MHz due to the pad characteristics
- TRACED0, D1, D2, D3— the instruction trace stream

The trace can be output on a single SWO pin:

- TRACESWO—trace out port over this single pin. Manchester encoded and UART formats are supported.

2.4. Hardware Implementation: Trace Connector CoreSight

Several target connectors can be used to capture up to four bits of parallel trace in the TPIU continuous mode:

- CoreSight™ 20—Instruction trace supported by ULINK*pro* and some third party debuggers, 20 pins
- MIPI 34—Defined by the Mobile Industry Processor Interface Alliance, 34 pins

The Atmel SAM V71 Xplained Ultra board implements a CoreSight20, 20-pin, 50-mil keyed connector (pin seven is removed). The figures below give the schematic view of the connector and show the location of the connector on the SAM V71 Xplained Ultra board.

Atmel | SMART SAM V7/E7/S7 microcontrollers support 4-bit parallel trace.

Figure 2-4 CoreSight20 Connector for 4-bit ETM

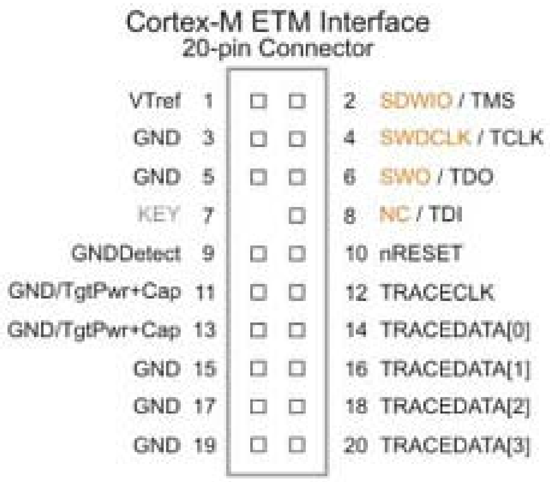
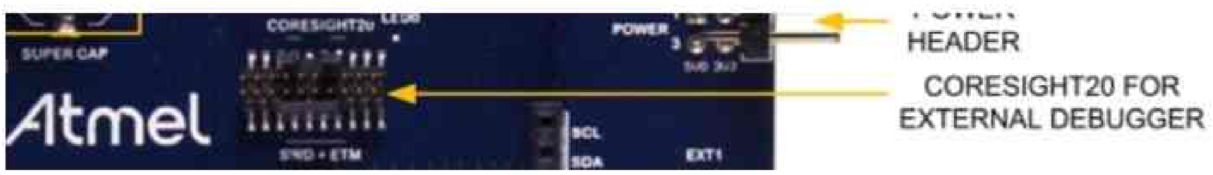


Figure 2-5 Atmel Connector on SAMV71 Xplained Ultra Board



The CoreSight20 connector can be used in either standard JTAG (IEEE 1149.1) mode or Serial Wire Debug (SWD) mode. It can also optionally capture up to four bits of parallel trace in TPIU continuous mode. Refer to the tables below for the pins and their descriptions.

Important: A boundary scan JTAG interface is available in the CoreSight20. This interface cannot be used for debug purposes for Atmel | SMART SAM V7/E7/S7 microcontrollers.

The JTAG debug ports TDI, TDO, TMS and TCK are inactive. They are provided for Boundary Scan Manufacturing Test purposes only.

Important: For Atmel | SMART SAM V7/E7/S7 microcontrollers, several of the trace signals are shared with Ethernet signals. As a result, there is no trace support if Ethernet is used in an application.

Table 2-1 CoreSight20 JTAG Connector Pins

Pin	Signal name	Description
1	VTREF	VCC_TARGET_P3V3
2	TMS	Inactive
3	GND	–
4	TCK	Inactive
5	GND	–

Pin	Signal name	Description
6	TDO	Inactive
9	GND	–
10	nSRST	Synchronous Microcontroller Reset

Table 2-2 CoreSight20 JTAG Connector Pins

Pin	Signal name	Description
1	VTREF	VCC_TARGET_P3V3
2	TMS	Inactive
3	GND	–
4	TCK	Inactive
5	GND	–
6	TDO	Inactive
9	GND	–
10	nSRST	Synchronous Microcontroller Reset

