AMBA™ AHB Trace Macrocell (HTM)

Revision: r0p4

Technical Reference Manual

ARM DDI 0328E

AMBA AHB Trace Macrocell (HTM)
Technical Reference Manual

Release Information

Change history

Date	Issue	Confidentiality	Change
30 September 2004	A	Non-Confidential	First release
22 March 2005	B	Non-Confidential	Updated for r0p1. Programmer's model revised.
17 November 2006	C	Non-Confidential	Updated for r0p2. Idle status added.
11 June 2007	D	Non-Confidential	Updated for r0p3.
18 April 2008	E	Non-Confidential	Updated for r0p4.

Product Status

The information in this document is final, that is for a developed product.

Web Address

`http://www.arm.com`

Contents
AMBA AHB Trace Macrocell (HTM) Technical Reference Manual

List of Tables
AMBA AHB Trace Macrocell (HTM) Technical Reference Manual

List of Figures
AMBA AHB Trace Macrocell (HTM) Technical Reference Manual

Preface

This preface introduces the *AMBA AHB Trace Macrocell (HTM) Technical Reference Manual* (TRM). It contains the following sections:

About this book

This is the TRM for the *AMBA AHB Trace Macrocell (HTM)*. This book is intended to be read in conjunction with the *AMBA AHB Specification* (Rev 2.0), the *CoreSight Architecture Specification*, and the *CoreSight System Design Guide*.

HTM-specific implementation matters and general protocol design features are described in this book.

Product revision status

The r*n*p*n* identifier indicates the revision status of the product described in this book, where:

r*n* Identifies the major revision of the product.

p*n* Identifies the minor revision or modification status of the product.

Intended audience

This book has been written for:

- Hardware and software engineers integrating CoreSight HTM into an ASIC or SoC
- Designers of development tools providing support for HTM functionality.

Readers must be familiar with the use of AMBA and strategies for using and controlling buses. Bus terminology as used in this manual is defined in the glossary.

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

Read this chapter for an introduction to the HTM.

Chapter 2 *Functional Description*

Read this chapter for a detailed description of the HTM.

Chapter 3 *Programmer's Model*

Read this chapter for a description of the programming registers of the HTM.

Chapter 4 ***Protocol Details***

Read this chapter for detailed information about the trace output and its formats.

Chapter 5 ***Implementation-specific Characteristics***

Read this chapter for a description of the implementation-defined features of the HTM. Features of the HTM64 and HTM32 versions of the HTM are described in this chapter.

Chapter 6 ***Programmer's Model for Test***

Read this chapter for a description of the test support in the HTM.

Appendix A ***Signal Descriptions***

Read this appendix for a description of the HTM signals.

Appendix B ***Troubleshooting***

Read this appendix for help with troubleshooting the HTM.

Appendix C ***Revisions***

Read this for a description of the technical changes between released issues of this book.

Glossary Read this for definitions of terms used in this book.

Conventions

The following conventions are used in this book:

- *Typographical*
- *Timing diagrams* on page xviii
- *Signals* on page xix.

Typographical

The typographical conventions are:

italic Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.

bold Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.

`monospace` Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.

`monospace` — Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

`monospace italic` — Denotes arguments to monospace text where the argument is to be replaced by a specific value.

`monospace bold` — Denotes language keywords when used outside example code.

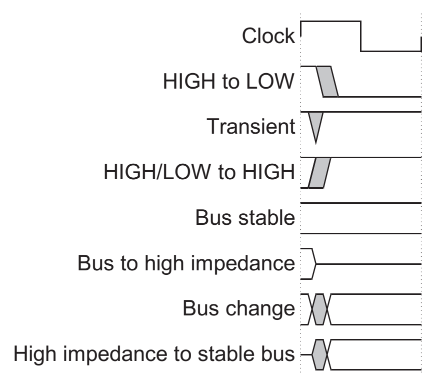
< and > — Angle brackets enclose replaceable terms for assembler syntax where they appear in code or code fragments. They appear in normal font in running text. For example:

- `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>`
- The Opcode_2 value selects which register is accessed.

Timing diagrams

The figure named *Key to timing diagram conventions* explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.



Key to timing diagram conventions

Signals

The signal conventions are:

Signal level The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means:
- HIGH for active-HIGH signals
- LOW for active-LOW signals.

Prefix H Denotes *Advanced High-performance Bus* (AHB) signals.

Prefix n Denotes active-LOW signals except in the case of AHB or *Advanced Peripheral Bus* (APB) reset signals.

Prefix P Denotes APB signals.

Suffix n Denotes AXI, AHB, and APB reset signals.

Further reading

This section lists publications by ARM and by third parties.

See `http://infocenter.arm.com/` for access to ARM documentation.

