EJTAG Trace Control Block Specification

Document Number: MD00148
Revision 1.04
March 21, 2002

MIPS Technologies, Inc.
1225 Charleston Road
Mountain View, CA 94043-1353

Table of Contents

List of Figures

List of Tables

EJTAG Trace Control Block Specification

1.1 Overview: The On-Chip Trace Control Block

The tracing logic within the processor core (not shown in Figure 1-1) outputs all trace information on the PDtrace™ interface (shown in Figure 1-1). This PDtrace™ interface connects to the on-chip trace control block (TCB) unit. The TCB is responsible for collecting the trace data sent every cycle on the PDtrace™ interface by the core's tracing logic. The TCB captures and stores this trace data in an on-chip trace memory or an off-chip trace memory using the Probe Interface Block (PIB). A separate document, the *PDtrace™ Interface Specification* Ref [2], describes the tracing control and mechanism on the processor core, and the PDtrace™ interface signals in detail. This document describes all trace-related blocks and interfaces that are external to the processor core and the PDtrace™ interface. This includes:

- the TCB, with details on the internal architecture, i.e., registers, and how these registers are used to control tracing,
- the formats used by the TCB to write the trace information to memory,
- the interface between the TCB and the TAP controller,
- the TCtrace IF,
- the PIB, and
- the external Probe interface including its electrical characteristics.

Figure 1-1 shows the TCB, the PIB, and the trace data path from the PDtrace™ IF to the Probe IF. It is optional whether the TCB implements on-chip trace memory and/or the TCtrace IF with a PIB and off-chip trace memory.

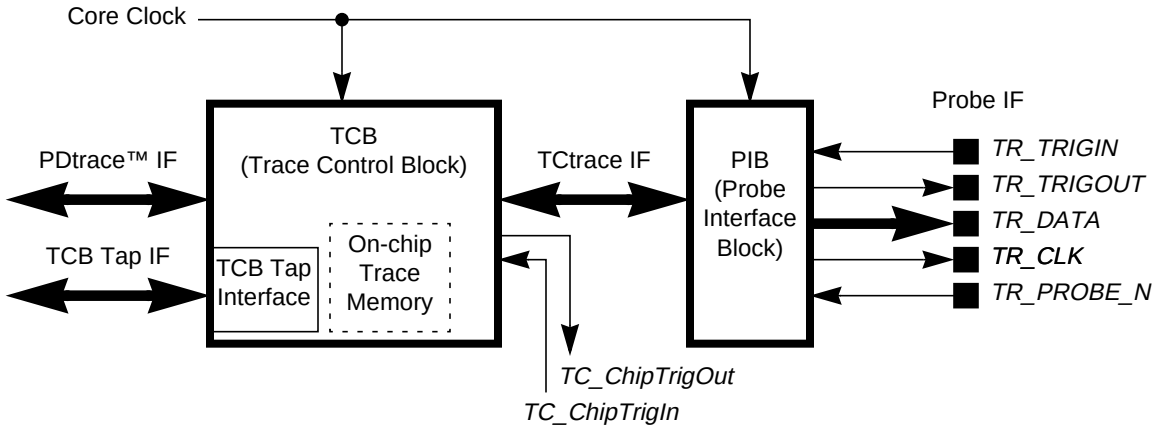


Figure 1-1 TCB and optional PIB overview

Figure 1-2 shows the full system configuration when the TCB is streaming data to off-chip trace memory through the PIB. The number of pins needed for trace data on the Probe IF is configurable to 4, 8, or 16.

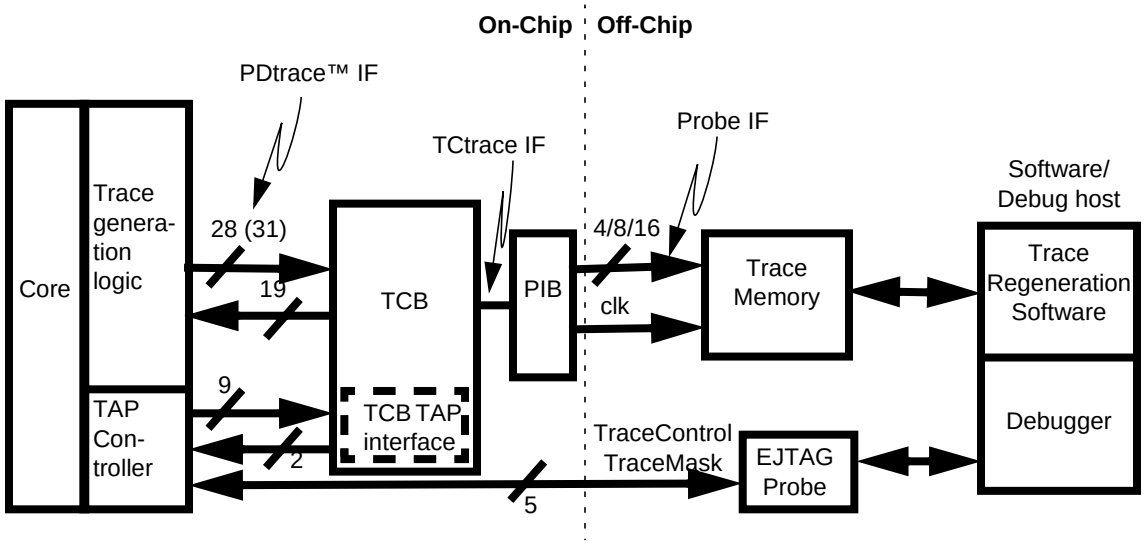


Figure 1-2 Illustration of the core and TCB with external trace memory

Figure 1-3 shows the configuration where the TCB is streaming data to an on-chip trace memory. The size of the on-chip trace memory is configurable. After trace capture has stoppe, the trace data in the on-chip memory is accessed through the EJTAG probe by the Trace Regeneration Software.

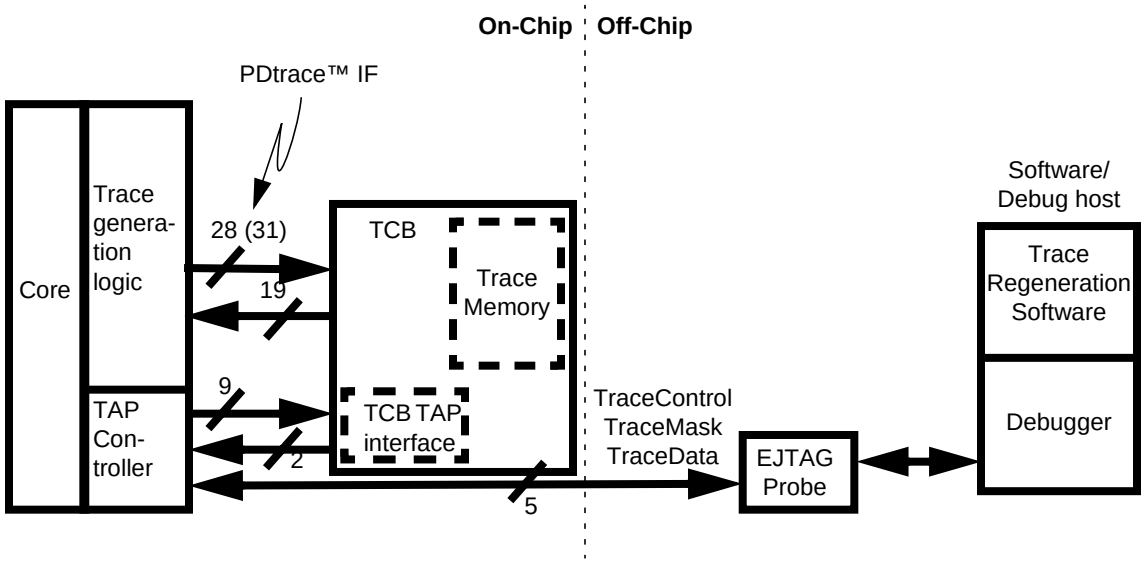


Figure 1-3 Illustration of the core and TCB with internal trace memory

The TCB includes these primary interfaces:

- The PDtrace™ interface to the processor core. A detailed description about the PDtrace™ signal interface is described in the *PDtrace™ Interface Specification* Ref [2].
- The TCB TAP interface, which connects the EJTAG TAP controller resident within the processor core to the TAP functionality present within the TCB. This interface is described in Chapter 7, “Trace Control Block TAP Interface,” on page 39.
- An optional TCtrace interface to the PIB. This interface is described along with the Probe IF in Chapter 8, “TCtrace IF,” on page 43 and Chapter 9, “Probe IF,” on page 47. If the TCB is configured with only on-chip trace memory, then the TCtrace IF and the PIB are not needed.

Trace Message Format

One main function of the TCB is to capture trace information from the PDtrace™ interface and store it to trace memory. This trace information is then analyzed by the trace reconstruction software in the debugger. Since tracing the entire run of a program can require a lot of storage, compression of trace information is a desirable goal. While the trace information undergoes one level of compression in the core, further compression is possible before the trace information is stored to trace memory by the TCB. The TCB achieves this compression using a number of trace formats which eliminate the storage of unnecessary trace bits in each cycle. This section describes these formats.

Note that the description of the trace formats refers to PDtrace™ interface signals. Hence, to fully understand the intent of some of these trace formats, the reader must have a basic understanding of these signals or have access to the PDtrace™ specification document.

2.1 Single-Pipe Tracing Formats

The formats discussed in this section are relevant only when the core or processor being traced is a single-issue, i.e., single pipeline implementation. The multi-pipeline case is discussed in Section 2.2, "Multi-Pipe Tracing Formats".

Recall that when the signal *PDO_IamTracing* is asserted by the processor, there is valid trace data on the other PDtrace™ interface signals, and these values must be captured and stored by the TCB. When *PDO_IamTracing* is de-asserted, no useful trace information is on the other PDtrace™ interface signals, and no trace records need to be stored to trace memory. Hence, whenever the processor has de-asserted the *PDO_IamTracing* signal, the TCB ignores the PDtrace™ interface signal values, and does not store anything to trace memory. In all the cases discussed below, the *PDO_IamTracing* signal is being asserted by the processor tracing logic.

2.1.1 Trace Format 1 (TF1)

A processor stall is identified when *PDO_InsComp[2:0]* is 000, *PDO_TType[2:0]* is 000, and *PDO_Overflow* is not asserted. When the processor is stalled, no execution trace information needs to be recorded except that this was a stall cycle. This can be done efficiently using a single bit “1” for this format. This is Trace Format 1 (TF1) as show in Figure 2-1. Note that this stall information is needed only when tracing is used to account for all execution cycles, i.e., cycle-accurate tracing ($TCBCONTROLB_{CA}$ = 1, see Section 4.2, "TCBCONTROLB Register").

Note that when parsing a trace format sequence, if the first bit of the trace format is a one, then this is TF1 and the next bit is the first bit of the next trace format.

Figure 2-1 TF1 (Trace Format 1)

2.1.2 Trace Format 2 (TF2)

A study of program traces shows that with only PC tracing enabled, nothing of significance needs to be captured a large percentage of the time. For instance, when *PDO_TType[2:0]* is **NT** (000), i.e., **N**o **T**ransmission, the address/data bits (*PDO_AD*) is don't-cares and therefore do not need to be saved. So, when *PDO_TType[2:0]* is **NT** and *PDO_Overflow* is 0, the only significant trace signal is *PDO_InsComp[2:0]*, which describes the completed instruction. Having used a single bit value of "1" for TF1, we indicate the combination of non-zero *PDO_InsComp[2:0]*, zero *PDO_TType[2:0]*, and zero *PDO_Overflow* in two bits (10_2). The next three bits of the format are the value of *PDO_InsComp[2:0]*. This trace format with five bits is called Trace Format 2 (TF2), as shown in Figure 2-2.

Figure 2-2 TF2 (Trace Format 2 Single-Pipe)

2.1.3 Trace Format 3 (TF3)

When *PDO_TType[2:0]* is not **NT** (000) and *PDO_Overflow* is set to 0, all trace information needs to be captured. This is the TF3 format shown in Figure 2-3. The *PDO_LoadOrder[2:0]* signal is an exception in that it only needs to be captured on the last cycle of a **D**ata **T**ransmission transaction (**DT** on the *PDO_TType[2:0]* signal). Hence, a slight distinction is made between this format TF3 (which excludes the *PDO_LoadOrder[2:0]* value, see Figure 2-3), and the format TF4 (which includes the *PDO_LoadOrder[2:0]* value, see Figure 2-4).

TF3 is distinguished from TF2 by having 000_2 on the first three bits. TF3 may be either 27 or 43 bits wide, depending on whether 16 or 32 bits of the *PDO_AD* bus are included in the AD field. The AD field width is determined by the $TCBCONTROLA_{ADW}$ field. (See Section 4.1, "TCBCONTROLA Register").

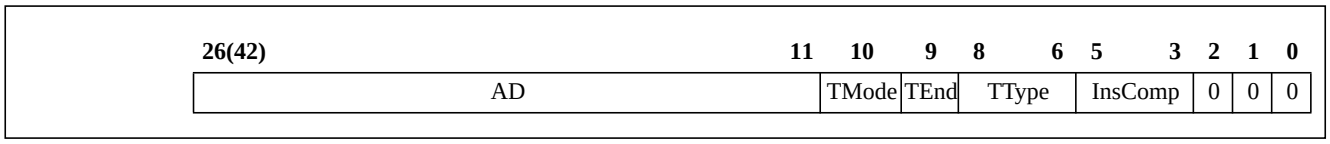


Figure 2-3 TF3 (Trace Format 3 Single-Pipe)

2.1.4 Trace Format 4 (TF4)

The TF4 format is shown in Figure 2-4. TF4 covers the case when *PDO_TType[2:0]* is set to **DT** and *PDO_TEnd* is set to 1, that is, the last cycle of the current data transmission. This is shown in Figure 2-4, where the pattern on bits [9:6] distinguishes TF4 from TF3. Bits [8:6] are equal to 001_2 for a *PDO_TType[2:0]* value of **DT** and bit 9 has a value of 1 for *PDO_TEnd*.

When capturing the cycle by cycle values on the PDtrace™ IF, the last cycle of a Load Data transmission cannot be distinguished from the last cycle of a Store Data transmission (without saving information from a previous cycle, i.e., the *PDO_InsComp* value from the first cycle of the data transaction). This means that the TF4 format will be used for the last cycle of both Load and Store Data transmission, a small inefficiency.

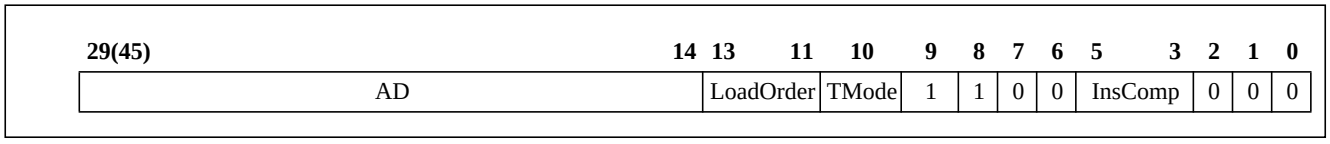


Figure 2-4 TF4 (Trace Format 4 Single-Pipe)

2.1.5 Trace Format 5 (TF5)

When *PDO_Overflow* is asserted, all other PDtrace™ IF trace values are undefined and hence all current cycle trace values can be discarded. (When an overflow does occur, the PDtrace™ IF always sends a full PC value in the next cycle. This is used for resynchronizing to the execution path.) The Trace Format 5 (TF5) shown in Figure 2-5 indicates this overflow.

Figure 2-5 TF5 (Trace Format 5)

2.1.6 Trace Format 6 (TF6)

Trace Format 6 (TF6) shown in Figure 2-6 is provided to the TCB to transmit information that does not directly originate from the cycle by cycle trace data on the PDtrace™ interface. That is TF6 can be used by the TCB to store any information it wants in the trace memory, within the contraints of the specified format. This information can then be used by software for any purpose. For example, TF6 can be used to indicate a special condition, trigger, semaphore, breakpoint, or break in tracing that is encountered by the TCB.