Table 2-3 CoreSight20 Serial Wire Debug (Asynchronous Trace) Pins

Pin	Signal name	Description
2	SWDIO	The Serial Wire Data I/O pin sends and receives serial data to and from the target during debugging. You are advised to series terminate SWDIO close to the target processor.
3	GND	–
4	SWCLK	The Serial Wire Clock pin clocks data into and out of the target during debugging.
5	GND	–
6	TRACESWO	The Serial Wire Output pin can be used to provide trace data to the IDE. You are advised to series terminate SWO close to the target processor
9	GND	–
10	nSRST	Synchronous Microcontroller Reset

Table 2-4 CoreSight20 Parallel Trace Source Pins

Pin	Signal name	Description
12	TRACECLK	The Trace Clock pin provides the IDE with the clock signal necessary to sample the trace data signals. TRACECLK is PCK3.
13	–	–
14	TRACED0	The Trace Data [0–3] pins provide the IDE with TPIU continuous mode trace data from the target.

Pin	Signal name	Description
15	Ground	–
16	TRACED1	–
17	Ground	–
18	TRACED2	–
19	Ground	–
20	TRACED3	–

3. Software Configuration

3.1. Pin Configuration / Multiplexing for SAM V7/E7/S7 Microcontrollers

When ETM is used to capture up to four bits of parallel trace, the pins listed in the table below must be configured:

Table 3-1 Pin Configuration 4-bit Parallel Trace

Pin Name	Peripheral	Pin Number
TRACECLK	D	PD8
TRACED0	C	PD4
TRACED1	C	PD5
TRACED2	C	PD6
TRACED3	C	PD7

When ETM is used in Serial Wire Debug (SWD) mode, the pins listed in the table below must be configured. These pins are not multiplexed with standard IOs and are selected by the System I/O Configuration Register (CCFG_SYSIO).

Table 3-2 Pin Configuration Serial Wire Debug Mode

Pin Name	Pin Number
TRACESWO	PB5
SWDIO	PB6
SWCLK	PB7

3.2. Generic Configuration of ETM

ETM registers are defined in the ARM document *ARM CoreSight ETM-M7*.

To use the ETM, some registers must be configured. Follow the steps below:

1. Set the TRCENA bit to 1 into the Debug Exception and Monitor Register (0xE000EDFC) to enable the use of trace and debug blocks.
2. Select the type of trace: parallel or serial: Write into the Selected Pin Protocol Register 0 (0xE00400F0) for the Parallel Port, and Selected Pin Protocol Register 1 for SWO Manchester or 2 for SWO NRZ.
3. Set the suitable clock for PCK3.
4. Unlock the ETM register access (ETM->TRCLAR register).
5. Configure the ETM (ETM->TRCCONFIGR register).
6. Enable the ETM (ETM->TRCPRGCTLR register).

Configure the TPIU

```
// Set the TRCENA bit to 1 into the Debug Exception and Monitor Register
(0xE000EDFC) to enable the use of
```

```
// trace and debug blocks
CoreDebug->DEMCR |= CoreDebug_DEMCR_TRCENA_Msk;
// Configure TPIU for trace port mode, 4 bits
TPI->CSPSR = (1<<3); // Current Parallel Port Size Register
// Selected Pin Protocol Parallel Port
// Use the Selected Pin Protocol Register to select which protocol to use
for trace output.
TPI->SPPR = TPI_PIN_TRACEPORT << TPI_SPPR_TXMODE_Pos;
// Set the suitable clock PCK3
PMC->PMC_SCDR = PMC_SCDR_PCK3;
while((PMC->PMC_SCSR)& PMC_SCSR_PCK3);
PMC->PMC_PCK[3] = PMC_MCKR_CSS_MAIN_CLK | PMC_MCKR_PRES_CLK_2; // 150 MHz
PMC->PMC_SCER = PMC_SCER_PCK3;
while(!((PMC->PMC_SR) & PMC_SR_PCKRDY3));
```