ARM publications

This book contains information that is specific to the product. See the following documents for other relevant information:
- *ARM Architecture Reference Manual*, ARM IHI 100
- *AMBA AHB Specification (Rev 2.0)*, ARM IHI 0011
- *CoreSight Architecture Specification*, ARM IHI 0029
- *CoreSight System Design Guide*, ARM DGI 0012
- *CoreSight Components Technical Reference Manual*, ARM DDI 0314
- *CoreSight Components Implementation Guide*, ARM DII 0143
- *Systems IP ARM11 AMBA (Rev 2.0) AHB Extensions*, ARM IHI 0023.

Feedback

ARM welcomes feedback on this product and its documentation.

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier giving:

- the product name
- a concise explanation of your comments.

Feedback on this book

If you have any comments on this book, send an e-mail to `errata@arm.com`. Give:

- the title
- the number
- the relevant page number(s) to which your comments apply
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Chapter 1
Introduction

The CoreSight *AHB Trace Macrocell* (HTM) is a real-time trace module capable of address and data tracing of the AHB bus. The HTM is an integral part of the ARM CoreSight Debug and Trace solution. This chapter contains the following sections:

1.1 About the CoreSight AHB Trace Macrocell (HTM)

The HTM provides address and data trace information about AHB buses. The information from an HTM can be used with the debugger to enable easy, accurate debugging on AHB-based embedded systems. The HTM provides extensive resources for event recognition to generate trigger events. The HTM generates trace data for output through the *AMBA Trace Bus* (ATB). The trace debug function is non-intrusive and HTM can be controlled using an APB (AMBA v3) interface.

The trace operation indicates the data transfers that have taken place to defined memory locations or regions. Other AMBA control information can also be included. However, other operations like IDLE cycles and BUSY cycles in AHB are not traced.

The HTM is an *Advanced Microcontroller Bus Architecture* (AMBA) compliant *System-on-Chip* (SoC) debug component.

The HTM is available in three versions:

- HTM64 for 64-bit AHB systems
- HTM32 for 32-bit AHB systems
- HTM32L for 32-bit AHB systems with a 64-byte FIFO.

See Chapter 5 *Implementation-specific Characteristics* for information about the differences between these versions.

1.2 HTM used in a CoreSight system

The *Debug Access Port* (DAP) drives the Debug APB and you can use the Debug APB to program the HTM. HTM captures the AHB activities of the AHB you want to monitor and the trace data is output using the *AMBA Trace Bus* (ATB) interface. The trace data is merged with trace data from other trace sources, such as the *Embedded Trace Macrocell* (ETM), to produce a single trace data stream. The trace data can be sent directly to the trace port through the *Trace Port Interface Unit* (TPIU), or connected to an ATB replicator so that the data can be stored in the *Embedded Trace Buffer* (ETB).

The HTM can also interface with the *Embedded Cross Trigger* (ECT) so that events from other CoreSight devices, such as the ETM, can be used to drive the HTM or events from the HTM can be used to drive other CoreSight devices.

Figure 1-1 on page 1-4 shows an example CoreSight system that includes an HTM.