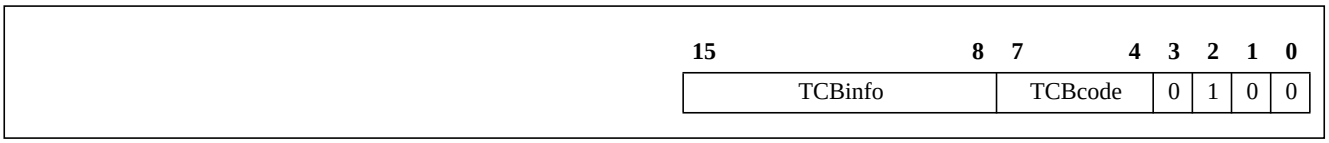


Figure 2-6 TF6 (Trace Format 6)

The definition of TCBcode and TCBinfo is shown in Table 2-1.

Table 2-1 TCBcode and TCBinfo fields of Trace Format 6 (TF6)

TCBcode	Description	TCBinfo
0000	**Trigger Start:** Identifies start-point of trace. TCBinfo identifies what caused the trigger.	Cause of trigger. Taken from the Trigger control register generating this trigger.
0100	**Trigger End:** Identifies end-point of trace. TCBinfo identifies what caused the trigger.	
1000	**Trigger Center:** Identifies center-point of trace. TCBinfo identifies what caused the trigger.	
1100	**Trigger Info:** Information-point in trace. TCBinfo identifies what caused the trigger.	
0001[a]	**No trace cycles:** Number of cycles where the processor is not sending trace data (*PDO_IamTracing* is deasserted), but a stall is not requested by the TCB (*PDI_StallSending* is not asserted). This can happen when the processor, during its execution, switches modes internally that take it from a trace output required region to one where trace output was not requested. For example, if it was required to trace in User-mode but not in Kernel-mode, then when the processor jumps to Kernel-mode from User-mode, the internal PDtrace™ FIFO is emptied, then the processor deasserts *PDO_IamTracing* and stops sending trace informaton. In order to maintain an accurate account of total execution cycles, the number of such no-trace cycles have to be tracked and counted. This TCBcode achieves this goal.	Number of cycles (All zeros is equal to 256). If more than 256 is needed, the TF6 format is repeated.
0101[a]	**Back stall cycles:** Number of cycles when *PDI_StallSending* was asserted, preventing the PDtrace™ interface from transmitting any trace information.	
1x01	Reserved for future use	Undefined
xx10		

Table 2-1 TCBcode and TCBinfo fields of Trace Format 6 (TF6) (Continued)

TCBcode	Description	TCBinfo
xx11	TCB implementation dependent	Implementation dependent
[a]: TF6 formats with this TCBcode is not transmitted when $TCBCONTROLB_{CA}$ is 0		

2.2 Multi-Pipe Tracing Formats

A processor with multiple pipelines requires additional support for sending trace information to trace memory. The TCB can perform some combining and the kind of format crunching as shown in the single-pipe case to reduce the number of bits that are sent out each cycle. If there are k pipelines within the core, 1, 2,... k, then for each cycle, the TCB generates a trace format from each pipeline, in that respective order. The external software programmer must refer to the User's Guide for that core to determine the order of the pipelines as hooked up to the PDtrace™ interface.

The trace format TF1 is usable by the TCB without change for multi-pipe tracing. The TF1 format indicates that the specific pipe did not complete an instruction and had no data to send.

TF5 is a common format. That is, all the pipes have to flush the trace buffer when just one of them has overflowed. Hence, a single instance of TF5 will suffice to cover all the 1..k pipeline stages. The trace reconstruction software must take this into account as it parses the trace formats in trace memory.

The TF6 format is also usable by the TCB without change, and as a common format. For example, the *PDO_IamTracing* and *PDO_StallSending* are common for all pipelines in a multi-pipeline processor. A TF6 format can be used after all the formats for the respective pipelines have been sent. Note that if needed, pipeline-specific information can be encoded within the TF6 format bits.

2.2.1 Multi-Pipe Trace Format 2-4 (TF2, TF3, TF4)

The TF2, TF3, and TF4 formats need the additional *PDO_PgmOrder[2:0]* value for multi-pipeline tracing. The **PgmOrder** field is added to all of them, right after the **InsComp** field, as shown in Figure 2-7, Figure 2-8, and Figure 2-9. The **PgmOrder** field is 3 bits wide to allow up to 8 pipelines. The number of processor pipelines is specified in the $TCBCONFIG_{PiN}$ field. See Section 4.4, "TCBCONFIG Register (Reg 0)" on page 24.

Figure 2-7 TF2 (Trace Format 2 Multi-Pipe)

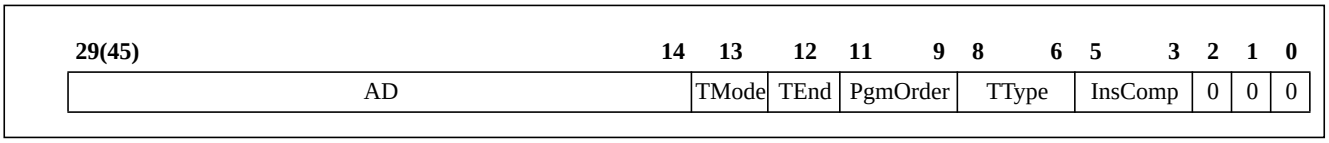


Figure 2-8 TF3 (Trace Format 3 Multi-Pipe)

TF4 for multi-pipe trace is defined as was the case for single-pipe trace. In the example in Figure 2-9, the TEnd bit (bit 12) is set, and the TType field (bits 8:6) is set to **DT** (100_2).

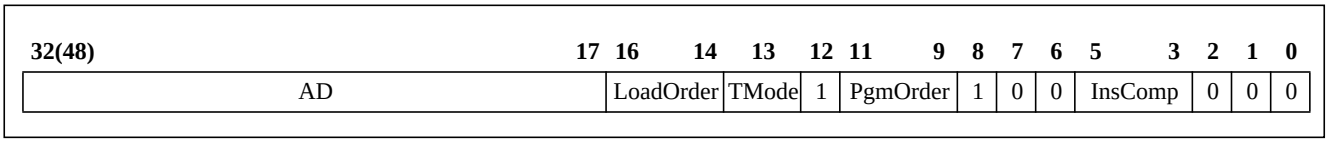


Figure 2-9 TF4 (Trace Format 4 Multi-Pipe)

Trace Word Format

After compression of data into the Trace Formats, the trace information must be streamed to either on-chip or off-chip dedicated trace memory. As seen in the previous chapter, each of the major Trace Formats are of different size. This complicates the efficient storage of this information into a fixed-width on-chip memory. It also complicates the transmission of this data through a fixed width interface to off-chip memory. To simplify the memory overhead and pin bandwidth issues, the Trace Formats are first gathered into Trace Words of regular width. This section describes these Trace Words.

3.1 Trace Word

A Trace Word (TW) is defined to be 64 bits wide. A TW has a 4 bit type indicator on bits [3:0], and regular TF's stacked up in the remaining 60 bits of the word. Figure 3-1 shows the 64-bit wide TW.

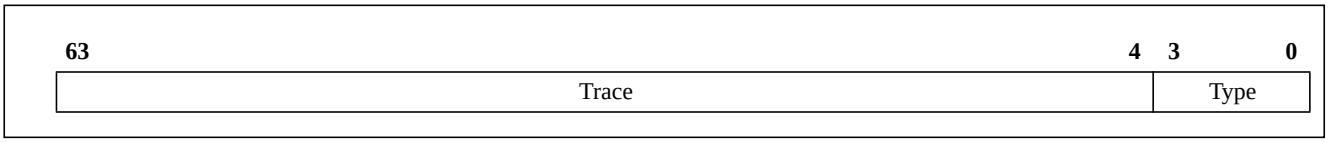


Figure 3-1 TW (Trace Word)

The **Trace** portion of a TW consists of one or more Trace Formats, TF1 through TF6. Note that trace formats TF1, TF2, TF5, and TF6 have a fixed size, while TF3 and TF4 can vary in size. The size of formats TF3 and TF4 is based on the number of *PDO_AD* bits on the PDtrace™ interface. A further optimization is possible with an address on *PDO_AD*. That is, the redundant sign bits (in the upper address bits) can be optionally chopped from the formats, especially if the format straddles two TWs. This happens when *PDO_TType* is set to **TPC**, **TLA**, or **TSA**, TEnd is set to 1, and TMode is set to 0.

The processor mode is traced as a TF3 with *PDO_TType* set to **TMOAS**. A **TMOAS** always uses 16 bits of AD, regardless of the PDO_AD bus width. Therefore, when *PDO_TType* is set to **TMOAS**, the TF3 format in a TW is defined to always have exactly 16 valid bits on the AD field, regardless of the $TCBCONTROLA_{ADW}$ field.

A TW is built by pushing in the TF's back to back until all 60 bits of the **Trace** field are used. If the last TF does not fit in **Trace**, it spills to the first bits of the **Trace** field in the next TW. The **Type** indicator is used to indicate where the first new TF starts in the new **Trace** field. This indirectly indicates the number of bits used to complete theTF from the previous TW.

Sometimes, when a TF cannot be completed in the remaining bits of a TW_n, it is more efficient to discard those bits of the TW_n and simply repeat all of them in the following TW_{n+1}. This is indicated in TW_{n+1} by setting **Type** to 1. When **Type** is 1, the first new TF of a TW starts at bit 0 in the **Trace** field. Since the previous TW_n ended with an uncompleted TF, a Type of 1 in TW_{n+1} instructs the decode software to discard the uncompleted TF in TW_n. Table 3-1 describes the word types for the TW.

Table 3-1 Trace Word Type field description

Decimal value of the Type field	The first new TF starts at this bit in the Trace field	Description
0	N/A	This TW does not carry any trace information. The **Trace** field is set to all zeroes. In the off-chip interface, the **Trace** field can be truncated to make the TW fit the bit-width of the off-chip interface. For on-chip trace, this TW is not to be stored in memory.
1	0	This indicates a situation where a new TF is started at the beginning of this TW. This can happen when: (1) a new trace is begun, (2) the TF in the previous TW was completed, and (3) an incomplete TF at the end of the previous TW is discarded. If the last trace format of the previous TW was a TF3 with: TType set to **TPC**, **TLA** or **TSA**, TEnd set to 1 and TMode set to 0, and with at least one AD bit, then that is considered a completed TF format, and no bits are discarded from the previous TW.
2 - 14	(Type - 1) * 4	The partial TF from the previous TW is completed in this TW in the bits available before the first new TF, i.e., bits 0..((Type -1)*4)-1) in the Trace field. If extra bits are available after completing the straddling TF, the rest of the bits until the first new TF start are undefined. TF3 formats sending the last part of a relative address are allowed to cut the AD bits to only show the needed sign bits. This enables compression of sign-extended *PDO_AD* bits when the TF3 straddles a TW.
15	No new TF[1]	The TF started in the previous TW could not be completed within 54 bits[2]. It might complete in this TW. But if it does not complete, then the next TW will have a **Type** value higher than one.
[1]: This will never happen with the currently defined TF's, as the longest TF is only 49 bits wide. [2]: 54 bits is the maximum allowable bits used to complete a TF from a previous TW_{n-1}, if a new one is to start in TW_n. This is so because a **Type** value of 14 indicates the maximum bit position (bit 54) in the **Trace** field, where a new TF will start.		

As an example of how a TW is built, consider the trace sequence shown in Table 3-2. In this example, the *PDO_AD* bus is assumed to be 16 bits wide (a zero value for $TCBCONTROLA_{ADW}$).

Table 3-2 Example Trace sequence

Cycle #	Trace Format	Cycle #	Trace Format
1	TF3 (16 significant AD bits)	2	TF3 (16 significant AD bits)
3	TF2	4	TF1
5	TF1	6	TF1
7	TF1	8	TF2
9	TF2	10	TF1
12	TF2	11	TF2
13	TF2	14	TF1
15	TF3 (5 significant AD bits)	16	TF1
17	TF2	18	TF2
19	TF2	20	TF2

Table 3-2 Example Trace sequence (Continued)

Cycle #	Trace Format	Cycle #	Trace Format
21	TF3 (11 significant AD bits)	22	TF1
23	TF3 (6 significant AD bits)	24	TF6 (Stop indicator)

The TF sequence in Table 3-2 will create the set of TW's shown in Figure 3-2. The grayed out boxes with a "u" are unused bits. Grayed out boxes with an "s", show redundant sign-bits from a TF3 format. These sign-bits could not be compressed out, and must be included as part of the AD field. A "1" indicates the single bit of 1 in a TF1 format.

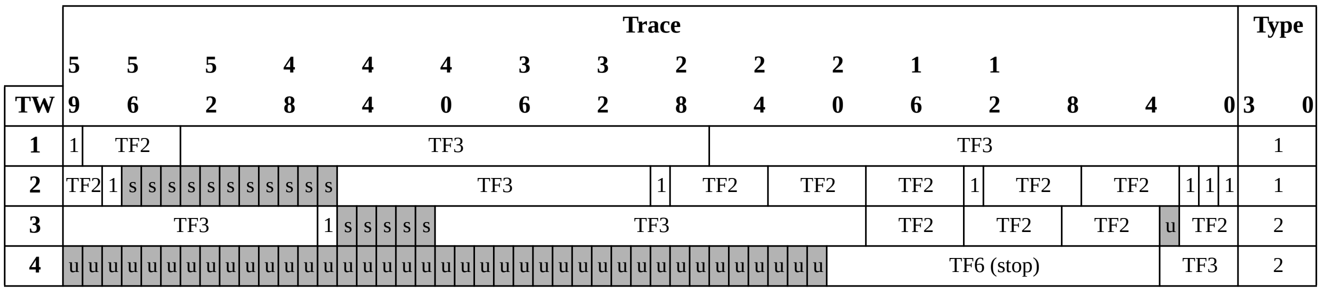


Figure 3-2 Trace Word from Example Trace in Table 3-2

In the example in Figure 3-2, the TF3 straddling TW_3/TW_4 have had insignificant sign bits cut from the full TF3 format. It is optional for TCB hardware to do this extra compression of TF3 formats, but TW decode software must always be designed to handle this extra compression.

3.1.1 Cycle Inaccurate Trace

The TF1 format is needed only when a sequence of the trace must show cycle-by-cycle behavior of the processor without missing any cycles. When the trace regeneration software only needs to show the sequence of instructions executed, the TF1 format which shows processor stall cycles can be omitted. In this latter situation, an additional optimization removes bit zero on the other TF's before storing to trace memory. The example trace sequence in Table 3-2 will then produce the TW's shown in Figure 3-3. Note that, to reconstruct the trace accurately, external software must know what type of tracing was enabled at the TCB.

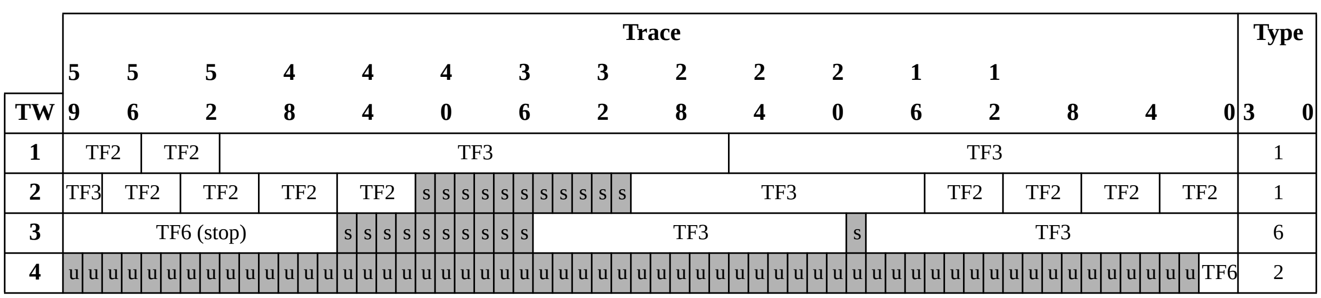


Figure 3-3 Trace Word from Example Trace in Table 3-2 (No TF1 trace)

In the example shown in Figure 3-3, the TF3 straddling TW_2/TW_3 have had insignificant sign bits cut from the full TF3 format. It is optional for TCB hardware to make this extra compression of TF3 format, but TW decode software must be able to handle this.