Configure the ETM

```
// When programming the ETM registers, you must enable all the changes at
the same time.
ETM->TRCLAR = ETM_TRCLAR_KEY_UNLOCK;
//ETM->TRCPRGCTLR = 0;
//while( (ETM->TRCSTATR & ETM_TRCSTATR_IDLE_Msk) == 0 );
ETM->TRCCONFIGR = ETM_TRCCONFIGR_TS_Msk // Global timestamp tracing enabled
| ETM_TRCCONFIGR_RS_Msk // Return stack enabled
| ETM_TRCCONFIGR_CCI_Msk;
// Event Control 0 Register
// Single selected resource
ETM->TRCEVENTCTL0R = 0;
// Event Control 1 Register
// Low power state behavior unaffected.
// ATB trigger disabled
// Event does not cause an event element.
ETM->TRCEVENTCTL1R = 0;
// Stall Control Register
// The trace unit must not discard any data trace elements
// The trace unit must not prioritize instruction trace.
// The trace unit must not stall the processor.
// Zero invasion. This setting has a greater risk of a FIFO overflow
ETM->TRCSTALLCTLR = 0;
// Synchronization Period Register
// Instruction trace cycle count threshold: 12. 8 to 20, 0: disabled.
// The threshold represents the minimum interval between cycle count trace
packets.
ETM->TRCSYNCPR = 0x0UL << ETM_TRCSYNCPR_PERIOD_Pos;
// Trace ID Register
// 0x22
// Global Timestamp Control Register
// Single selected resource
// When TYPE is 0, selects a single selected resource from 0-15 defined by
bits[3:0].
ETM->TRCTSCTLR = 0;
// ViewInst Main Control Register
// Start/stop logic is in the started state.
// When TYPE is 1, selects a Boolean combined resource pair from 0-7
defined by bits[2:0].
ETM->TRCVICTLR = (0x1UL << ETM_TRCVICTLR_SSSTATUS_Pos); // | (0x1UL <<
ETM_TRCVICTLR_EVENT_Pos);
// ViewInst Include/Exclude Control Register
ETM->TRCVIIECTLR = 0;
// ViewInst Start/Stop Control Register
```

```
ETM->TRCVISSCTLR = 0;
// Enable ETM
// Programming Control Register
// The external pin STIMREQ is HIGH, and clocks are enabled except for
when the CPUACTIVE
// input is deasserted, or non-invasive debug is disabled, and all trace
has been drained. The trace unit
// is enabled. Writes to most registers are ignored.
ETM->TRCPRGCTLR = ETM_TRCPRGCTLR_EN_Msk;
while( (ETM->TRCSTATR & ETM_TRCSTATR_IDLE_Msk) == ETM_TRCSTATR_IDLE_Msk );
```

4. Using Keil MDK-ARM IDE with ULINK*pro*

The Keil® ULINK*pro* Debug and Trace Unit connects the USB port of the PC to the targeted system via a JTAG, Cortex Debug, or Cortex Debug+ETM connector.

It allows you to program, debug, and analyze the application using its unique streaming trace technology.

The instruction trace for ARM Cortex-M7 is theoretically possible up to 800Mbit/s.

4.1. Software Implementation

Keil MDK-ARM IDE implements its own ETM configuration.

The user must configure only the ETM master clock and the pins listed in Table 3-1 Pin Configuration 4-bit Parallel Trace on page 10 and Table 3-2 Pin Configuration Serial Wire Debug Mode on page 10.

This is done in an .ini file created by the user. The path to the file is specified in Keil MDK-ARM. This file will be launched on an MDK-ARM reset.

