# ARM Tarmac Specification

**PD**

Document Number: ARM-EPM-041435 1.0
Date of Issue: 9th April 2013



## Abstract

This document describes the representation of standard architectural execution trace events for ARM processors.

# CONTENTS

# 1 ABOUT THIS DOCUMENT

## 1.1 Change Control

### 1.1.1 Current Status and Anticipated Changes

### 1.1.2 Change History

Only the details of approved issues are recorded here. Changes between drafts are documented in the auspex revision history.

| ***Issue*** | ***Date*** | ***Change*** |
|---|---|---|
| 1.0 | 9th April 2013 | First release |

## 1.2 References

This document refers to the following documents.

| Ref | Doc No | Author(s) | Title |
|---|---|---|---|
| 1. | ARM DDI 0487 | | ARM Architecture Reference Manual ARMv8, for ARMv8-A architecture profile |
| 2. | ARM DDI 0406 | | ARM Architecture Reference Manual ARMv7-A and ARMv7-R edition |
| 3. | ARM DUI 0403 | | ARMv7M Architecture Reference Manual |

## 1.3 Terms and Abbreviations

The terms and abbreviations used in this document are all defined in the ARM architecture reference manuals (see references).

## 1.4 Scope

This document describes the representation of standard architectural execution trace events. Note, some implementations might use trace records that are implementation-specific. Therefore, to use the trace output you might also require other documents from the implementation bundle.

The exact format of each event depends on its type. Typically all addresses, data, and IDs use hexadecimal with leading zeros.

# 2 TRACE SPECIFICATION

Our trace consists of a header followed by a series of execution events. Each execution event appears as a multi-line record in the trace.

## 2.1 Header

The header has the following format::

```
Tarmac Text Rev <REV><FLAGS>
```

Field descriptions:

- <REV> - this document describes revision 3
- <FLAGS> - currently only one flag implemented:
  - 't' – indicates that timestamp is present

Example Header:

```
Tarmac Text Rev 3t
```

## 2.2 Timestamp

The timestamp, if present (if the 't' flag is set in the header), is prepended to the first line of each Execution Event. It is printed with 11 significant digits (prepended with spaces as necessary to maintain alignment) followed by the timestamp unit which can be one of:

- s,ms,us,ns,ps,fs – unit of time (second, millisecond, microsecond, etc.)
- clk – the timestamp is a count of clock cycles (typically of the processor clock)

If an execution event has no timestamp for some reason, 11 dashes and 4 spaces are printed instead.

Example of timestamp prepended to the first line of an execution event record (an instruction record in this case):

```
       9627 clk ES   (ab465108:e2800010) A hyp_ns:          ADD      r0,r0,#0x10
```

## 2.3 Execution Event

After the header, the trace consists of a series of execution events. Each event consists of one or more lines of text. The possible types of lines (described in the subsections below) are:

- Instruction Execution line
- Asynchronous Event line
- Exception entry line
- Memory access line
- Register update line
- Branch line

Every execution event begins with a timestamp, followed by an *Execution Step* 'ES' tag and then by either an instruction execution line or an asynchronous event line. Depending on the event, it will then contain a mix of exception entry, memory access, register update, and/or branch lines.

### 2.3.1 Instruction execution line

Instruction execution is represented by a single line with the following format:

```
(20216ea8:e3510002) A svc_ns:          CMP      r1,#2
(00002358:1a000792) A svc:     CCFAIL BNE      {pc}+0x1e50 ; 0x41a8
```

```
(000022e0:--------) A svc:
(-------1:-------2) 3 --4--5:  -----6 ............................7
```

Field descriptions:

1. Instruction Address : 16 hex digits when executing in A64, else 8 hex digits. This is the virtual address of the instruction being executed.
    - If the cpu is in debug state, instead of the instruction address this field will contain:
        - DBGITR if v7debug is implemented
        - EDITR if v8debug is implemented
2. Instruction Opcode : The opcode is printed with only the meaningful number of digits (4 hex digits for a 16-bit opcode, 8 hex digits for 32-bit), and the correct padding is added after the closing parentheses to maintain alignment. When no opcode is present (e.g. on a prefetch abort) 8 dashes are printed instead.
3. ISA : takes one of the following values:
    - A : AArch32 ARM
    - T : AArch32 Thumb
    - E : AArch32 ThumbEE
    - O : AArch64
4. Execution mode
    - If ISA==AArch32, possible values are: usr, sys, svc, abt, und, irq, fiq, hyp, mon
    - If ISA==AArch64, possible values are: el0t, el1t, el1h, el2t, el2h, el3t, el3h
5. Security state : If the core is in Nonsecure state, '_ns' is displayed, else nothing
6. CCFAIL : If ISA==AArch32, and a conditional instruction is ccfailed, 'CCFAIL' is displayed, else this field remains blank.
7. Disassembly : The disassembled opcode, possibly followed by some source code label information.