Additionally when not tracing for cycle accurate information, the TF6 formats TCBcode 0001 and 0101 are omitted from the Trace Words (not shown in Figure 3-2 and Figure 3-3). Cycle accurate versus cycle inaccurate tracing in controlled by the $TCBCONTROLB_{CA}$ bit.

3.1.1.1 Trace Word collection.

Figure 3-4 shows how the TCB builds the Trace Words from the Trace Formats cycle by cycle, as the PDtrace interface sends trace information. Trace Words from Figure 3-2 are used.

Cycle	TW	Trace (bits 59 … 0)	Type (3 … 0)
1	1	free … TF3	1
2	1	free … TF3 … TF3	1
3	1	f TF2 … TF3 … TF3	1
4	1	1 TF2 … TF3 … TF3	1
5	2	free … 1	1
6	2	free … 1 1	1
7	2	free … 1 1 1	1
8	2	free … TF2 1 1 1	1
9	2	free … TF2 TF2 1 1 1	1
10	2	free … 1 TF2 TF2 1 1 1	1
11	2	free … TF2 1 TF2 TF2 1 1 1	1
12	2	free … TF2 TF2 1 TF2 TF2 1 1 1	1
13	2	free … TF2 TF2 TF2 1 TF2 TF2 1 1 1	1
14	2	free … 1 TF2 TF2 TF2 1 TF2 TF2 1 1 1	1
15	2	free s s s s s s s s s s TF3 1 TF2 TF2 TF2 1 TF2 TF2 1 1 1	1
16	2	free 1 s s s s s s s s s s TF3 1 TF2 TF2 TF2 1 TF2 TF2 1 1 1	1
17	2	TF2 1 s s s s s s s s s s TF3 1 TF2 TF2 TF2 1 TF2 TF2 1 1 1	1
18	3	free … u TF2	2
19	3	free … TF2 u TF2	2
20	3	free … TF2 TF2 TF2 u TF2	2
21	3	free … s s s s s TF3 … TF2 TF2 TF2 u TF2	2
22	3	free … 1 s s s s s TF3 … TF2 TF2 TF2 u TF2	2
23	3	TF3 … 1 s s s s s TF3 … TF2 TF2 TF2 u TF2	2
24	4	free … TF3	2
		u … u TF6 (stop) TF3	2

Figure 3-4 Cycle by cycle Trace Word from Example Trace in Table 3-2

3.2 End of Trace indication.

In the examples in the previous section, the Trigger TF6 (stop: TCBcode == 0100) was used to indicate an End Trigger and this implied an end to the tracing as well. This stop trigger deasserts $TCBCONTROLB_{EN}$ and the TCB flushes out

the current TW. However, the *TCBCONTROLB*$_{EN}$ bit can be deasserted for other reasons, and this trace end must be indicated externally using a different mechanism to distinguish this from the end-trigger case. The recommended method to accomplish this is to let the TCB fill the un-used bits in the last TW with zeroes. Note that nine bits of consecutive zeroes in the Trace field will be identified as a TF3 with no information, that is, InsComp and TType are both zero. This will never be ordinarily generated by the Trace Format generator, and can therefore be used as a end-of-trace indicator.

If less than nine bits remain in the last TW, then an incomplete TF is detected by trace software. After that no additional TW are generated by the TCB. This should not be a problem for trace re-generating software, as this is just like any other arbitrary cut in the trace stream.

3.3 On-chip Trace Memory Format

The on-chip trace memory is defined to be a 64-bit wide memory. The TW's defined in Section 3.1, "Trace Word", are stored in consecutive address locations. The trace memory is only written when a full TW is available, hence a new TW might not be written each cycle, since a new TW might not be created each cycle.

The memory image will exactly match the TW sequence shown in Figure 3-2 on page 11 or Figure 3-3 on page 11, depending on whether TF1 formats are included.

3.4 Probe Trace Word transmission

Please see Chapter 8, "TCtrace IF," on page 43 for a detailed description of the TCtrace IF and Chapter 9, "Probe IF," on page 47 for a detailed description of the PIB module.

The Probe interface can support a *TR_DATA* bus width of 4, 8, or 16 bits. When a TW is ready to be sent, it is put on the *TC_Data* pins to the PIB. The PIB will feed the TW through on the available *TR_DATA* pins, starting with *TC_Data[n:0]* on the *TR_DATA[n:0]* utilized pins. Depending on the value of n, this will take 16, 8, or 4 transmissions. If a clock multiplier is used in the PIB, then 2, 4, 8, or 16 transmissions can be completed in one core clock cycle.

As long as no new TW is available for transmission, the *TC_Data* bus will show all zeros, allowing the PIB to keep transmitting this on the *TR_DATA* bits to also show all zeros.

On an 8 pin *TR_DATA* trace interface, running at core-clock frequency, the trace from the TW's in Figure 3-4 will look as shown in Figure 3-5 on the Probe IF. This assumes sufficient buffering to hold the TW's in the TCB when they become available for transmission, and a latency of one clock before the first part of an available TW on the *TC_data* bus, appears on the *TR_DATA* pins.

Cycle	TR_DATA[7:0]
1	zero
2	zero
3	zero
4	zero
5	TW_1[7:0]
6	TW_1[15:8]
7	TW_1[23:16]
8	TW_1[31:24]
9	TW_1[39:32]
10	TW_1[47:40]

Cycle	TR_DATA[7:0]
11	TW_1[55:48]
12	TW_1[63:56]
13	zero
14	zero
15	zero
16	zero
17	zero
18	TW_2[7:0]
19	TW_2[15:8]
20	TW_2[23:16]

Cycle	TR_DATA[7:0]
21	TW_2[31:24]
22	TW_2[39:32]
23	TW_2[47:40]
24	TW_2[55:48]
25	TW_2[63:56]
26	TW_3[7:0]
27	TW_3[15:8]
28	TW_3[23:16]
29	TW_3[31:24]
30	TW_3[39:32]

Cycle	TR_DATA[7:0]
31	TW_3[47:40]
32	TW_3[55:48]
33	TW_3[63:56]
34	TW_4[7:0]
35	TW_4[15:8]
36	TW_4[23:16]
37	zero
38	zero
39	zero
40	zero

Figure 3-5 Cycle by cycle *TR_DATA* (8-bit) of Example Trace in Table 3-2

The probe sampling the *TR_DATA* pins should look for a non-zero transmission. When that happens, the following bits up to a collective count of 64-bits (i.e. including the first non-zero 4/8/16-bit value) will form a TW. After 64-bits, the probe should re-start looking for a new non-zero transmission. A non zero transmission can start at any time after a full TW is received.

Trace Control Block Registers

The TCB uses several registers to control its operation. These registers are accessed via the EJTAG TAP interface. This chapter describes these registers in detail. These registers are shown in Table 4-1 and Table 4-2.

Table 4-1 Registers in the Trace Control Block

Register Name	EJTAG TAP controller instruction value	Description
TCBCONTROLA	0x10	Control register in the TCB mainly used for controlling the trace input signals to the core on the PDtrace interface.
TCBCONTROLB	0x11	Control register in the TCB that is mainly used to specify what to do with the trace information. The REG [25:21] field in this register specifies the number of the TCB internal register accessed by the TCBDATA register. A list of all the registers that can be accessed by the TCBDATA register is shown in.
TCBDATA	0x12	This is used to access registers specified by the REG field in the TCBCONTROLB register.

Table 4-2 Registers selected by *TCBCONTROLB*$_{REG}$(acessed through TCBDATA).

REG[4:0]	Register Selected	Register Description	Compliance
0	*TCBCONFIG*	TCB Configuration register--holds information about the hardware configuration of the TCB.	Required
1-3	Reserved	Reserved for future use.	Reserved
4	*TCBTW*	Trace Word read register. This register holds the Trace Word just read from on-line trace memory.	Required if on-chip memory exists.
5	*TCBRDP*	Trace Word Read pointer. It points to the location in the on-line trace memory where the next Trace Word will be read. A TW read has the side-effect of post-incrementing this register value to point to the next TW location. (A maximum value wraps the address around to the beginning of the trace memory).	
6	*TCBWRP*	Trace Word Write pointer. It points to the location in the on-line trace memory where the next new Trace Word will be written.	
7	*TCBSTP*	Trace Word Start Pointer. This points to the location of the oldest TW in the on-line trace memory.	
8-15	Reserved	Reserved for future use.	Reserved
16-23	*TCBTRIGx*	Trigger Control registers 0-7 are used to specify some conditions that cause the firing of triggers, and to control the resulting action.	Optional
24-31	Reserved	Reserved for future use.	Reserved

4.1 *TCBCONTROLA* Register

The trace output from the processor on the PDtrace interface can be controlled by the trace input signals to the processor from the TCB. The TCB uses a control register, TCBCONTROLA, whose values are used to change the signal values on the PDtrace input interface. External software (i.e., debugger), can therefore manipulate the trace output by writing the TCBCONTROLA register.

The *TCBCONTROLA* register is written by an EJTAG TAP controller instruction, *TCBCONTROLA* (0x10). See Ref. [1] for more details regarding new TAP instructions.

Compliance: This register is required.

The format of the *TCBCONTROLA* register is shown below, and the fields are described in Table 4-3.

***TCBCONTROLA* Register Format**

31 27	26	25 24	23	22 20	19	18	17	16	15	14	13	12 5	4	3 1	0
Impl	0	VModes	ADW	SyP	TB	IO	D	E	S	K	U	ASID	G	Mode	On

Table 4-3 *TCBCONTROLA* Register Field Descriptions

Fields		Description	Read/ Write	Reset State	Compliance
Name	**Bits**				
Impl	31:27	This field is reserved for implementation specific use. Refer to the processor specification for the format and definition of this field.		Undefined	Optional
0	26	Reserved for future use. Must be written as zero; returns zero on read.	0	0	Reserved
VModes	25:24	This field specifies the type of tracing that is supported by the processor, as follows: **Encoding** / **Meaning** 00 — PC tracing only 01 — PC and load and store address tracing only 10 — PC, load, and store address, and load and store data. 11 — Reserved This field is preset to the value of *PDO_ValidModes*.	R	Preset	Required
ADW	23	*PDO_AD* bus width. 0: The *PDO_AD* bus is 16 bits wide. 1: The *PDO_AD* bus is 32 bits wide.	R	Preset	Required

Table 4-3 *TCBCONTROLA* Register Field Descriptions (Continued)

<table>
<tr><th colspan="2">Fields</th><th rowspan="2">Description</th><th rowspan="2">Read/ Write</th><th rowspan="2">Reset State</th><th rowspan="2">Compliance</th></tr>
<tr><th>Name</th><th>Bits</th></tr>
<tr><td>SyP</td><td>22:20</td><td>Used to indicate the synchronization period.

The period (in cycles) between which the periodic synchronization information is to be sent is defined as shown in the table below, when the trace buffer is either on-chip or off-chip (as determined by the TCBCONTROLB_{OfC} bit).

<table><tr><th>SyP</th><th>On-chip</th><th>Off-chip</th></tr><tr><td>000</td><td>2^2</td><td>2^7</td></tr><tr><td>001</td><td>2^3</td><td>2^8</td></tr><tr><td>010</td><td>2^4</td><td>2^9</td></tr><tr><td>011</td><td>2^5</td><td>2^{10}</td></tr><tr><td>100</td><td>2^6</td><td>2^{11}</td></tr><tr><td>101</td><td>2^7</td><td>2^{12}</td></tr><tr><td>110</td><td>2^8</td><td>2^{13}</td></tr><tr><td>111</td><td>2^9</td><td>2^{14}</td></tr></table>
This field defines the value on the PDI_SyncPeriod signal.</td><td>R/W</td><td>100</td><td>Required</td></tr>
<tr><td>TB</td><td>19</td><td>Trace All Branches. This signal is used to indicate that the core must trace either full or incremental PC values for all branches. Not just the unpredictable ones.

This field defines the value on the PDI_TraceAllBranch signal.</td><td>R/W</td><td>Undefined</td><td>Required</td></tr>
<tr><td>IO</td><td>18</td><td>Inhibit Overflow. This signal is used to indicate to the core trace logic that slow but complete tracing is desired. Hence, the core tracing logic must not allow a FIFO overflow and discard trace data. This is achieved by stalling the pipeline when the FIFO is nearly full so that no trace records are ever lost.

This field defines the value on the PDI_InhibitOverflow signal.</td><td>R/W</td><td>Undefined</td><td>Required</td></tr>
<tr><td>D</td><td>17</td><td>When set to one, this enables tracing in Debug mode, i.e., when the DM bit is one in the Debug register. For trace to be enabled in Debug mode, the On bit must be one and either the G bit must be one, or the current process must match the ASID field in this register.

When set to zero, trace is disabled in Debug mode, irrespective of other bits.

This field defines the value on the PDI_DM signal.</td><td>R/W</td><td>Undefined</td><td>Required</td></tr>
<tr><td>E</td><td>16</td><td>This controls when tracing is enabled. When set, tracing is enabled when either the EXL or ERL bits in the Status register is one, provided that the On bit (bit 0) is also set, and either the G bit is set, or the current process ASID matches the ASID field in this register.

This field defines the value on the PDI_E signal.</td><td>R/W</td><td>Undefined</td><td>Required</td></tr>
</table>

Table 4-3 *TCBCONTROLA* Register Field Descriptions (Continued)

<table>
<tr><th colspan="2">Fields</th><th rowspan="2">Description</th><th rowspan="2">Read/ Write</th><th rowspan="2">Reset State</th><th rowspan="2">Compliance</th></tr>
<tr><th>Name</th><th>Bits</th></tr>
<tr><td>S</td><td>15</td><td>When set, this enables tracing when the core is in Supervisor mode as defined in the MIPS32 or MIPS64 architecture specification. This is provided the On bit (bit 0) is also set, and either the G bit is set, or the current process ASID matches the ASID field in this register.

This field defines the value on the PDI_S signal.</td><td>R/W</td><td>Undefined</td><td>Required</td></tr>
<tr><td>K</td><td>14</td><td>When set, this enables tracing when the On bit is set and the core is in Kernel mode. Unlike the usual definition of Kernel Mode, this bit enables tracing only when the ERL and EXL bits in the Status register are zero. This is provided the On bit (bit 0) is also set, and either the G bit is set, or the current process ASID matches the ASID field in this register.

This field defines the value on the PDI_K signal.</td><td>R/W</td><td>Undefined</td><td>Required</td></tr>
<tr><td>U</td><td>13</td><td>When set, this enables tracing when the core is in User mode as defined in the MIPS32 or MIPS64 architecture specification. This is provided the On bit (bit 0) is also set, and either the G bit is set, or the current process ASID matches the ASID field in this register.

This field defines the value on the PDI_U signal.</td><td>R/W</td><td>Undefined</td><td>Required</td></tr>
<tr><td>ASID</td><td>12:5</td><td>The ASID field to match when the G bit is zero. When the G bit is one, this field is ignored.

This field defines the value on the PDI_ASID signal.</td><td>R/W</td><td>Undefined</td><td>Required</td></tr>
<tr><td>G</td><td>4</td><td>When set, this implies that tracing is to be enabled for all processes, provided that other enabling functions (like U, S, etc.,) are also true.