SAMx7_TP.ini file

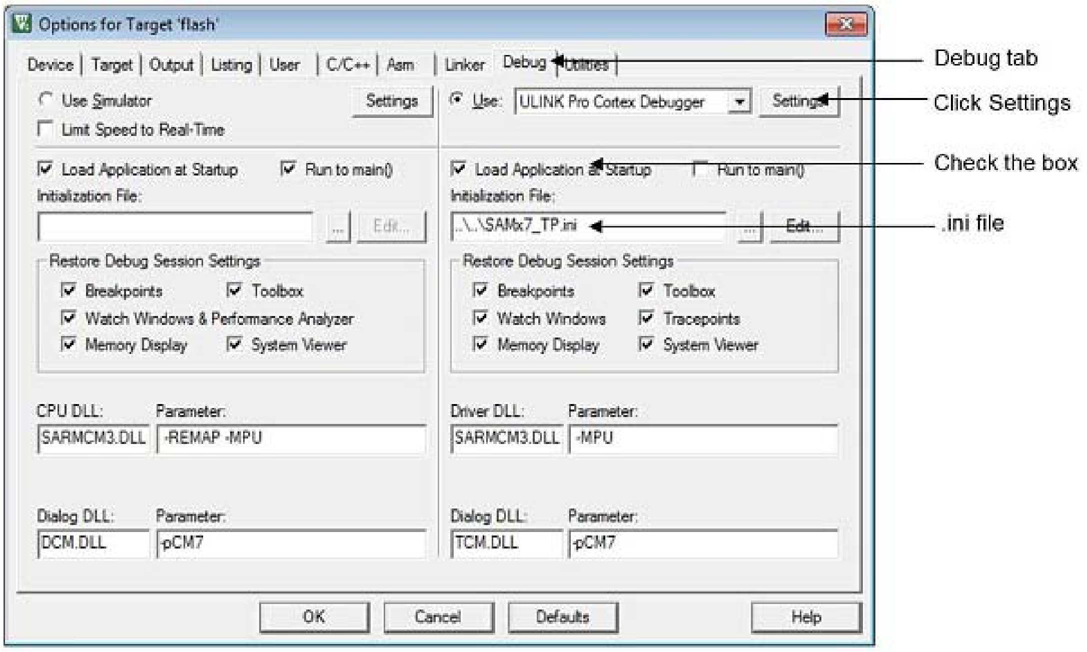
```
// Trace Clock Setup
_WDWORD (0x400E06E4, 0x504D4300); // Disable PMC write protection
_WDWORD (0x400E064C, 0x4); // Select Master clock for ETM
_WDWORD (0x400E0600, _RDWORD(0x400E0600) | (1<< 11) ); // Enable PCK3
_WDWORD (0x400E0610, _RDWORD(0x400E0610) | (1 << 16)); // Enable PIOD clock
// Trace Pin Setup
_WDWORD (0x400E14E4, 0x50494F00); // Disable PIOD write protection
_WDWORD (0x400E1404, 0x1F0); // Disable Peripherals PD04..PD08
// Set Peripheral C for PD04..PD07 and Peripheral D for PD08
_WDWORD (0x400E1470, (_RDWORD(0x400E1470) & (~0x0F0))|0x100);
_WDWORD (0x400E1474, _RDWORD(0x400E1474) | 0x1F0);
TP_Setup();
```

4.2. Configuring Keil MDK-ARM IDE

To configure Keil MDK-ARM IDE, follow the steps below:

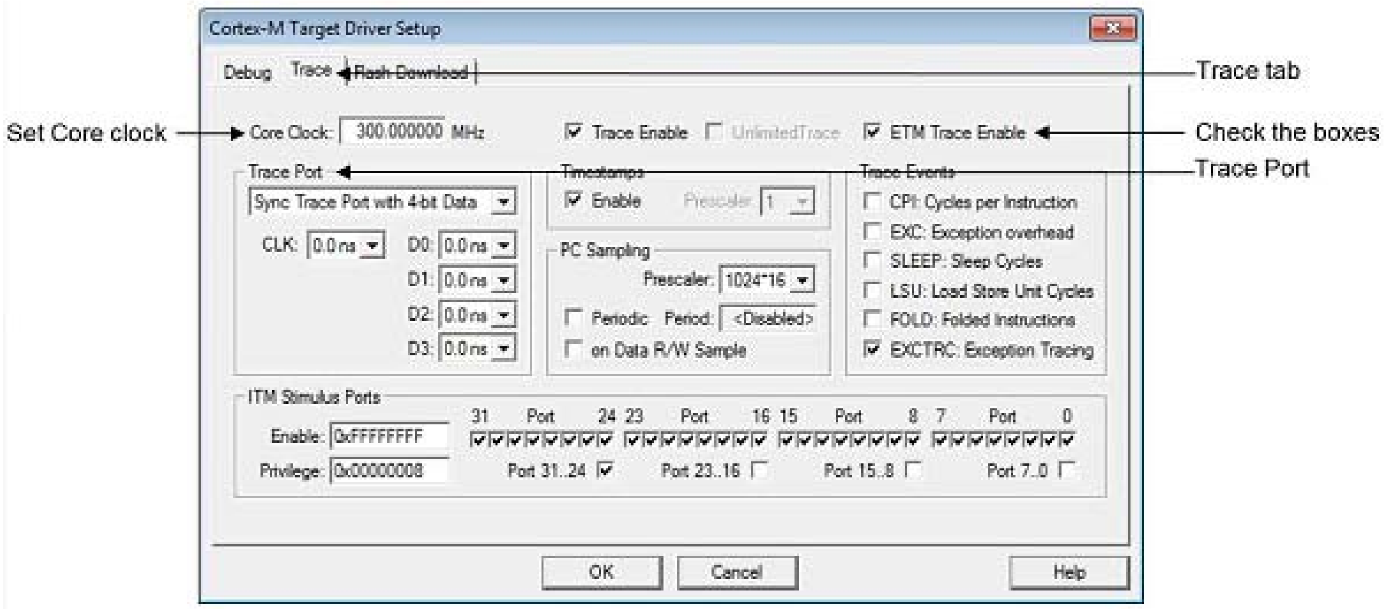
1. Select the tab *Debug*.
2. Check the box *Load Application at Startup*.
3. Complete the field *Initialization File* with the pathname to the .ini file.
4. Click the *Settings* button.
5. A new window opens. Select the tab *Trace*.
6. Check the boxes *Trace Enable* and *ETM Trace Enable*.
7. Set the parameters *Trace Port* and *Core Clock* to the device configuration.