### 2.3.2 Asynchronous event line

When an asynchronous event is taken by the processor, a line is added to indicate the type of event::

```
Reset
....1
```

Field description:

1. Type of asynchronous event. Can be one of:
    - Reset
    - External Debug Event
    - IRQ
    - FIQ
    - System Error (Abort)
    - Virtual IRQ
    - Virtual FIQ
    - Virtual System Error (Virtual Abort)
    - Background Update

### 2.3.3 Exception entry line

Both instruction execution and asynchronous events can trigger exception entry. In this case a line is added to indicate the vector the exception is jumping to::

```
EXC [0x04] Undefined Instruction
--- ............................1
```

Field Description:

1. Type of Exception. The hex value in brackets is the vector offset of this exception
    - For v7A/R and v8 with ISA==AArch32, the possible values are:
        - [0x00] Reset

        - [0x04] Undefined Instruction
        - [0x08] System Call
        - [0x0c] Prefetch Abort
        - [0x10] Data Abort
        - [0x14] Hypervisor Trap
        - [0x18] IRQ Interrupt
        - [0x1c] FIQ Interrupt
        - [-8] Array Bound Check : ThumbEE only
        - [-4] Null Pointer Check : ThumbEE only
    - For v8 with ISA==AArch64, the possible values are:
        - [0x000] Synchronous Current EL with SP_EL0
        - [0x080] IRQ/vIRQ Current EL with SP_EL0
        - [0x100] FIQ/vFIQ Current EL with SP_EL0
        - [0x180] SError/vSError Current EL with SP_EL0
        - [0x200] Synchronous Current EL with SP_ELx
        - [0x280] IRQ/vIRQ Current EL with SP_ELx
        - [0x300] FIQ/vFIQ Current EL with SP_ELx
        - [0x380] SError/vSError Current EL with SP_ELx
        - [0x400] Synchronous Lower EL using AArch64
        - [0x480] IRQ/vIRQ Lower EL using AArch64
        - [0x500] FIQ/vFIQ Lower EL using AArch64
        - [0x580] SError/vSError Lower EL using AArch64
        - [0x600] Synchronous Lower EL using AArch32
        - [0x680] IRQ/vIRQ Lower EL using AArch32
        - [0x700] FIQ/vFIQ Lower EL using AArch32
        - [0x780] SError/vSError Lower EL using AArch32
    - For v7M, the possible values are :
        - [1] Reset
        - [2] NMI
        - [3] HardFault
        - [4] MemManage
        - [5] BusFault
        - [6] UsageFault
        - [11] SVCall
        - [12] Debug Monitor
        - [14] PendSV
        - [15] SysTick
        - [<N>] External Interrupt
    - Debug entry
        - Debug Halt Mode Entry (no vector offset for this exception)

### 2.3.4 Memory access line

Memory accesses are represented as a set of aligned chunks of memory values. Each chunk uses the following format:

```
ST 0000cff0  00001ce8 ........ ........ ........   S:000000cff0/0  SO
ST 13000000  ........ ........ ........ ......6f   S:0013000000/20 DV OSH
ST 00001770  0000dc08 ........ ........ ........   S:0000001770/12 NM NSH INC ONC
LD 00002f50  00c51879 ........ ........ ........   S:0000002f50/12 NM NSH INC ONC
LD 00000040  ........ 00001798 ........ ........   S:0000000040/12 NM NSH IWBRWA OWBRWA
LD 00000040  ........ 00001798 ........ ........   S:0000000040/12 NM NSH IWBRWA OWBRWA
-1 -------2  ----------------------------------3   4:---------5/-6 -7 --8 .....9 ....10
```

Field Descriptions (fields 4-10 are optional):