This field defines the value on the PDI_G signal.</td><td>R/W</td><td>Undefined</td><td>Required</td></tr>
<tr><td>Mode</td><td>3:1</td><td>When tracing is turned on, this signal specifies what information is to be traced by the core

<table>
<tr><th>Mode</th><th>Trace Mode</th></tr>
<tr><td>000</td><td>Trace PC</td></tr>
<tr><td>001</td><td>Trace PC and load address</td></tr>
<tr><td>010</td><td>Trace PC and store address</td></tr>
<tr><td>011</td><td>Trace PC and both load/store addresses</td></tr>
<tr><td>100</td><td>Trace PC and load data (optional for all PDtrace specification revisions less than 3.0 and supporting TCBs).</td></tr>
<tr><td>101</td><td>Trace PC and load address and data</td></tr>
<tr><td>110</td><td>Trace PC and store address and data</td></tr>
<tr><td>111</td><td>Trace PC and both load/store address and data</td></tr>
</table>

The VModes field determines which of these encodings are supported by the processor. The operation of the processor is UNPREDICTABLE if Mode is set to a value which is not supported by the processor

This field defines the value on the PDI_TraceMode signal.</td><td>R/W</td><td>Undefined</td><td>Required</td></tr>
</table>

Table 4-3 *TCBCONTROLA* Register Field Descriptions (Continued)

Fields		Description	Read/ Write	Reset State	Compliance
Name	Bits				
On	0	This is the global trace enable switch to the core. When zero, tracing from the core is always disabled, unless enabled by core internal software override of the *PDI_xx* input pins. When set to one, tracing is enabled whenever the other enabling functions are also true. This field defines the value on the *PDI_TraceOn* signal.	R/W	0	Required

4.2 *TCBCONTROLB* Register

The TCB includes a second control register, *TCBCONTROLB* (0x11). This register generally controls what happens to the trace information once it arrives at the TCB.

Compliance: This register is required.

The format of the *TCBCONTROLB* register is shown below, and the fields are described in Table 4-4.

***TCBCONTROLB* Register Format**

31	30–28	27–26	25–21	20	19–17	16	15	14	13–12	11–10	9–8	7	6–3	2	1	0
WE	Impl	0	REG	WR	0	RM	TR	BF	TM	0	CR	Cal	0	CA	OfC	EN

Table 4-4 *TCBCONTROLB* Register Field Descriptions

Fields		Description	Read/ Write	Reset State	Compliance
Name	Bits				
WE	31	Write Enable. Only when set to 1 will the other bits be written in *TCBCONTROLB*. This bit will always read 0.	R	0	Required
Impl	30:28	This field is reserved for implementations. Refer to the processor specification for the format and definition of this field.		Undefined	Optional
0	27:26	Reserved for future use. Must be written as zero; returns zero on read.	0	0	Reserved
REG	25:21	Register select: This field specifies the register, (one among the set of registers in Table 4-2), to access through the *TCBDATA* register.	R/W	0	Required

Table 4-4 *TCBCONTROLB* Register Field Descriptions (Continued)

Fields		Description	Read/ Write	Reset State	Compliance
Name	Bits				
WR	20	The write register field, when set, allows the register selected by the REG field to be written as well as read when *TCBDATA* is accessed. Otherwise, the selected register is only read. Note that a JTAG register cannot be only written, it is always read and written. Therefore, a register that has a side-effect on read (see Section 4.6, "TCBRDP Register (Reg 5)"), will have the same side-effect when written, since a read also happens on a write. Hence, it is specified that when this field WR is set, it is implementation dependent whether a side-effect of a read will occur when writing.	R/W	0	Required
0	19:17	Reserved for future use. Must be written as zero; returns zero on read.	0	0	Reserved
RM	16	Read on-chip trace memory. When written to 1, the read address-pointer of the on-chip memory in register *TCBRDP* is set to point to the oldest memory location written since the last reset of pointers. Subsequent access to the *TCBTW* register (through the *TCBDATA* register), will automatically increment the read pointer in register *TCBRDP* after each read. When the write pointer is reached, this bit is automatically reset to 0, and the *TCBTW* register will read all zeros. Once set to 1, writing 1 again will have no effect. The bit is reset by setting the TR bit or by reading the last Trace word in *TCBTW.*	R/W1	0	Required if on-chip memory exists. Otherwise reserved.
TR	15	Trace memory reset. When written to one, the address pointers for the on-chip trace memory *TCBRDP and TCBWRP* are reset to zero. Also the RM and BF bits are reset to 0. This bit is automatically reset back to 0, when the reset specified above is completed.	R/W1	0	Required if on-chip memory exists. Otherwise reserved.
BF	14	Buffer Full indicator that the TCB uses to communicate to external software that the on-chip trace memory is full. Note that this applies only in the situation that the on-chip trace memory is being deployed in the **trace-from** and **trace-to** mode. (See Chapter 10, “On-Chip Trace Memory.”) This bit is cleared when writing a 1 to the TR bit	R	0	Required if on-chip memory exists. Otherwise reserved.

Table 4-4 *TCBCONTROLB* Register Field Descriptions (Continued)

<table>
<tr><th colspan="2">Fields</th><th rowspan="2">Description</th><th rowspan="2">Read/ Write</th><th rowspan="2">Reset State</th><th rowspan="2">Compliance</th></tr>
<tr><th>Name</th><th>Bits</th></tr>
<tr><td>TM</td><td>13:12</td><td>Trace Mode. This field determines how the trace memory is filled when using the simple-break control in the PDtrace™ IF to start or stop trace.

<table><tr><th>TM</th><th>Trace Mode</th></tr><tr><td>00</td><td>Trace-To</td></tr><tr><td>01</td><td>Trace-From</td></tr><tr><td>10</td><td>Reserved</td></tr><tr><td>11</td><td>Reserved</td></tr></table>
In Trace-To mode, the on-chip trace memory is filled, continuously wrapping around, overwriting older Trace Words, as long as there is trace data coming from the core.

In Trace-From mode, the on-chip trace memory is filled from the point that PDO_IamTracing is asserted, and until the on-chip trace memory is full.

In both cases, de-asserting the EN bit in this register will also stop fill to the trace memory.

If a TCBTRIGx trigger control register is used to start/stop tracing, then this field should be set to Trace-To mode.</td><td>R/W</td><td>0</td><td>Required

if on-chip memory exists.

Otherwise reserved.</td></tr>
<tr><td>0</td><td>11</td><td>Reserved for future use. Must be written as zero; returns zero on read.</td><td>0</td><td>0</td><td>Reserved</td></tr>
<tr><td>CR</td><td>10:8</td><td>Off-chip Clock Ratio. Writing this field, sets the ratio of the core clock to the off-chip trace memory interface clock. The clock-ratio encoding is shown in Table 4-5.

Remark: For example, a clock ratio of 1:2 implies a two times slow down of the Probe interface clock to the core clock. But, one data packet is sent per core clock rising edge, while a data packet is sent on every edge of the Probe interface clock, since the Probe interface works in double data rate (DDR) mode.</td><td>R/W</td><td>100</td><td>Required

if off-chip trace interface exists.

Otherwise reserved.</td></tr>
</table>

Table 4-4 *TCBCONTROLB* Register Field Descriptions (Continued)

<table>
<tr><th colspan="2">Fields</th><th rowspan="2">Description</th><th rowspan="2">Read/ Write</th><th rowspan="2">Reset State</th><th rowspan="2">Compliance</th></tr>
<tr><th>Name</th><th>Bits</th></tr>
<tr><td>Cal</td><td>7</td><td>Calibrate off-chip trace interface.

If set, the off-chip trace pins will produce the following pattern in consecutive trace clock cycles. If more than 4 data pins exist, the pattern is replicated for each set of 4 pins. The pattern repeats from top to bottom until the Cal bit is de-asserted.

<table>
<tr><th></th><th colspan="4">Calibrations pattern</th></tr>
<tr><th></th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td rowspan="14">This pattern is replicated for every 4 bits of TR_DATA pins.</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td></tr>
</table>

Note: The clock source of the TCB and PIB must be running.</td><td>R/W</td><td>0</td><td>Required

if off-chip trace interface exists.

Otherwise reserved.</td></tr>
<tr><td>0</td><td>6:3</td><td>Reserved for future use. Must be written as zero; returns zero on read.</td><td>0</td><td>0</td><td>Reserved</td></tr>
<tr><td>CA</td><td>2</td><td>Cycle accurate trace.

When set to 1 the trace will include stall information. When set to 0 the trace will exclude stall information, and remove bit zero from all transmitted TF’s.

The stall information included/excluded is:

• TF6 formats with TCBcode 0001 and 0101.

• All TF1 formats except within the context of multi-pipe processor tracing (when it is used for individual pipes within the sequence of pipe outputs).</td><td>R/W</td><td>0</td><td>Required</td></tr>
<tr><td>OfC</td><td>1</td><td>If set to 1, trace is sent to off-chip memory using TR_DATA pins.

If not set, trace info is sent to on-chip memory.

This bit is read only if either off-chip or on-chip option exists.</td><td>R/W</td><td>Preset</td><td>Required</td></tr>
</table>

Table 4-4 *TCBCONTROLB* Register Field Descriptions (Continued)

Fields		Description	Read/ Write	Reset State	Compliance
Name	**Bits**				
EN	0	Enable trace. This is the master enable for trace to be generated from the TCB. This bit can be set or cleared, either by writing this register or from a start/stop/center trigger. When set to “1”, trace information is sampled on the *PDO_xx* pins. Trace Words are generated and sent to either on-chip memory or to the Trace Probe. The target of the trace is selected by the OfC bit. When set to “0”, trace information on the PDO_xx pins is ignored. A potential TF6-stop (from a stop trigger) is generated as the last information, and the TCB pipe-line is flushed, and trace output is stopped.	R/W	0	Required

Table 4-5 Clock Ratio encoding of the CR field

CR/CRMin/CRMax	Clock Ratio
000	8:1 (Trace clock is eight times that of core clock)
001	4:1 (Trace clock is four times that of core clock)
010	2:1 (Trace clock is double that of core clock)
011	1:1 (Trace clock is same as core clock)
100	1:2 (Trace clock is one half of core clock)
101	1:4 (Trace clock is one fourth of core clock)
110	1:6 (Trace clock is one sixth of core clock)
111	1:8 (Trace clock is one eighth of core clock)

4.3 *TCBDATA* Register

The *TCBDATA* register (0x12) is used to access the registers defined by the $TCBCONTROLB_{REG}$ field, see Table 4-2. Regardless of which register or data entry is accessed through *TCBDATA*, the register is only written if the $TCBCONTROLB_{WR}$ bit is set. For read only registers the $TCBCONTROLB_{WR}$ is a don’t-care.

Compliance: This register is required.

The format of the *TCBDATA* register is shown below, and the field is described in Table 4-6. The width of *TCBDATA* is 64 bits when on-chip trace words (TWs) are accessed (*TCBTW* access).

***TCBDATA* Register Format**

31(63)	**0**
Data	

Table 4-6 *TCBDATA* Register Field Descriptions

Fields		Description	Read/Write	Reset State	Compliance
Names	Bits				
Data	31:0 63:0	Register fields or data as defined by the $TCBCONTROLB_{REG}$ field	Only writable if $TCBCONTROLB_{WR}$ is set	0	Required

4.4 *TCBCONFIG* Register (Reg 0)

The *TCBCONFIG* register holds hardware configuration information in the TCB.

Compliance: This register is required.

***TCBCONFIG* Register Format**

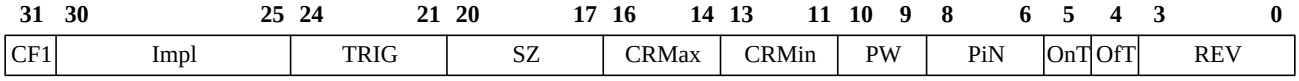


Table 4-7 *TCBCONFIG* Register Field Descriptions

Fields		Description	Read/ Write	Reset State	Compliance
Name	Bits				
CF1	31	This bit is set if a *TCBCONFIG1* register exists. In this revision, *TCBCONFIG1* does not exist, and this bit reads zero.	R	0	Required
Impl	30:25	This field is reserved for implementations. Refer to the processor specification for the format and definition of this field.	0	Undefined	Optional
TRIG	24:21	Number of triggers implemented. This also indicates the number of *TCBTRIGx* registers that exist.	R	Legal values are 0 - 8	Required
SZ	20:17	On-chip trace memory size. This field holds the encoded size of the on-chip trace memory. The size in bytes is given by $2^{(SZ+8)}$. I.e., the lowest value is 256 bytes, and the highest is 8Mb.	R	Preset	Required if on-chip memory exists. Otherwise reserved.
CRMax	16:14	Off-chip Maximum Clock Ratio. This field indicates the maximum ratio of the core clock to the off-chip trace memory interface clock. The clock-ratio encoding is shown in Table 4-5.	R	Preset	Required if off-chip trace interface exists. Otherwise reserved.
CRMin	13:11	Off-chip Minimum Clock Ratio. This field indicates the minimum ratio of the core clock to the off-chip trace memory interface clock. The clock-ratio encoding is shown in Table 4-5.	R	Preset	Required if off-chip trace interface exists. Otherwise reserved.

Table 4-7 *TCBCONFIG* Register Field Descriptions (Continued)

<table>
<tr><th colspan="2">Fields</th><th rowspan="2">Description</th><th rowspan="2">Read/ Write</th><th rowspan="2">Reset State</th><th rowspan="2">Compliance</th></tr>
<tr><th>Name</th><th>Bits</th></tr>
<tr><td>PW</td><td>10:9</td><td>Probe Width: Number of bits available on the off-chip trace interface TR_DATA pins. The number of TR_DATA pins is encoded, as shown in the table.

<table><tr><th>PW</th><th>Number of bits used on TR_DATA</th></tr><tr><td>00</td><td>4 bits</td></tr><tr><td>01</td><td>8 bits</td></tr><tr><td>10</td><td>16 bits</td></tr><tr><td>11</td><td>reserved</td></tr></table>
This field is preset based on input signals to the TCB and the actual capability of the TCB.</td><td>R</td><td>Preset</td><td>Required

if off-chip trace interface exists.

Otherwise reserved.</td></tr>
<tr><td>PiN</td><td>8:6</td><td>Pipe number.

For single-pipeline processors this field must read 0.

For multi-pipeline processor, this field indicates the number of pipes which are traced. If non-zero this also indicates, that the 3-bit PgmOrder field is included in the TF2, TF3 and TF4 Trace Formats, as shown in Figure 2-7, Figure 2-8 and Figure 2-9 on page 7.

The table below indicates the number of bits in PgmOrder for the possible values of PiN.

<table><tr><th>PiN</th><th>Number of Pipes traced</th><th>PgmOrder field included in the TF2, TF3 and TF4 Trace Formats</th></tr><tr><td>000</td><td>1</td><td>No</td></tr><tr><td>001</td><td>2</td><td rowspan="7">Yes</td></tr><tr><td>010</td><td>3</td></tr><tr><td>011</td><td>4</td></tr><tr><td>100</td><td>5</td></tr><tr><td>101</td><td>6</td></tr><tr><td>110</td><td>7</td></tr><tr><td>111</td><td>8</td></tr></table></td><td>R</td><td>Preset</td><td>Required</td></tr>
<tr><td>OnT</td><td>5</td><td>When set, this bit indicates that on-chip trace memory is present. This bit is preset based on the selected option when the TCB is implemented.</td><td>R</td><td>Preset</td><td>Required</td></tr>
<tr><td>OfT</td><td>4</td><td>When set, this bit indicates that off-chip trace interface is present. This bit is preset based on the selected option when the TCB is implemented, and on the existence of a PIB module (TC_PibPresent asserted).</td><td>R</td><td>Preset</td><td>Required</td></tr>
<tr><td>REV</td><td>3:0</td><td>Revision of TCB. An implementation that conforms to the described architecture in this document must have revision 0.</td><td>R</td><td>0</td><td>Required</td></tr>
</table>

4.5 *TCBTW* Register (Reg 4)

The *TCBTW* register is used to read Trace Words from the on-chip trace memory. The TW read is the TW pointed to by the *TCBRDP* register. A side effect of reading the *TCBTW* register is that the *TCBRDP* register increments to the next TW in the on-chip trace memory. If *TCBRDP* is at the max size of the on-chip trace memory, the increment wraps back to address zero.