Refer to the screenshots below.


Options for Target 'flash'
Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities
Debug tab
Click Settings
Check the box
.ini file
OK
Cancel
Defaults
Help

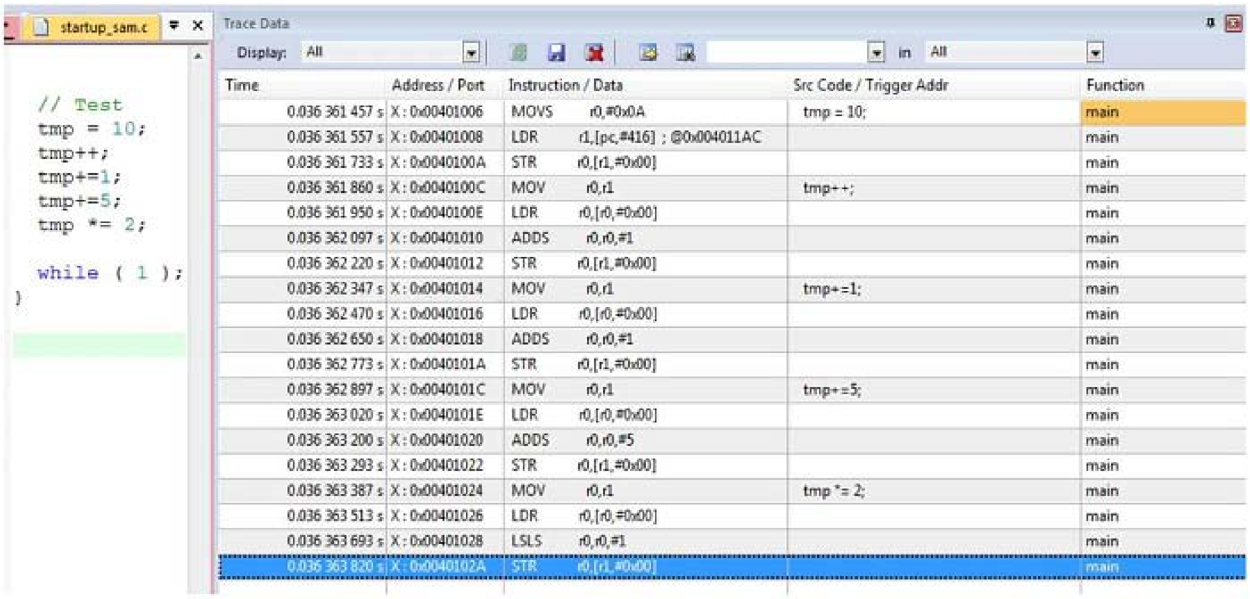
Use Simulator — Settings
Limit Speed to Real-Time
[✓] Load Application at Startup [✓] Run to main()
Initialization File:
Restore Debug Session Settings
[✓] Breakpoints [✓] Toolbox
[✓] Watch Windows & Performance Analyzer
[✓] Memory Display [✓] System Viewer
CPU DLL: SARMCM3.DLL Parameter: -REMAP -MPU
Dialog DLL: DCM.DLL Parameter: -pCM7