1. Type of Access : possible values are:
    - LD – load
    - LA – aborted load
    - ST – store
    - SA – aborted store
    - SX – failed exclusive store
2. Virtual address of the access : always aligned on a 16-byte boundary. 16 hex digits if ISA==AArch64, else 8 hex digits
3. Memory values of the bytes architecturally accessed, organized as 4 32-bit words of 8 hex digits each, rightmost word is at offset 0x0, leftmost word is at offset 0xc. Bytes not accessed appear as '..'.
4. (External) Secure bit of this access : 'S' is secure, 'NS' if Non-secure.
5. Physical address : 10 hex digits, aligned just as the virtual address
6. Page Order : logbase2 of the size of the page accessed (e.g. 12 for a 4k page because $2^{12}$=4k)
7. Memory type : possible values are:
    - For v7:
        - SO – Strongly-ordered
        - DV - Device memory
        - NM – Normal memory
    - For v8:
        - nGnRnE – nonGathering, nonReordering, nonEarly-write acknowledging (equivalent to v7 SO)
        - nGnRE - nonGathering, nonReordering, Early-write acknowledging (equivalent to v7 DV)
        - nGRE – nonGathering, Reordering, Early-write acknowledging
        - GRE – Gathering, Reordering, Early-write acknowledging
        - NM – Normal memory
8. Shareability – when the memory type is DV or NM, this field can take the following values:
    - OSH – Outer Shareable
    - ISH – Inner Shareable
    - NSH – Non-shareable
9. Inner cache attributes – when memory type is NM, this field can take the following values (with a prefix of I to indicate 'inner'):
    - NC – Non-cacheable
    - WTNA – Write-Through No-Allocate
    - WTRA – Write-Through Read-Allocate
    - WTWA – Write-Through Write-Allocate
    - WTRWA: Write-Through Read/Write-Allocate
    - WBNA: Write-Back No-Allocate
    - WBRA: Write-Back Read-Allocate
    - WBWA: Write-Back Write-Allocate
    - WBRWA: Write-Back Read/Write-Allocate
10. Outer cache attributes – when memory type is NM, this field can take the same values as field 9, except with a prefix of O to indicate 'outer'.

### 2.3.5 Register update line

A given register update is represented using the following format::

```
R R12 (FIQ) 0000e598
- ..1 (..2) .......3


R Q0 (AARCH32) -------- -------- -------- 00000000
- .1 (......2) ................................3
```

Field Descriptions:

1. Register Name
2. Context – more identifying information about the register updated. For example:

- For ARM core register file updates in AArch32 mode, the context specifies which logical version of the register was updated (The FIQ copy of R12 was updated in the example above).

3. Updated data for this register – the width of this field is appropriate for the register being updated. On partial updates, the un-updated parts appear as dashes.

### 2.3.6 Branch line

If the implementation supports it, branches are represented in the trace using a line similar to:

```
BR (00001c98) A
-- (.......1) 2
```

Field Descriptions:

1. Target Instruction Address – displayed with 16 hex digits for AArch64, else 8 hex digits. This value is a virtual address.
2. Target Instruction Set – same single letter encodings as field 3 of the instruction record.

## 2.4 Example

An example of an execution trace output is:

```
Tarmac Text Rev 3t
     1715 ns  IT  (00000000:e59ff038) A svc:           LDR      pc,{pc}+0x40 ; 0x40
                  LD 00000040  ........ ........ ........ 00001c98   S:0000000040/20 SO
                  BR (00001c98) A
     2225 ns  IT  (00001c98:e10f0000) A svc:           MRS      r0,APSR ; formerly CPSR
                  R R0 (USR) 000001d3
     2225 ns  IT  (00001c9c:e200001f) A svc:           AND      r0,r0,#0x1f
                  R R0 (USR) 00000013
     2415 ns  IT  (00001ca0:e350001a) A svc:           CMP      r0,#0x1a
                  R CPSR (AARCH32) 800001d3
     2415 ns  IT  (00001ca4:1a00000e) A svc:           BNE      {pc}+0x40 ; 0x1ce4
                  BR (00001ce4) A

[...]

    10625 ns  IT  (0000d800:ef000002) A svc:           SVC      #0x2
                  EXC [0x08] System Call
                  R LR (SVC) 0000d804
                  R SPSR (SVC) 600000d3
                  R CPSR (AARCH32) 600000d3
                  BR (00000008) A
    10975 ns  IT  (00000008:e59ff038) A svc:           LDR      pc,{pc}+0x40 ; 0x48
                  #0000000000004840
                  LD 00000040  ........ 00001798 ........ ........   S:0000000040/20 SO
                  BR (00001798) A
```

---