Compliance: Required if on-chip trace memory is implemented.

The format of the *TCBTW* register is shown below, and the field is described in Table 4-8.

***TCBTW* Register Format**

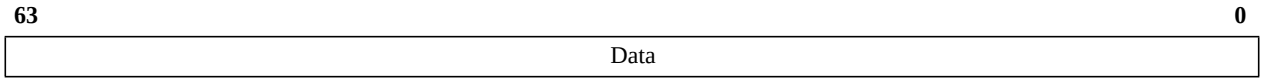


Table 4-8 *TCBTW* Register Field Descriptions

Fields		Description	Read/ Write	Reset State	Compliance
Names	Bits				
Data	63:0	Trace Word	R/W	0	Required

4.6 *TCBRDP* Register (Reg 5)

The *TCBRDP* register is the address pointer to on-chip trace memory. It points to the TW read when reading the *TCBTW* register. When writing the $TCBCONTROLB_{RM}$ bit to 1, this pointer is reset to the current value of *TCBSTP*.

Compliance: Required if on-chip trace memory is implemented.

The format of the *TCBRDP* register is shown below, and the field is described in Table 4-8. The value of n depends on the size of the on-chip trace memory. As the address points to a 64-bit TW, lower three bits are always zero.

***TCBRDP* Register Format**

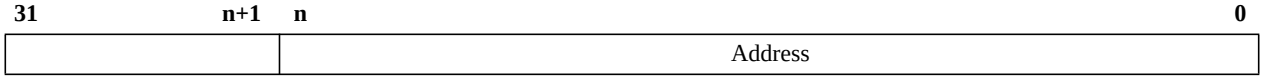


Table 4-9 *TCBRDP* Register Field Descriptions

Fields		Description	Read/ Write	Reset State	Compliance
Names	Bits				
Data	31:(n+1)	Reserved. Must be written zero, reads back zero.	0	0	Required
Address	n:0	Byte address of on-chip trace memory word.	R/W	0	Required

4.7 *TCBWRP* Register (Reg 6)

The *TCBWRP* register is the address pointer to on-chip trace memory. It points to the location where the next new TW for on-chip trace will be written.

Compliance: Required if on-chip trace memory is implemented.

The format of the *TCBWRP* register is shown below, and the field is described in Table 4-8. The value of n depends on the size of the on-chip trace memory. As the address points to a 64-bit TW, lower three bits are always zero.

***TCBWRP* Register Format**

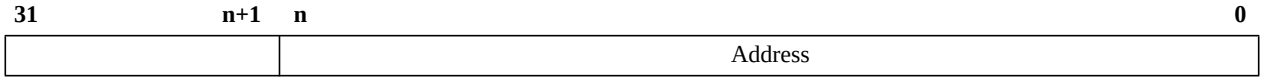


Table 4-10 *TCBWRP* Register Field Descriptions

Fields		Description	Read/ Write	Reset State	Compliance
Names	Bits				
Data	31:(n+1)	Reserved. Must be written zero, reads back zero.	0	0	Required
Address	n:0	Byte address of on-chip trace memory word.	R/W	0	Required

4.8 *TCBSTP* Register (Reg 7)

The *TCBSTP* register is the start pointer register. This register points to the on-chip trace memory address at which the oldest TW is located. This pointer is reset to zero when the $TCBCONTROLB_{TR}$ bit is written to 1. If a continuous trace to on-chip memory wraps around the on-chip memory, *TSBSTP* will have the same value as *TCBWRP*.

Compliance: Required if on-chip trace memory is implemented.

The format of the *TCBSTP* register is shown below, and the field is described in Table 4-8. The value of n depends on the size of the on-chip trace memory. As the address points to a 64-bit TW, lower three bits are always zero.

***TCBSTP* Register Format**

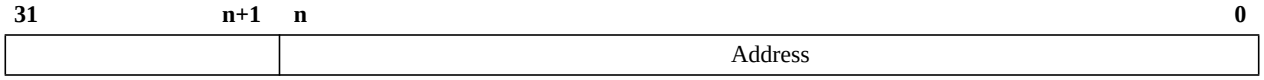


Table 4-11 *TCBSTP* Register Field Descriptions

Fields		Description	Read/ Write	Reset State	Compliance
Names	Bits				
Data	31:(n+1)	Reserved. Must be written zero, reads back zero.	0	0	Required
Address	n:0	Byte address of on-chip trace memory word.	R/W	0	Required

4.9 *TCBTRIGx* Register (Reg 16-23)

Eight Trigger Control registers are defined. Each register is named *TCBTRIGx*, where *x* is a single digit number from 0 to 7 (*TCBTRIG0* is Reg 16). The actual number of trigger registers implemented is defined in the $TCBCONFIG_{TRIG}$ field. An unimplemented register will read all zeros and writes are ignored.

Each Trigger Control register controls when an associated trigger is fired and the resulting action. Please also read Chapter 5, “Trigger Logic,” on page 33, for a detailed description of trigger logic issues.

Compliance: The number of implemented trigger registers must be equal to the number in $TCBCONFIG_{TRIG}$.

***TCBTRIGx* Register Format**

31 24	23	22 20	19 16	15	14	13 11	10 7	6	5	4	3 2	1	0
TCBinfo	Trace	Impl	0	CHTro	PDTro	Impl	0	DM	CHTri	PDTri	Type	FO	TR

Table 4-12 *TCBTRIGx* Register Field Descriptions

Fields		Description	Read/ Write	Reset State	Compliance
Names	**Bits**				
TCBinfo	31:24	TCBinfo to be used in a possible TF6 trace format when this trigger fires.	R/W	0	Required
Trace	23	When set, generate a TF6 trace information when this trigger fires. Use TCBinfo field for the TCBinfo of TF6 and use Type field for the two MSB of the TCBtype of TF6. The two LSB of TCBtype are 00. The write value of this bit always controls the action from the firing of this trigger. When this trigger fires, if another higher priority trigger fires simultaneously, then the action of this trigger can be suppressed. That is, the issue of the TF6 format would be suppressed. If this ever happens, this can be detected by reading the value of this field. If the Trace field was set to 1, and this trigger action was suppressed, then the read of this Trace field will return a 0. (Note that the read value is always 0 if the write value was 0). The read value of 0 indicating a suppressed trigger action is valid until the TCBTRIGx register is again written. That is, the read value is 0 if the trigger fires but the trigger action was ever suppressed, since the last write.	R/W	0	Required
Impl	22:20	These bits are reserved for implementation specific trigger actions (internal to the TCB). Refer to the processor specification for the format and definition of this field.		0	Optional
0	19:16	Reserved. Must be written zero, reads back zero	0	0	Reserved
CHTro	15	When set, when this trigger fires, generate a single cycle strobe on *TC_ChipTrigOut*.	R/W	0	Required
PDTro	14	When set, when this trigger fires, generate a single cycle strobe on *TC_ProbeTrigOut*.	R/W	0	Required
Impl	13:11	These bits are reserved for implementation specific trigger sources (internal or external to the TCB). Refer to the processor specification for the format and definition of this field.		0	Optional
0	10:7	Reserved. Must be written zero, reads back zero	0	0	Reserved

Table 4-12 *TCBTRIGx* Register Field Descriptions (Continued)

Fields		Description	Read/ Write	Reset State	Compliance
Names	Bits				
DM	6	When set, this Trigger will fire when a rising edge on the Debug mode indication from the core is detected. The write value of this bit always controls when this trigger will fire. If this trigger fires because this DM field is set, i.e., this is the cause of the trigger firing, then this can be determined by reading this DM field. If the DM field was written 1, then a read value of 1 indicates that this trigger has fired since the last write. Note that the action from a firing trigger could have been suppressed, and therefore, reading this field would be the only definite way to tell if the trigger fired and whether this was the cause. This special read value is valid until the *TCBTRIGx* register is written. Note that if the write value was 0 the read value is always 0.	R/W	0	Optional
CHTri	5	When set, this Trigger will fire when a rising edge on *TC_ChipTrigIn* is detected. The write value of this bit always controls when this trigger will fire. If this trigger fires because this CHTri field is set, i.e., this is the cause of the trigger firing, then this can be determined by reading this CHTri field. If the CHTri field was written 1, then a read value of 1 indicates that this trigger has fired since the last write. Note that the action from a firing trigger could have been suppressed, and therefore, reading this field would be the only definite way to tell if the trigger fired and whether this was the cause. This special read value is valid until the *TCBTRIGx* register is written. Note that if the write value was 0 the read value is always 0.	R/W	0	Required
PDTri	4	When set, this Trigger will fire when a rising edge on *TC_ProbeTrigIn* is detected. The write value of this bit always controls when this trigger will fire. If this trigger fires because this PDTri field is set, i.e., this is the cause of the trigger firing, then this can be determined by reading this PDTri field. If the PDTri field was written 1, then a read value of 1 indicates that this trigger has fired since the last write. Note that the action from a firing trigger could have been suppressed, and therefore, reading this field would be the only definite way to tell if the trigger fired and whether this was the cause. This special read value is valid until the *TCBTRIGx* register is written. Note that if the write value was 0 the read value is always 0.	R/W	0	Required

Table 4-12 *TCBTRIGx* Register Field Descriptions (Continued)

<table>
<tr><th colspan="2">Fields</th><th rowspan="2">Description</th><th rowspan="2">Read/ Write</th><th rowspan="2">Reset State</th><th rowspan="2">Compliance</th></tr>
<tr><th>Names</th><th>Bits</th></tr>
<tr><td>Type</td><td>3:2</td><td>Trigger Type: The Type indicates the action to take when this trigger fires. The table below show the Type values and the corresponding Trigger action.
<table><tr><th>Type</th><th>Trigger action</th></tr><tr><td>00</td><td>Trigger Start: Trigger start-point of trace.</td></tr><tr><td>01</td><td>Trigger End: Trigger end-point of trace.</td></tr><tr><td>10</td><td>Trigger Center: Trigger center-point of trace.</td></tr><tr><td>11</td><td>Trigger Info: No action trigger, only for trace info.</td></tr></table>The action is to set or clear the TCBCONTROLB_{EN} bit. A Start trigger will set TCBCONTROLB_{EN}, an End trigger will clear TCBCONTROLB_{EN}. The Center trigger will clear TCBCONTROLB_{EN} half way through the trace memory from the trigger point. The trace memory size is determined by the TCBCONFIG_{SZ} field for on-chip memory, or from the TCBCONTROLA_{SyP} field for off-chip trace memory.

If Trace is set, then a TF6 format is added to the trace words. For Start and Info triggers this is done before any other TF’s in that same cycle. For End and Center triggers, the TF6 format is added after any other TF’s in that same cycle.

If the TCBCONTROLB_{TM} field is implemented it must be set to Trace-To mode (00), for the Type field to control on-chip trace fill.

The write value of this bit always controls the behavior of this trigger.

When this trigger fires, the read value will change to indicate if the trigger action was ever suppressed. If so the read value will be 11. If the write value was 11 the read value is always 11. This special read value is valid until the TCBTRIGx register is written.

If the condition is not true, i.e., either the trigger did not fire, or it fired and the action was not suppressed, then it is valid for the read value to read anything but 11.</td><td>R/W</td><td>0</td><td>Required</td></tr>
<tr><td>FO</td><td>1</td><td>Fire Once. When set, this trigger will not re-fire until the TR bit is de-asserted. When de-asserted this trigger will fire each time one of the trigger sources indicates trigger.</td><td>R/W</td><td>0</td><td>Required</td></tr>
<tr><td>TR</td><td>0</td><td>Trigger happened. When set, this trigger fired since the TR bit was last written 0.

This bit is used to inspect if the trigger fired since this bit was last written zero.

When set, all the trigger source bits (bit 4 to 13) will change their read value to indicate if the particular bit was the source to fire this trigger. Only enabled trigger sources can set the read value, but more than one is possible.

Also, when set, the Type field and the Trace field will have read values which indicate if the trigger action was ever suppressed by a higher priority trigger.</td><td>R/W0</td><td>0</td><td>Required</td></tr>
</table>

4.10 Reset State

Reset state for all register fields is entered when one or both of the following two things happen:

1. *ETT_SoftReset* input is set high.
2. *ETT_TRST_N* input is set low.

Most fields can be reset synchronously on the next rising edge of *ETT_TCK*.

The fields $TCBCONTROLA_{On}$ and $TCBCONTROLB_{EN}$ should be reset asynchronously on any of the above two events. Internal registers in the core-clock domain that need to reset must treat *ETT_SoftReset* and *ETT_TRST_N* as asynchronous reset inputs. It is not guaranteed that the core-clock is running when either of the two resets are asserted. For synchronous register reset, the reset event must be remembered until the core-clock starts running.

Please see Chapter 7, “Trace Control Block TAP Interface,” on page 39 for a description of the *ETT_* pins.

Trigger Logic

The TCB is defined to optionally feature a trigger unit. Most of the actual implementation and functionality is implementation dependent, but if implemented the base-line structure must be as defined in this section.

5.1 Trigger Logic Overview

The trigger logic is functionally split in three parts.

- Trigger Source logic.
- Trigger Control logic
- Trigger Action logic.

Figure 5-1 shows the functional overview of the trigger flow in the TCB.

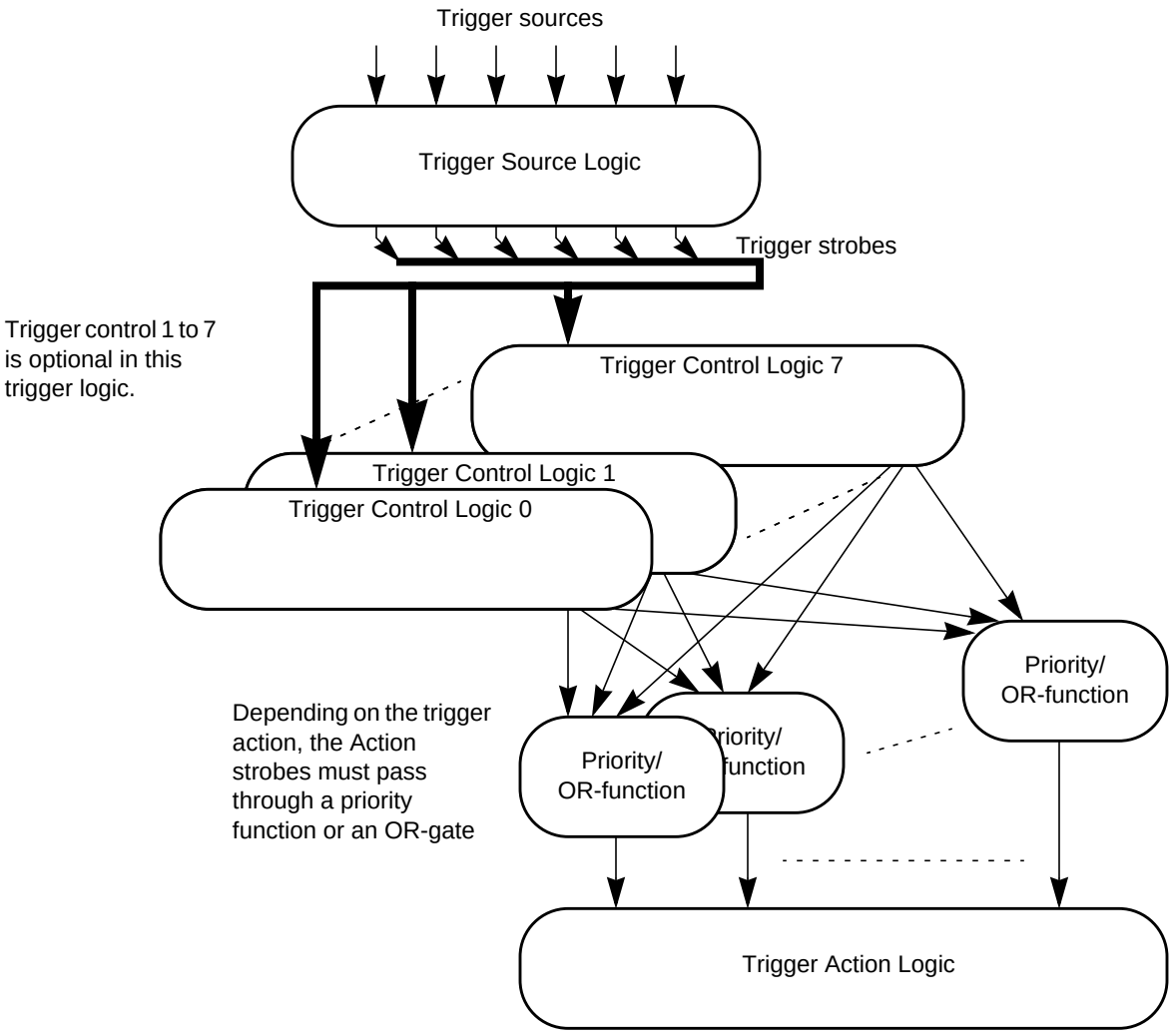


Figure 5-1 TCB Trigger processing overview

5.1.1 Trigger Source Logic

A number of source events can be defined that cause a trigger to fire when the corresponding source condition is satisfied.