(•) Use: ULINK Pro Cortex Debugger — Settings
[✓] Load Application at Startup [ ] Run to main()
Initialization File: ..\..\SAMx7_TP.ini
Restore Debug Session Settings
[✓] Breakpoints [✓] Toolbox
[✓] Watch Windows [✓] Tracepoints
[✓] Memory Display [✓] System Viewer
Driver DLL: SARMCM3.DLL Parameter: -MPU
Dialog DLL: TCM.DLL Parameter: -pCM7



Core Clock: 300.000000 MHz
[✓] Trace Enable [ ] Unlimited Trace [✓] ETM Trace Enable
Trace Port: Sync Trace Port with 4-bit Data
CLK: 0.0 ns D0: 0.0 ns D1: 0.0 ns D2: 0.0 ns D3: 0.0 ns
Timestamps: [✓] Enable Prescaler: 1
PC Sampling: Prescaler: 1024*16
[ ] Periodic Period: <Disabled>
[ ] on Data R/W Sample
Trace Events:
[ ] CPI: Cycles per Instruction
[ ] EXC: Exception overhead
[ ] SLEEP: Sleep Cycles
[ ] LSU: Load Store Unit Cycles
[ ] FOLD: Folded Instructions
[✓] EXCTRC: Exception Tracing
ITM Stimulus Ports
Enable: 0xFFFFFFFF
Privilege: 0x00000008
Port 31..24 [✓] Port 23..16 [ ] Port 15..8 [ ] Port 7..0 [ ]

4.3. Trace Result



```
// Test
tmp = 10;
tmp++;
tmp+=1;
tmp+=5;
tmp *= 2;

while ( 1 );
}
```

Time	Address / Port	Instruction / Data	Src Code / Trigger Addr	Function
0.036 361 457 s	X : 0x00401006	MOVS r0,#0x0A	tmp = 10;	main
0.036 361 557 s	X : 0x00401008	LDR r1,[pc,#416] ; @0x004011AC		main
0.036 361 733 s	X : 0x0040100A	STR r0,[r1,#0x00]		main
0.036 361 860 s	X : 0x0040100C	MOV r0,r1	tmp++;	main
0.036 361 950 s	X : 0x0040100E	LDR r0,[r0,#0x00]		main
0.036 362 097 s	X : 0x00401010	ADDS r0,r0,#1		main
0.036 362 220 s	X : 0x00401012	STR r0,[r1,#0x00]		main
0.036 362 347 s	X : 0x00401014	MOV r0,r1	tmp+=1;	main
0.036 362 470 s	X : 0x00401016	LDR r0,[r0,#0x00]		main
0.036 362 650 s	X : 0x00401018	ADDS r0,r0,#1		main
0.036 362 773 s	X : 0x0040101A	STR r0,[r1,#0x00]		main
0.036 362 897 s	X : 0x0040101C	MOV r0,r1	tmp+=5;	main
0.036 363 020 s	X : 0x0040101E	LDR r0,[r0,#0x00]		main
0.036 363 200 s	X : 0x00401020	ADDS r0,r0,#5		main
0.036 363 293 s	X : 0x00401022	STR r0,[r1,#0x00]		main
0.036 363 387 s	X : 0x00401024	MOV r0,r1	tmp *= 2;	main
0.036 363 513 s	X : 0x00401026	LDR r0,[r0,#0x00]		main
0.036 363 693 s	X : 0x00401028	LSLS r0,r0,#1		main
0.036 363 820 s	X : 0x0040102A	STR r0,[r1,#0x00]		main

5. Suggested Reading

5.1. Device Documentation

Each SAM V71, SAM V70, SAM E70 and SAM S70 datasheet contains block diagrams of the peripherals and details on implementing firmware for the device. These datasheets also contain the electrical specifications and expected characteristics of the device.

The datasheets are available on http://www.atmel.com/ in the Datasheets section of the product page.