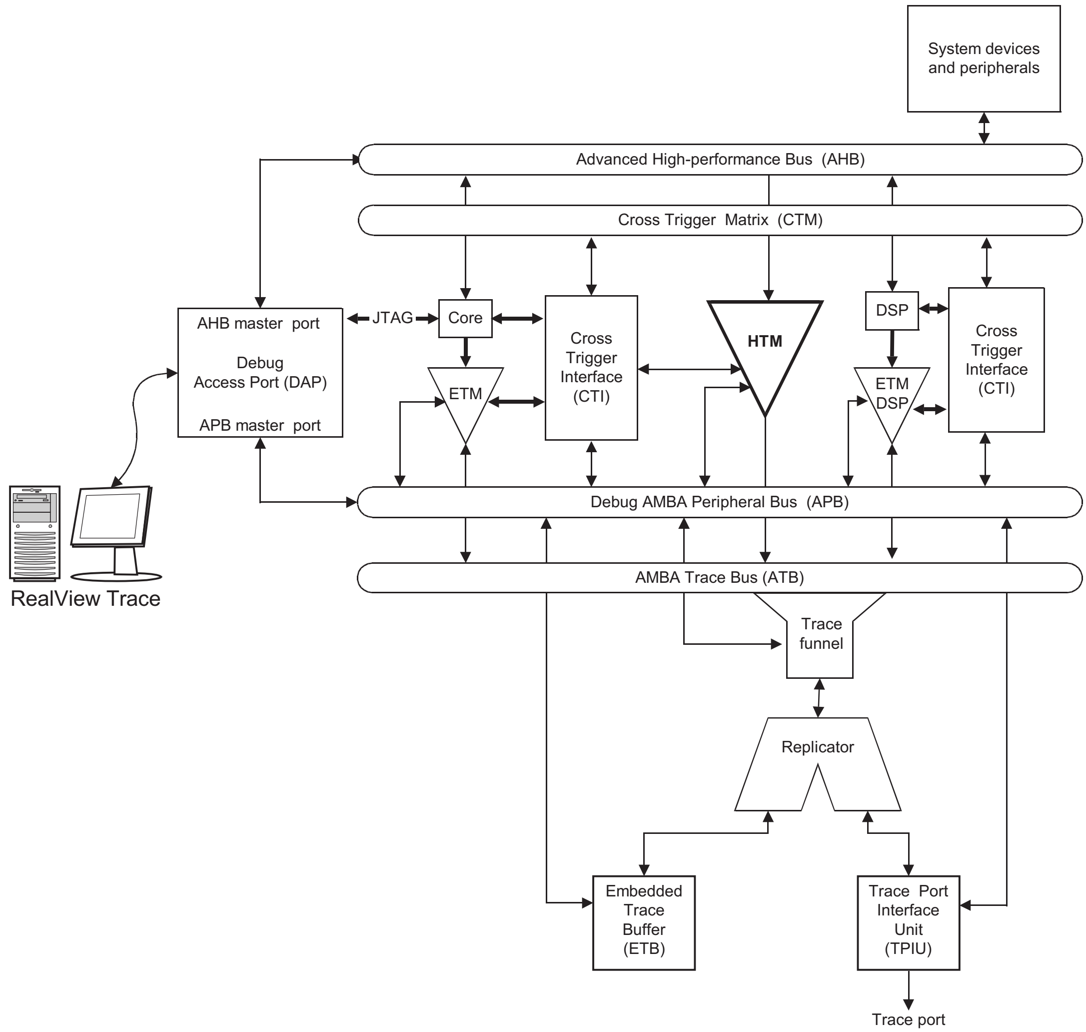


Figure 1-1 HTM in an example CoreSight system

1.3 Structure of the HTM

Figure 1-2 shows a block diagram of the HTM.

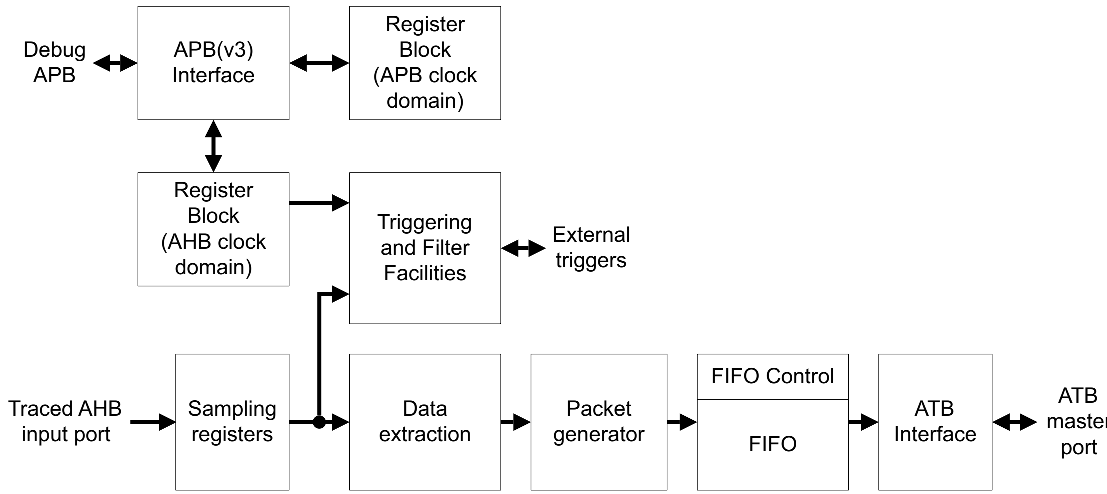


Figure 1-2 HTM block diagram

The HTM comprises the following main components:

APB interface

Enables the HTM register block to be accessed by the Debug APB (AMBA v3).

Register blocks

Control and data registers of the HTM. There are two register blocks in the design. The APB register block contains global enable signals, status and ID registers. The AHB register block contains the remainder of the registers.

Traced AHB input

Input port connected to the AHB bus being monitored.

Triggering and filtering resources

Enable you to control tracing by specifying the exact set of triggering and filtering resources required for a particular application. Resources include address comparators and data comparators, counters, and sequencers.

Sampling registers

Logic to sample the activities on the AHB bus on the rising clock edge.

Data extraction

Logic to extract valid data from the data bus. For example, a byte transfer on a 64-bit bus might have only one byte of valid data. However, its position on the data bus depends on address and endian settings. The data extraction logic extracts the valid data from the 64-bit data bus, or 32-bit data bus for HTM32.

Packet generator

Converts AHB bus activities into trace data packets.