In this version of the TCB, three sources have been defined. These are the two trigger inputs *TC_ChipTrigIn* and *TC_ProbeTrigIn* (see Section 5.3, "TCB Trigger Input/Output Signals"), and the Debug Mode (DM) indication from the processor core. The input triggers are all rising-edge triggers, and the Trigger Source logic must convert the edge into a single cycle strobe to the Trigger Control logic.

5.1.2 Trigger Control Logic

Eight possible Trigger Control registers (*TCBTRIGx, x={0..7}*) are defined. Each of these registers controls a trigger fire mechanism. They can have each of the Trigger Sources as the trigger event and they can fire one or more of the Trigger Actions. This is defined in the Trigger Control register *TCBTRIGx* (see Section 4.9, "TCBTRIGx Register (Reg 16-23)").

5.1.3 Trigger Action logic

A number of possible trigger actions in this version of the TCB are:

- Two output trigger strobes, *TC_ChipTrigOut* and *TC_ProbeTrigOut*. These are explained in Section 5.3, "TCB Trigger Input/Output Signals".
- The TF6 trace format as information output into trace memory. This is explained in Section 4.9, "TCBTRIGx Register (Reg 16-23)". Also read Section 5.2, "Simultaneous Triggers".
- The Start, End, and Center trigger actions. These are also explained in the sections pointed to above.

5.2 Simultaneous Triggers

Two or more triggers can fire simultaneously. The resulting behavior depends on trigger action set for each of them, and whether they should produce a TF6 trace information output or not. There are two groups of trigger actions: Prioritized and OR'ed.

5.2.1 Prioritized Trigger Actions

For prioritized simultaneous trigger actions, the trigger control register which has the lowest number takes precedence over the higher numbered *TCBTRIGx* registers. The oldest trigger takes precedence over everything.

The following trigger actions are prioritized when two or more *TCBTRIGx* registers fire simultaneously:

- Trigger Start, End, and Center type triggers ($TCBTRIGx_{Type}$ field set to 00, 01 or 10), which will assert/deassert the $TCBCONTROLB_{EN}$ bit. The Center trigger is delayed and will always change $TCBCONTROLB_{EN}$ because it is the oldest trigger when it deasserts $TCBCONTROLB_{EN}$. A Center trigger will not start the countdown if an even older Center trigger is using the frame counter.
- Triggers which produce TF6 trace information in the trace flow ($TCBTRIGx_{Trace}$ bit is set).

Regardless of priority, the $TCBTRIGx_{TR}$ bit is set when the trigger fires, even if the trigger action was suppressed. If the trigger is set to only fire once (the $TCBTRIGx_{FO}$ bit is set), then the suppressed trigger action will not be possible until after $TCBTRIGx_{TR}$ is written 0.

If a Trigger action is suppressed by a higher priority trigger, then the read value, when the $TCBTRIGx_{TR}$ bit is set, for the $TCBTRIGx_{Trace}$ field will be 0 for suppressed TF6 trace information actions. The read value in the $TCBTRIGx_{Type}$

field for suppressed Start/End/Center triggers will be 11. This indication of a suppressed action is sticky. If any of the two actions (Trace and Type) are ever suppressed for a multi-fire trigger (the $TCBTRIGx_{FO}$ bit is zero), then the read values in $TCBTRIGx_{Trace}$ and/or $TCBTRIGx_{Type}$, are set to indicate a suppressed action.

5.2.1.1 Center Trigger

The Center trigger's delayed deassertion of the $TCBCONTROLB_{EN}$ bit is always executed, regardless of priority from another Start trigger at the time of the $TCBCONTROLB_{EN}$ change. This means that if a Center trigger acts on the $TCBCONTROLB_{EN}$ bit to turn it off, n/2 frames after the trigger actually fires, and a Start trigger hit the same cycle and attempts to turn on the bit, then the Center trigger wins, regardless of the trigger number. This is because the oldest trigger takes precedence.

However, if a Center trigger has started the count down from n/2, but not yet reached zero, then a new Center trigger will NOT be executed. Only one Center trigger can have the cycle counter. This second Center trigger will store 11 in the $TCBTRIGx_{Type}$ field. But, if the $TCBTRIGx_{Trace}$ bit is set, a TF6 trace information will still go in the trace. This makes it possible to determine the read value of the $TCBTRIGx_{Type}$ field at the time of trigger source strobe.

5.2.2 OR'ed Trigger Actions

The simple trigger actions CHTro, PDTro and CTATrg from each *TCBTRIGx* register's action logic, are effectively OR'ed together to produce the final trigger. For example, one or more expected trigger strobes on *TC_ChipTrigOut* can disappear. External logic should therefore not rely on counting of strobes to predict a specific event unless simultaneous triggers are known not to occur.

5.3 TCB Trigger Input/Output Signals

Two sets of trigger input/outputs are defined on the TCB. One set is defined to be chip internal, and the other set is defined to be part of the probe interface. Table 5-1 shows the TCB signal names, and the related probe pin name for the probe trigger signals.

Table 5-1 TCB Trigger input and output

TCB pin name	Probe pin name	Description
TC_ChipTrigIn	N/A	Rising edge trigger input. The source should be on-chip. The input is considered async. I.e. double registered in the TCB.
TC_ChipTrigOut	N/A	Single cycle (relative to core clock) high strobe, trigger output. The target is supposed to be an on-chip unit.
TC_ProbeTrigIn	*TR_TRIGIN*	Rising edge trigger input. The source should be the Probe Trigger input. The input is considered async. I.e. double registered in the TCB.
TC_ProbeTrigOut	*TR_TRIGOUT*	Single cycle (relative to probe clock *TC_ProbeClk*) high strobe, trigger output. The target is supposed to be the Probes Trigger output.

PDtrace™ Interface

The TCB is on the receiving end of the PDtrace™ Interface, the processor core being the sender. The PDtrace™ Interface is described in detail in the reference document [2]. Several control and configuration signals exist on the PDtrace IF. This chapter in brief describes the TCB source of the *PDI_xx* signals to the processor core.

Most of the *PDI_xx* pins are controlled from the *TCBCONTROLA* register. When written, this register is checked and the values of corresponding *PDI_xx* input signals to the core are modified. Table 6-1 show a list of the *PDI_xx* signals, with the register/functional source in the TCB. The complete PDtrace™ signal list can be found in the reference document [2].

Table 6-1 PDtrace™ IF core input controls from TCB

Signal Name	TCB source register/function
PDI_TCBPresent	Either force to 1, or set to the evaluation of the following function: (TCBCONFIG$_{OnT}$ or TCBCONFIG$_{OfT}$)
PDI_TBImpl	Set to the evaluation of the following function: (TCBCONFIG$_{OnT}$ and TCBCONFIG$_{OfT}$)
PDI_StallSending	Set when additional trace data on the *PDO_xx* signal will cause the TCB fifo to overflow.
PDI_OffChipTB	Set to the evaluation of the following function: (TCBCONTROLB$_{OfC}$)
PDI_SyncOffEn	Either force to 1, or cycle 0->1->0 when either *TCBCONTROLB*$_{OfC}$ or *TCBCONTROLA*$_{SyP}$ is modified.
All other *PDI_xx* signal	Controlled directly from the bit settings in *TCBCONTROLA*. Please refer to Section 4.1, "TCBCONTROLA Register" on page 16 for details.

Trace Control Block TAP Interface

The Trace Control Block Registers are accessed through EJTAG TAP interface on the core. For this reason there must be a way for the core EJTAG TAP controller to access the Trace Control Block registers in a TAP-like manner. This section will describe the interface used for TAP access.

Since the core already implements an EJTAG TAP controller there is no need to duplicate the entire state-machine in the Trace Control Block. The interface described in this section assumes that the reader is familiar with the (E)JTAG TAP state machine and how it works.

7.1 Signal list

Table 7-1 Trace Control Block TAP Interface signals

Signal Name	Direction	Description	Compliance
ETT_TCK	In	EJTAG TAP controller clock. This signal is not an output from the core, but is the input to the TAP controller in the core. The TCB should use the same.	Required
ETT_TDI	In	This is the TDI signal from the EJTAG probe. As was the case with *ETT_TCK*, the TCB must use the same input as the TAP controller in the core.	Required
ETT_TRST_N	In	TAP reset. Async. reset input from the EJTAG probe. This is also not a signal from the core, but directly from the probe input.	Required
ETT_SoftReset	In	The TAP controller state-machine is in Test-Logic-Reset state.	Required
ETT_Capture	In	The TAP controller state-machine is in Data-Capture state. This indicates the *ETT_Inst[4:0]* input is valid.	Required
ETT_Shift	In	The TAP controller state-machine is in Data-Shift state.	Required
ETT_Update	In	The TAP controller state-machine is in Data-Update state. This indicates the *ETT_Inst[4:0]* input is valid.	Required
ETT_Inst[4:0]	In	Current value of the instruction register in the TAP controller. This selects which TCB register is the target in the Capture and Update cycles. ***ETT_Inst[4:0]*** / **TCB Register** 0x10 / *TCBCONTROLA* 0x11 / *TCBCONTROLB* 0x12 / *TCBDATA* All Other's / *Bypass* This table has the full list of registers which are supported by the TAP controller in the core. Any un-implemented register above must select the *Bypass* register. *Bypass* is a single bit register which always resets to '0' on Capture.	Required
ETT_TCBData	Out	Serial output data, synchronous to *ETT_TCK* rising edge. When the *ETT_Shift* is asserted and *ETT_Inst[4:0]* selects one of the three EJTAG TCB Registers, this output must present data	Required

7.2 Interface description

The TCB TAP interface is fully synchronous to the clock *ETT_TCK*. All inputs are captured at rising clock edge and the *ETT_TCBData* output must also change only as a result of a rising edge.

All the *ETT_xx* inputs from the core are fully registered outputs of the core. The TCB can anticipate very early timing on the assertion of these inputs. The *ETT_TCBData* output is NOT captured into a rising edge register in the core, but rather multiplexed directly to the *ETT_TDO* output register, which changes state on falling edge *ETT_TCK*. The TCB must guarantee good timing on this output, preferably directly from a rising edge *ETT_TCK* register.

Figure 7-1 shows a timing diagram for an access to the *TCBCONTROLA* register.

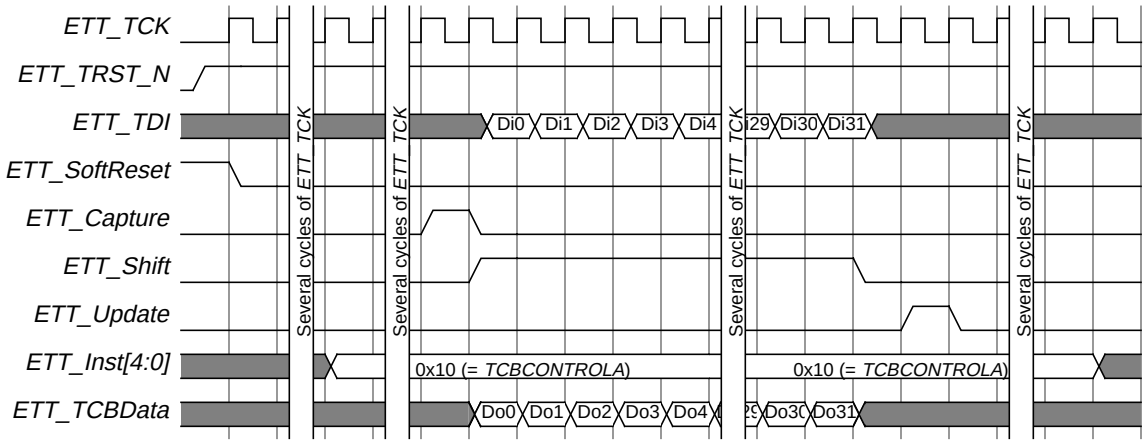


Figure 7-1 TCB TAP register access timing diagram

In the first two cycles *ETT_TRST_N* is released, and the selection of an instruction register is started in the TAP controller state-machine using *ETT_TMS* (not used by the TCB TAP). In the first multi-cycle block, the core TAP controller has its internal instruction register set to 0x10 (= *TCBCONTROLA* register). This is reflected on *ETT_Inst[4:0]*.

After the other multi-cycle block, the core TAP controller is in Capture Data Register state. This is reflected on *ETT_Capture*. When *ETT_Capture* is set, the next rising edge on *ETT_TCK* should update the TCB TAP shift register, with the value of the register selected by *ETT_Inst[4:0]* (in this case *TCBCONTROLA*). In the following 32 clock cycles the shift register should then receive write data on *ETT_TDI*, and present read data on *ETT_TCBData* (LSB first on both busses).

One or more cycles after *ETT_Shift* is de-asserted, the *ETT_Update* signal will be asserted for one cycle. Assertion of *ETT_Update* is the signal to write the current contents of the shift register to the register selected by *ETT_Inst[4:0]* (in this case *TCBCONTROLA*).

The EJTAG TAP controller will be moved to access other registers, which eventually changes the contents of the *ETT_Inst[4:0]* pins. Even though *ETT_Inst[4:0]* is asserted long before *ETT_Capture*, and de-asserted long after *ETT_Update*, the TCB TAP should only sample the value, when either *ETT_Capture* or *ETT_Update* is asserted.

Figure 7-2 shows a top-level view of the data-path of the TCB TAP.

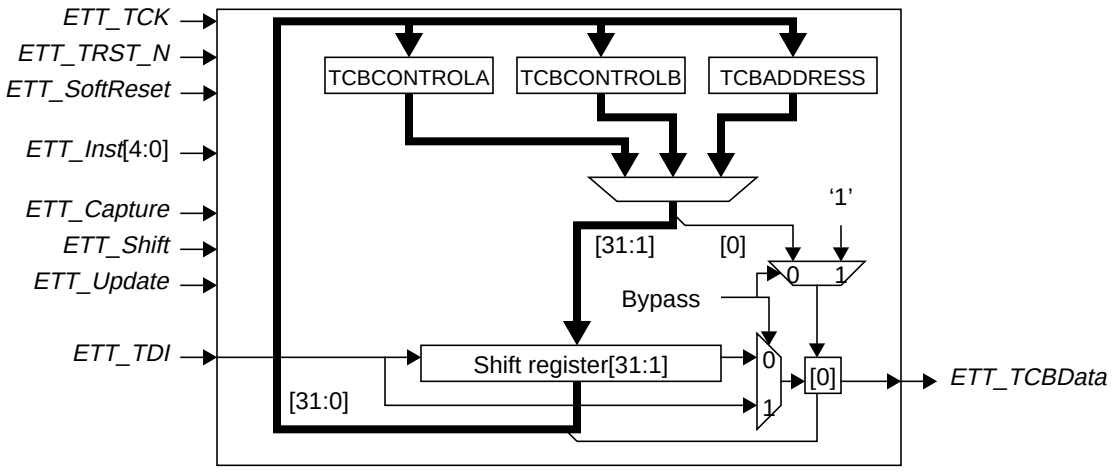


Figure 7-2 TCB TAP data-path

If one of the three TCB registers is unimplemented, access to an unimplemented register should work as a bypass register.