5.2. CoreSight System Trace Macrocell Documentation

These documents are available at http://www.arm.com/ in the Infocenter section.

http://www.arm.com/products/system-ip/debug-trace/trace-macrocells-etm/coresight-system-trace-macrocell.php

http://infocenter.arm.com/help/topic/com.arm.doc.ddi0314h/DDI0314H_coresight_components_trm.pdf

5.3. ARM Cortex-M7 Processor Documentation

These documents are available at http://www.arm.com/ in the Infocenter section.

- Cortex-M7 Devices Generic User Guide (ARM DUI 0646A)
- Cortex-M7 Technical Reference Manual (revision r0p2)
- ARM CoreSight ETM-M7 (revision r0p1)

6. Conclusion

ETM is a simple but powerful tool for the software programmer.

On Keil MDK-ARM IDE, the ETM can be easily implemented using the software provided in this document.

This software will be used with IAR IDE when the IDE supports the ETM.

Keil IDE has its own configuration, and can be used very easily using the .ini file.

7. Appendix A – Using IAR

Important: ETM/ETB trace mode for ARM Cortex-M7 is not implemented in this software version. Trace is disabled.

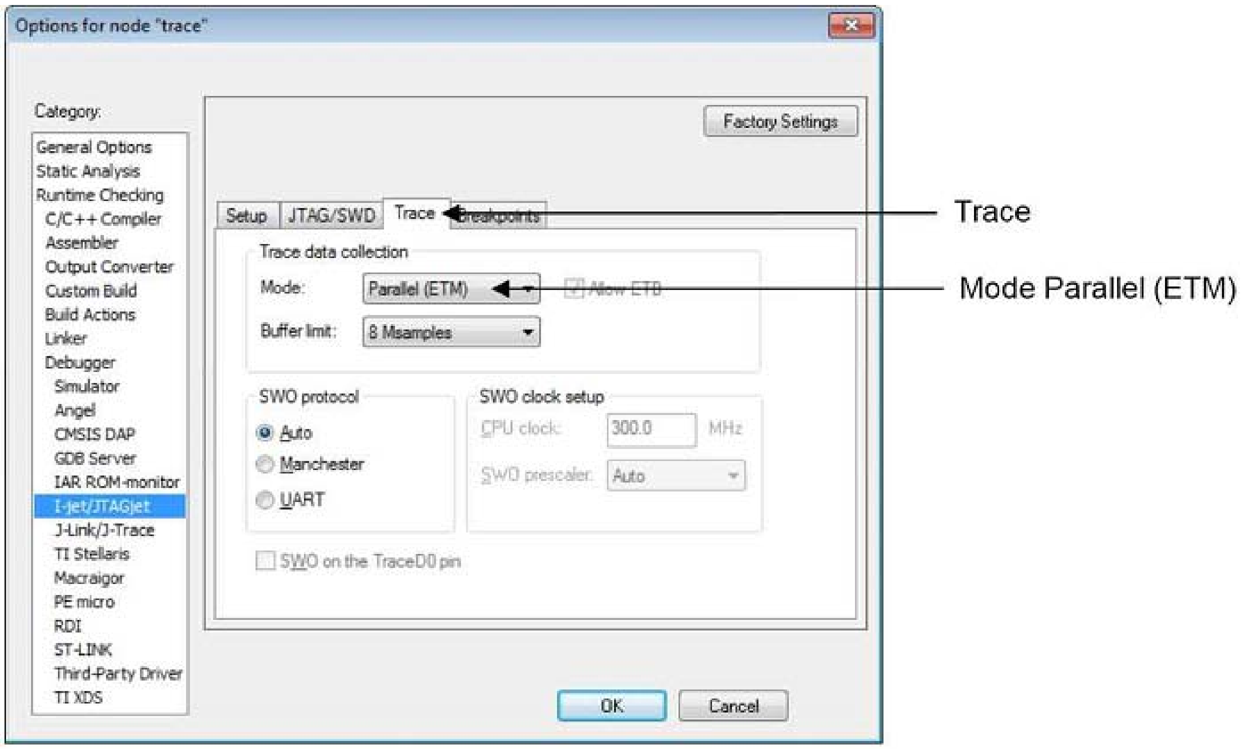
IAR® I-jet Trace provides extensive debugging and trace functionality. It delivers large trace memory capacities and high-speed communication via SuperSpeed USB 3.0. I-jet Trace is equipped with Embedded Trace Macrocell (ETM) trace, supporting all ARM Cortex-M devices with ETM.