It is likely that the three TCB registers reside, or have counterparts which reside, in the core clock domain. In this case, proper synchronization must be observed between the TAP and core clock domains when the shift register contents are captured or when updates from the shift register are performed. In doing so, it is not guaranteed that the core clock is "much" faster than the *ETT_TCK* clock, or that the core clock is running at all for that matter.

A more detailed implementation specification is beyond the scope of this document.

TCtrace IF

When the TCB is implemented with the ability to send the trace information to a probe, this is done through an intermediate interface called the TCtrace IF. The TCtrace IF is used to connect a small Probe Interface Block (PIB) to the TCB. The PIB module is the module driving the actual Probe I/O pads which creates the Probe IF. The PIB is left as a separate unit, because the I/O timing will benefit from the PIB being placed physically close to the pads. Also the PIB can be more or less advanced with internal clock-multiplier to enable higher trace bandwidth on a limited number of *TR_DATA* trace pins.

The entire TCtrace IF is required in the TCB if Off-Chip Trace memory is implemented, otherwise it is optional. The Chip-level trigger input and outputs (*TC_ChipTrigIn* and *TC_ChipTrigOut*) are required if one or more trigger control registers are implemented.

8.1 Signal description

Unless otherwise noted, all the signals are synchronous with respect to the “core-clock”. The SIn inputs are static inputs which should not change except when the core is reset.

Table 8-1 TCB TCtrace IF signals

Signal Name	Direction	Description
TC_PibPresent	SIn	Must be asserted when a PIB is attached to the TC Interface. When de-asserted (low) all the other inputs are disregarded.
TC_TrEnable	Out	Trace Enable, when asserted the PIB must start the TR_Clk output running and can expect valid data on all other outputs.
TC_CRMax[2:0]	SIn	Maximum Clock ratio supported. This static input sets the $TCBCONFIG_{CRMax}$ field. It defines the capabilities of the PIB module.This sets the highest clock-ratio (lowest binary value) that the *TC_ClockRatio* can ever get.
TC_CRMin[2:0]	SIn	Minimum Clock ratio supported. This input sets the $TCBCONFIG_{CRMin}$ field. It defines the capabilities of the PIB module. This sets the lowes clock-ratio (highest binary value) that the *TC_ClockRatio* can ever get.
TC_ClockRatio[2:0]	Out	Clock ratio. This is the software set clock-ratio in $TCBCONTROLB_{CR}$. The simple PIB shown in Figure 9-3 on page 50, would support only a *TC_ClockRatio* value of “1:2”.
TC_ProbeWidth[1:0]	SIn	This static input will set the $TCBCONFIG_{PW}$ field. If this interface is not driving a PIB module, but some chip-level TCB-like module, then this field should be set to 11 (reserved value for PW).

Table 8-1 TCB TCtrace IF signals (Continued)

<table>
<tr><th>Signal Name</th><th>Direction</th><th>Description</th></tr>
<tr><td>TC_DataBits[2:0]</td><td>In</td><td>This input to the TCB identifies the number of bits picked up by the probe interface module in each “cycle”.

If TC_ClockRatio indicates a clock-ratio higher than 1:2, i.e., clock multiplication in the Probe logic is used. The “cycle” is equal to each core clock cycle.

If TC_ClockRatio indicates a clock-ratio lower than or equal to 1:2. Then “cycle” is (Clock-ratio * 2) of the core clock cycle. for 1:2; “cycle” is equal to core clock cycle. For 1:4; “cycle” is equal to one half of core clock cycle.

This input controls the down-shifting amount and frequency of the Trace-Word on TC_Data[63:0]. The bit widths and the corresponding TC_DataBits value are shown in the table below.
<table>
<tr><th>TC_DataBits[2:0]</th><th>Probe use following bits from TC_Data each cycle</th></tr>
<tr><td>000</td><td>TC_Data[3:0]</td></tr>
<tr><td>001</td><td>TC_Data[7:0]</td></tr>
<tr><td>010</td><td>TC_Data[15:0]</td></tr>
<tr><td>011</td><td>TC_Data[31:0]</td></tr>
<tr><td>100</td><td>TC_Data[63:0]</td></tr>
<tr><td>101</td><td rowspan="3">Unused</td></tr>
<tr><td>110</td></tr>
<tr><td>111</td></tr>
</table>
This input might change as the value on TC_ClockRatio[3:0] change.</td></tr>
<tr><td>TC_Valid</td><td>Out</td><td>This is asserted when a new TW is started on the TC_Data[63:0] signals. TC_Valid is only asserted when TC_DataBits is 100.</td></tr>
<tr><td>TC_Stall</td><td>In</td><td>When asserted, an new TC_Valid in the following cycle is stalled. TC_Valid is still asserted, but the TC_Data value and TC_Valid is kept static, until the cycle after TC_Stall is sampled low.

TC_Stall is only sampled in the cycle before a new TC_Valid cycle. And only when TC_DataBits is 100, indicating full word of TC_Data.</td></tr>
<tr><td>TC_Calibrate</td><td>Out</td><td>This signal is asserted when the TCBCONTROLB$_{Cal}$ bit is set.

For a simple PIB which only serves one TCB, this pin can be ignored. For a multi-core capable PIB which also uses TC_Valid and TC_Stall, the PIB must start producing the calibration pattern when this signal is asserted. See Section 4.2, "TCBCONTROLB Register" on page 19 under the Cal bit in Table 4-4, for a definition of the calibration pattern.</td></tr>
<tr><td>TC_Data[63:0]</td><td>Out</td><td>Trace word data. The value on this 64-bit interface is shifted down as indicated in TC_DataBits[2:0]. In the first cycle where a new TW is valid on all the bits and TC_DataBits[2:0] is 100, TC_Valid is also asserted.</td></tr>
<tr><td>TC_ProbeTrigIn</td><td>In</td><td>Rising edge trigger input. The source should be the Probe Trigger input. The input is considered async. I.e., double registered in the TCB.</td></tr>
<tr><td>TC_ProbeTrigOut</td><td>Out</td><td>Single cycle (relative to the “cycle” defined the description of TC_DataBits) high strobe, trigger output. The target is supposed to be the Probe’s Trigger output.</td></tr>
<tr><td>TC_ChipTrigIn</td><td>In</td><td>Rising edge trigger input. The source should be on-chip. The input is considered async. I.e., double registered in the TCB.</td></tr>
<tr><td>TC_ChipTrigOut</td><td>Out</td><td>Single cycle (relative to core clock) high strobe, trigger output. The target is supposed to be an on-chip unit.</td></tr>
</table>

8.2 *TC_Valid* and *TC_Stall* timing

Figure 8-1 shows the timing relationship between *TC_Valid* and *TC_Stall*, and when *TC_Data* will change value depending on *TC_Stall*.

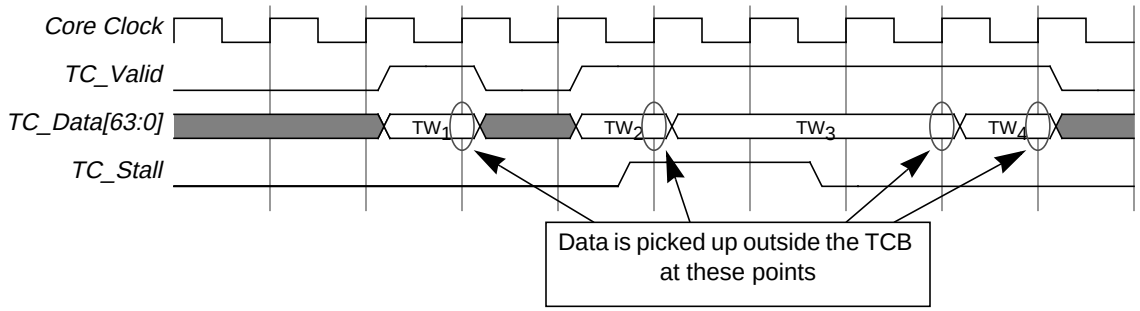


Figure 8-1 *TC_Valid* and *TC_Stall* timing

The stall/valid timing shown in Figure 8-1 is the same as is used in the PDtrace™ interface, between *PDI_StallSending* and *PDO_IamTracing*.

Probe IF

A trace system configured with a Probe is shown in Figure 1-2 on page 2.

Irrespective of the format that is being sent to external trace memory each cycle, the TCB sends out a number of bits via PIB on the Probe IF. The TCB must have some internal buffering that allows this type and rate of transmission.

9.1 Interface definition

The Probe IF can be implemented in a number of widths, allowing a trade-off between the number of pins used, and the available bandwidth for tracing. The ratio of the frequency on this interface to the CPU core clock frequency can also be configured, to give the maximum bandwidth possible. The signals are defined below.

Table 9-1 External Probe Interface signals

Signal Name	Direction	Description
TR_CLK	Out	Clock to the probe containing the external trace memory. This may be a double data-rate clock (DDR) and therefore both of its edges may be significant.
TR_DATA[15:0]	Out	Data signals to external trace memory. These may be limited to the following set of possible widths: 4, 8, 16.
TR_TRIGIN	In	Trigger Input. Rising edge trigger input.
TR_TRIGOUT	Out	Trigger Output. Single cycle trigger output.
TR_PROBE_N	In	Active Low Indicates that a probe is attached to the device. If this signal is inactive (high), the TR_ outputs can be disabled. It can also be used to control EJTAG signal routing if useful. This signal is optional on a PDtrace™-compatible device, but required on all probes.
TR_DM	Out	Debug mode: When asserted, indicates that the core has entered Debug Mode. In a multi-core chip, this output can be an AND or an OR or some other function of all the Debug-mode indications from each core. The actual function must be clearly specified in the multi-core chip documentation.

9.2 Probe Interface Block

The Probe Interface Block (PIB) is the module defined to be the on-chip link between the TCtrace IF and the Probe IF. A PIB can be more or less advanced, and capable in term of clock-multiplication/clock-division. This section will show three example implementations of a PIB.

9.2.1 Simple Probe Interface Block

Figure 9-1 shows the simplest implementation of a PIB. This PIB will support only a clock-ratio of 1:2. The number of *TR_DATA* pins can be 4, 8, or 16. In this example the maximum, i.e.,16 bits is shown. The 4 bit option may be impractical for some tracing options because does not provide sufficient bandwidth for the TCB to output trace data from the processor without back-stalling the processor pipeline. The PIB data pin width would have to be determined on a per application basis based on various trade-offs that are relevant for a specific user.

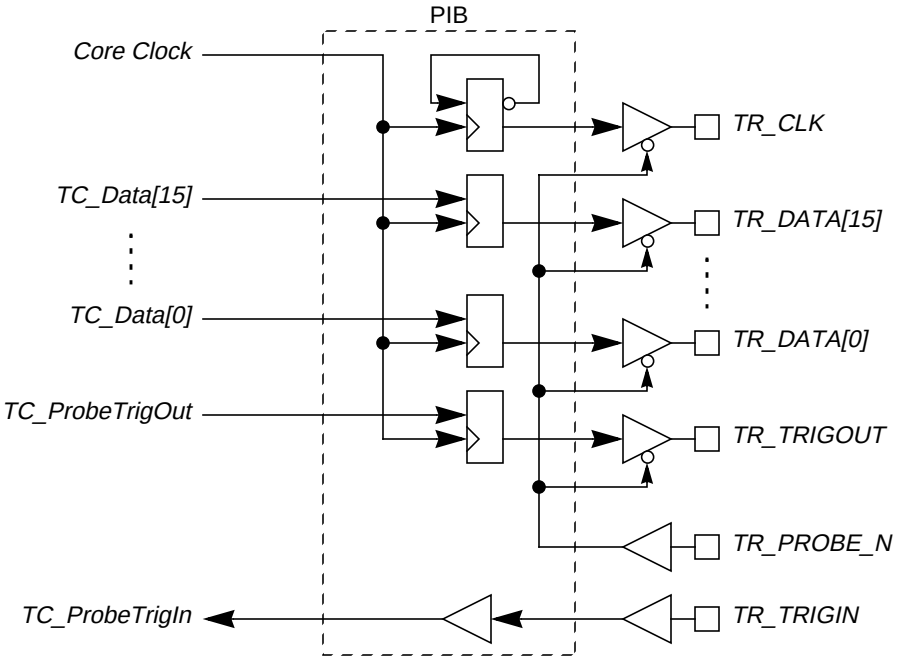


Figure 9-1 Simple Probe Interface Block

Table 9-2 shows the static control inputs to the TCB for the PIB in Figure 9-1.

Table 9-2 TCB Static Inputs for Simple PIB

Signal name	Value	Description
TC_CRMax[2:0]	100	Maximum clock-ratio is 1:2.
TC_CRMin[2:0]	100	Minimum clock-ratio is 1:2.
TC_ProbeWidth[1:0]	10	All the data pins *TR_DATA[15:0]* are used in this chip.
TC_DataBits[2:0]	010	The PIB will pick-up the 16 *TC_Data[15:0]* bits each cycle. The other *TC_Data* bits are not used by this PIB. This informs the TCB to do down-shifting by 16 bits each cycle.

9.2.2 Probe Interface Block with Clock-Multiplier

When the core-clock frequency is below the data-rate that can be handled by the probe used for debugging, the PIB can be implemented with a clock-multiplier between the Core-clock and the output registers. This can decrease the need for dedicated *TR_DATA* pins and/or increase the bandwidth of the Probe IF.

If the multiplication factor is made programmable, then *TC_CRMin* and *TC_CRMax* will have different values. The *TC_DataBits* value, will become a function of the *TC_ClockRatio* output from the TCB. Figure 9-2 shows a PIB with a clock-multiplier capable of doing 1x, 2x, and 4x clock multiplication. Note that only one output buffer for the 8 *TC_DATA[7:0]* pins is shown.

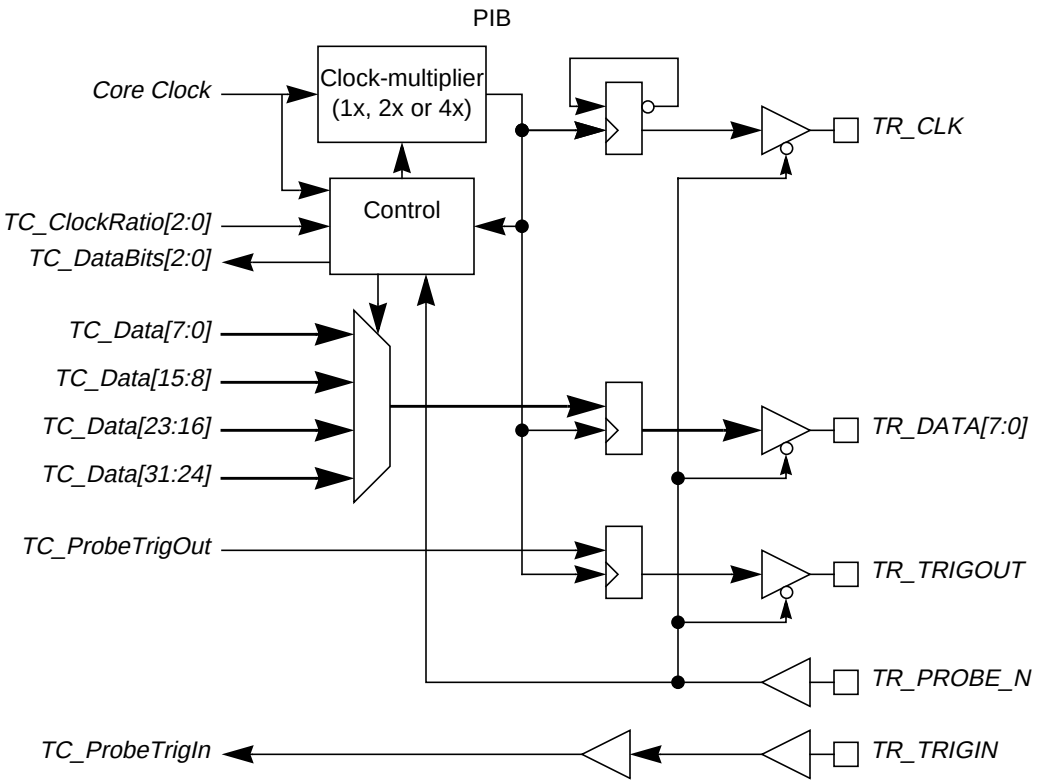


Figure 9-2 Probe Interface Block with Clock-multiplier

Table 9-3 shows the static control inputs to the TCB for the PIB in Figure 9-2.

Table 9-3 TCB Static Inputs for Clock-multiplier PIB

Signal name	Value	Description
TC_CRMax[2:0]	010	Maximum clock-ratio is 2:1 (4x on clock-multiplier is 2x on *TR_CLK*).
TC_CRMin[2:0]	100	Minimum clock-ratio is 1:2.
TC_ProbeWidth[1:0]	01	Only the data pins *TR_DATA[7:0]* are used in this chip.
TC_DataBits[2:0]	001	When *TC_ClockRatio* is 1:2 (100). TCB will shift down the *TC_Data[63:0]* with 8 bits for each Core-clock cycle.
	010	When *TC_ClockRatio* is 1:1 (011). TCB will shift down the *TC_Data[63:0]* with 16 bits for each Core-clock cycle.
	011	When *TC_ClockRatio* is 2:1 (010). TCB will shift down the *TC_Data[63:0]* with 32 bits for each Core-clock cycle.

9.2.3 Probe Interface Block with Clock-Divider

When a probe is not capable of capturing data at a data-rate equal to the Core-clock, then the PIB can divide down the TR_CLK. A PIB capable of Clock-division is shown in Figure 9-3. Note that this is very similar to the Simple PIB shown in Figure 9-1. The TCB is doing all the delay on the *TC_Data* pins, so there is no need to actually run the output registers slower than Core-clock. The Clock-divide is essentially just a counter which delays the inverted *TR_CLK* feedback, with one or more cycles. The Clock-divider can handle any of the defined divisions.

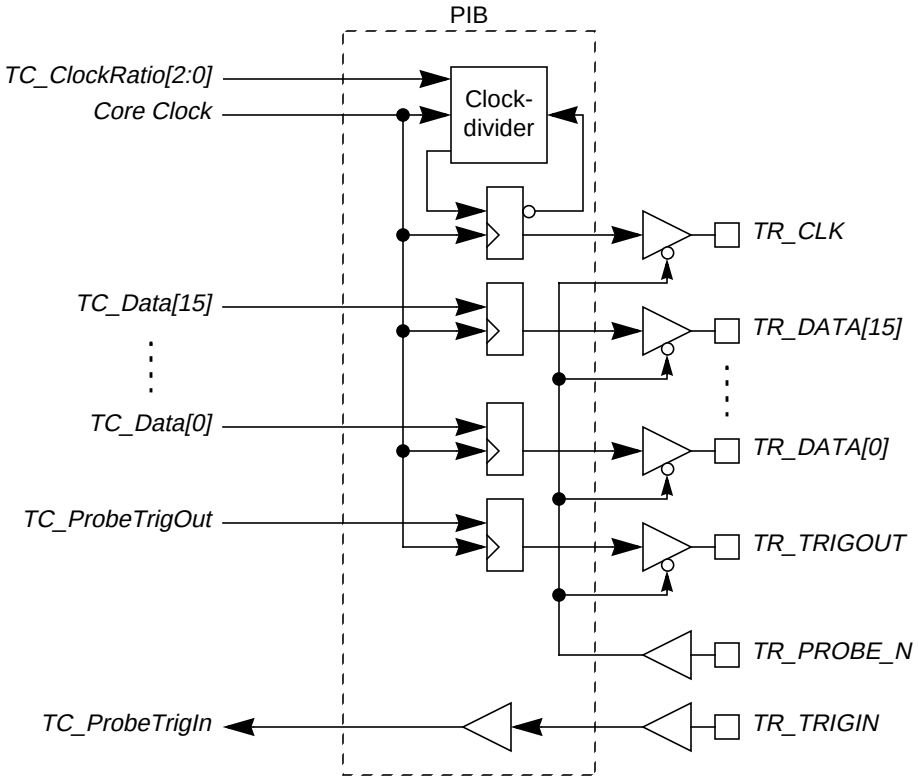


Figure 9-3 Probe Interface Block with Clock-divider

Table 9-4 shows the static control inputs to the TCB for the PIB shown in Figure 9-3.

Table 9-4 TCB Static Inputs for Simple PIB

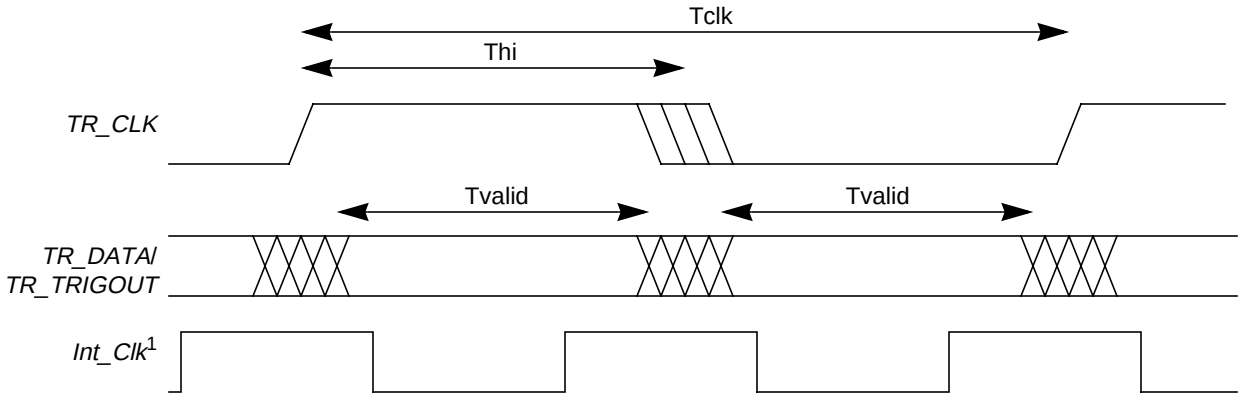
Signal name	Value	Description
TC_CRMax[2:0]	100	Maximum clock-ratio is 1:2.
TC_CRMin[2:0]	111	Minimum clock-ratio is 1:8.
TC_ProbeWidth[1:0]	10	All the data pins *TR_DATA[15:0]* are used in this chip.
TC_DataBits[2:0]	010	The PIB will pick-up the 16 *TC_Data[15:0]* bits each *TR_CLK* cycle. When the Clock-ratio is set lower then 1:2 in the TCB, the TCB will keep the *TC_Data* outputs constant for 2, 3 or 4 Core-clock cycles (for Clock-ratios of 1:4, 1:6 and 1:8).

9.3 DC Specifications

The signaling voltage on the interface is dependent on the technology used to implement the code, and is signaled to the probe using the VIO signal in the EJTAG interface. Therefore, the IO standard on the EJTAG and the PDtrace™ external memory interfaces shall use the same signaling voltages.

The output impedance of the drivers, and the impedance for signal traces on the board from the chip to the probe shall be 50R, +/- 10%.

9.4 AC Specifications



[1]: Int_Clk is the target (PIB) internal clock used to generate the TR_CLK and data outputs. It is shown only for reference.

Figure 9-4 Probe interface signal timing diagram

9.4.1 Required target AC timing specs

- Duty cycle: (Tclk x 0.45) < Thi < (Tclk x 0.55)
- Data valid period: Tvalid > (Tclk x 0.33)
- Tvalid is centered in each half-clock period of an ideal (50:50 mark/space ratio) *TR_CLK*.

At *TR_CLK* = 133MHz, skew and jitter requirements are the same as DDR SDRAM (+/- 0.62ns w.r.t. clock edges) but centered about the edges. (DDR spec gives +0.75/-0.50).

The output frequency is related by a fixed ratio to the core clock frequency as specified by the $TCBCONTROLB_{CR}$ field. The core frequency must be clearly indicated in the target documentation.

9.4.2 Required Probe AC timing specs

All probes must be capable of operating in a range of 25MHz < Tclk < 50MHz. The range of Tclk frequencies that a probe is capable of must be clearly indicated in the probe documentation.

The data valid period is relative to *Int_Clk*, not *TR_CLK*, and so is independent of *TR_CLK* duty cycle. At high speeds, it is assumed that probes will only use the rising edges of *TR_CLK* to lock a PLL. At lower speeds, it may be possible to use the delayed edges of *TR_CLK* to latch data in a simple probe, as the overall margins will be larger.

9.4.3 Probe - Target Calibration

To enable probes to perform dynamic phase adjustment, a test pattern can be generated by the target, consisting of a fixed bit pattern that is rotated through the available data bits. This is controlled by the *TCBCONTROLB*$_{Cal}$ bit.

9.5 Connector

The connector used is a 38-pin AMP Mictor connector, part number AMP 2-0767004-2, as used by a number of logic analyzers.

Table 9-5 Mictor Connector Pin Out

Pin no.	Signal	Pin no.	Signal
1	Reserved	2	Reserved
3	TR_PROBE_N	4	VIO
5	TR_CLK	6	TR_CLK
7	TR_DATA[15]	8	TCK
9	TR_DATA[14]	10	TMS
11	TR_DATA[13]	12	TDI
13	TR_DATA[12]	14	TDO
15	TR_DATA[11]	16	TRST*
17	TR_DATA[10]	18	RST*
19	TR_DATA[9]	20	DINT
21	TR_DATA[8]	22	TR_DM
23	TR_DATA[7]	24	Reserved
25	TR_DATA[6]	26	Reserved
27	TR_DATA[5]	28	Reserved
29	TR_DATA[4]	30	Reserved
31	TR_DATA[3]	32	Reserved
33	TR_DATA[2]	34	Reserved
35	TR_DATA[1]	36	TR_TRIGOUT
37	TR_DATA[0]	38	TR_TRIGIN

The number of used *TR_DATA* pins is configurable for a target system. The *TCBCONFIG*$_{PW}$ field indicates the used number of outputs on the implementation of the TCB/PIB. Note that this needs to be hard-wired to the TCB from the PIB using the *TC_ProbeWidth* signal.

Any probe must support all 16 data pins. There is no way to make the TCB/PIB use less data pins than what is already on the target chip.

All probes shall implement the PDtrace™ and EJTAG signals as specified. Target systems may optionally provide a 14-pin connector (as specified in the *EJTAG Specification* Ref [1]) in addition to the Mictor connector, in which case they should be effectively wired in parallel.

The additional *TR_DM* output, on pin 22, is an indication that the target processor-core has entered Debug Mode. This is an optional pin on any target, but all probes must have the input. In a multi-core trace environment, the *TR_DM* signal can be an implementation dependent function of the Debug-Mode indication from all the cores.

Reserved pins are reserved for future extentions.

For this revision of the spec.:

- Target systems must leave these pins un-connected.
- Probes should not test for any value on these pins.

9.6 Logic analyzer probing

The *TR_PROBE_N* pin is usually not accessible to logic analyzer probes, and tied high through a pull-up resistor. If the *TR_PROBE_N* pin is used in the chip, to tri-state all the *TR_* outputs, it is recommended to add a jumper to force *TR_PROBE_N* active (low), through a smaller pull-down resistor. This will enable a logic analyzer to monitor the probe interface, without the need to drive *TR_PROBE_N*.

On-Chip Trace Memory

When trace memory is available on-chip, as shown in Figure 1-3 on page 2, the memory is typically of smaller size than if it were external. The assumption is that it is of some value to trace a smaller piece of the program.

With on-chip trace memory, the TCB can work in three possible modes:

1. Trace-From mode.
2. Trace-To mode.
3. Under Trigger unit control.

Software can select this mode using the $TCBCONTROLB_{TM}$ field. If one or more trigger control registers (*TCBTRIGx*) are implemented, and they are using Start, End, or Center triggers, then the trace mode in $TCBCONTROLB_{TM}$ should be set to Trace-To mode.

10.1 On-Chip Trace Memory size

The supported On-chip trace memory size is 256 byte to 8 Mbytes, defined as powers of 2. The actual size is set in the $TCBCONFIG_{SZ}$ field.

10.2 Trace-From Mode

In the Trace-From mode, tracing begins when the processor enters in to a processor mode/ASID value which is defined to be traced or when an EJTAG hardware breakpoint trace trigger turns on tracing. Trace collection is stopped when the buffer is full. The TCB then signals buffer full using $TCBCONTROLB_{BF}$. When external software, polling this register, finds the $TCBCONTROLB_{BF}$ bit set, it can then read out the internal trace memory. Saving the trace into the internal buffer will re-commence again only when the $TCBCONTROLB_{BF}$ bit is reset and if the core is sending valid trace data (i.e., *PDO_IamTracing* not equal 0).

10.3 Trace-To Mode

In the Trace-To mode, the TCB keeps writing into the internal trace memory, wrapping over and overwriting the oldest information, until the processor is reaches an end of trace condition. End of trace is reached by leaving the processor mode/ASID value which is trace, or when an EJTAG hardware breakpoint trace trigger turns tracing off. At this point, the on-chip trace buffer is then dumped out in a manner similar to that described above in Section 10.2, "Trace-From Mode".

References

[1] EJTAG Specification
Revision 2.60
MD00047
MIPS Technologies, Inc.

[2] PDtrace™ Interface Specification
Revision 2.06
MD00136
MIPS Technologies, Inc.

Revision History

Table B-1 Revision History

Revision	Date	Who	Description
01.00		FT	Initial release
01.01	August 28, 2001	FT	Changes this revision. • Added a new Chapter 6, “PDtrace™ Interface,”. (pushed the following chapters down one number). • Clarified the purpose of *TCBCONTROLA* in Section 4.1, "TCBCONTROLA Register" on page 16. Special attention to the On field. • Updated the description of the $TCBCONTROLB_{WR}$ bit in Section 4.2, "TCBCONTROLB Register" on page 19 • Clarified/Updated the description of the $TCBCONTROLB_{EN}$ bit in Section 4.2, "TCBCONTROLB Register" on page 19. • Added sub-section heading in Chapter 2, “Trace Message Format,”. • Added PDO_StallSending as a common signal for multi-pipeline tracing in Section 2.2, "Multi-Pipe Tracing Formats" on page 6. • Re-arranged Table 3-2 on page 10, to go left-right and not top down. • Modified reset value of $TCBCONTROLB_{CR}$ to 100. • Added compliance column to Table 4-2 on page 15. • Modified all register field references to be $REGISTER_{FIELD}$. • Renamed the AB field in TCBCONTROLA to TB.
01.02	September 18, 2001	FT	• Declassified document.
01.03	November 28, 2001	FT	• Renamed *TR_PROBE* to *TR_PROBE_N*. Changed explanation to define it as an active low pin.
01.04	March 21, 2002	RT/FT	• Added complete description of Trace Formats and Trace Words. Old chapter 2 replaced by chapters 2 and 3. All following chapters bumped one number. • Modified NC (No Connect) pins on Trace Connector to Reserved in Section 9.5, "Connector" on page 52. • Small clarification on TC_CRMax and TC_CRMin input values in Table 8-1 on page 43. • Removed the TCBTRIGx field CTATrg and the register TCBPDCH since they were incompletely defined. • In the TCBTRIGx register, bits 16..19 and 7..10 have been redefined as Reserved and now only bits 20..22 and 11..13 are implementation dependent. • Change name “Trigger About” to “Trigger Center”.