The ETM trace clock is supported up to 150 MHz.

7.1. Software Implementation

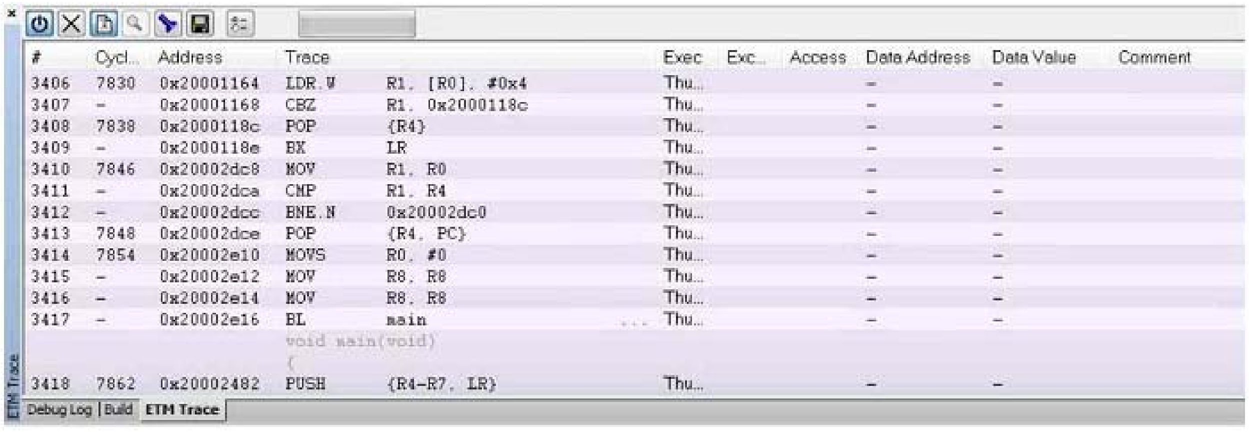
IAR has not yet implemented its own ETM configuration. The configuration must be added in the user software.

7.2. How to Configure IAR Embedded Workbench



Trace data collection
Mode: Parallel (ETM)
Allow ETB
Buffer limit: 8 Msamples
SWO protocol: Auto
CPU clock: 300.0 MHz
SWO prescaler: Auto
SWO on the TraceD0 pin

7.3. Trace Result



#	Cycl...	Address	Trace		Exec	Exc...	Access	Data Address	Data Value	Comment
3406	7830	0x20001164	LDR.W	R1, [R0], #0x4	Thu...			–	–	
3407	–	0x20001168	CBZ	R1, 0x2000118c	Thu...			–	–	
3408	7838	0x2000118c	POP	{R4}	Thu...			–	–	
3409	–	0x2000118e	BX	LR	Thu...			–	–	
3410	7846	0x20002dc8	MOV	R1, R0	Thu...			–	–	
3411	–	0x20002dca	CMP	R1, R4	Thu...			–	–	
3412	–	0x20002dcc	BNE.N	0x20002dc0	Thu...			–	–	
3413	7848	0x20002dce	POP	{R4, PC}	Thu...			–	–	
3414	7854	0x20002e10	MOVS	R0, #0	Thu...			–	–	
3415	–	0x20002e12	MOV	R8, R8	Thu...			–	–	
3416	–	0x20002e14	MOV	R8, R8	Thu...			–	–	
3417	–	0x20002e16	BL	main ...	Thu...			–	–	
			void main(void)							
			{							
3418	7862	0x20002482	PUSH	{R4-R7, LR}	Thu...			–	–	

8. Appendix B - Glossary

The table below provides commonly-used abbreviations and their definitions.

Abbreviation	Definition
ATB	Advanced Trace Bus
ETM	Embedded Trace Macrocell
IDE	Integrated Development Environment
ITM	Instrumentation Trace Macrocell
NVIC	Nested Vector Interrupt Controller
PPB	Private Peripheral Bus
SWD	Serial Wire Debug
SWO	Serial Wire Output
TCM	Tightly-Coupled Memory
TPIU	Trace Port Interface Unit

9. Revision History

Doc. Rev.	Comments
A	First issue