FIFO and FIFO control

The FIFO accepts up to four packets per clock cycle. The FIFO controller monitors the data input and output operations. The FIFO controller also monitors the space available in the FIFO.

AMBA Trace Bus (ATB) master port

Outputs signals on the ATB that provide information about the operation of the AHB. The trace protocol provides a real-time trace capability for the AHBs that are deeply embedded in a much larger ASIC design. When the trace is captured using the CoreSight infrastructure, the debugger extracts the information and decompresses it to provide trace of the AHB bus activity.

The ATB is a common bus used by trace devices to send trace data from a source to a link. Trace sources are equivalent to AMBA bus masters, and trace sinks such as the TPIU or ETB are equivalent to an AMBA bus slave.

ATB supports:

- stalling of trace sources
- identification of the source that generated the trace
- capture of any number of the bytes of the data bus on a given cycle
- signaling to indicate that holding FIFOs must be flushed.

1.3.1 Clocks and resets in HTM

There are three clock domains in the HTM design:

- Debug APB
- AHB
- ATB.

Each clock domain has its own asynchronous reset input.

Table 1-1 describes the HTM resets.

Table 1-1 HTM reset signal descriptions

Name	Source	Description
PRESETDBGn	System controller	Asynchronous reset for Debug APB bus.
HTMHRESETn	System controller	Asynchronous reset for the HTM circuits that are in the AHB clock domain. Asserting this signal resets resource registers and erases data within the HTM FIFO. It is active when Debug APB is reset, but the signal timing is synchronized to **HCLK**.
HRESETn	System controller	AHB system reset, used as a status input. It does not reset any registers in the HTM. If HTM trace is active when **HRESETn** is asserted, the HTM outputs a special packet in the ATB packet stream to indicate an AHB reset.
ATRESETn	System controller	Asynchronous reset for ATB.

The AHB clock, **HCLK**, can be asynchronous to other clock signals. A synchronization bypass control signal, **SYNCBYPASS**, enables reduction of register access latency if the **HCLK** and **PCLKDBG** clock signals are synchronous. There is no bypass control signal for HCLK-to-ATCLK because it is assumed to always be asynchronous and latency on ATB is not an important issue.

The Debug APB and ATB also have clock enable signals called **PCLKENDBG** and **ATCLKEN** respectively. These enable the bus interface to be driven by a higher frequency clock, which is an integer multiple of the equivalent clock frequency. If the clock enable is HIGH, then the next rising clock edge is valid. If the clock divider function is not required, then these clock enable signals must be tied HIGH.

1.3.2 Data buses

32-bit data buses and 64-bit data buses can be traced. See *Endianness support and bus width support* on page 2-24 and *Bus width support* on page 2-24.

See *HTM64 and HTM32 features summary* on page 5-2 for details of data bus widths that can be traced using HTM64 and HTM32.

1.4 HTM features

Other features of the HTM are described in the following sections:

- *Register access lock*
- *Security pins*
- *Security and software protection* on page 1-10
- *ASIC control output* on page 1-10
- *Bus select output* on page 1-10
- *ARM11 AHB extensions* on page 1-12
- *Unsupported signals* on page 1-12
- *Power saving and clamping logic* on page 1-12.

1.4.1 Register access lock

A lock mechanism prevents accidental overwrite of control registers in HTM. To write to a register, the register bank in the HTM must be unlocked by writing a unique access code, hard-coded in the design as `0xC5ACCE55`, to the Lock Access Register. To activate the lock again, write any value other than the access code to the Lock Access Register. The current lock status can be checked in the HTM Status Register.

The lock can be bypassed by accessing the HTM using the upper 2GB of memory address. The upper 2GB of the debug APB can only be accessed by the DAP, so there is no risk of the register accidentally being erased by another bus master.

See *HTM Status Register, HTMSTATUS* on page 3-8, *HTM Lock Access Register, HTMLOCK_ACCESS* on page 3-44, and *HTM Lock Status Register, HTMLOCK_STATUS* on page 3-45 for details of these registers.

1.4.2 Security pins

TrustZone is supported by monitoring the following security pins:

NIDEN Non-Invasive Debug Enable.

SPNIDEN Secure Privileged Non-Invasive Debug Enable.

DBGEN Debug Enable.

SPIDEN Secure Privileged Invasive Debug Enable.

Trace filtering logic determines if the transaction is traced based on the values of these signals and the **HPROT** information. Currently the usage of **HPROT[6]** is not defined in the AMBA specification. It is used in some ARM systems to indicate secure accesses, that is **HPROT[6]** = 0 when the transfer is secure. Otherwise the transfer is non-secure. If **HPROT[6]** is not available in the bus system, this pin must be tied HIGH so that all transfer is traced.

The TrustZone control signals operate as follows:

- If both **NIDEN** and **DBGEN** are LOW, no AHB transfers are traced and all debug functionality is disabled. This includes disabling components such as the counters and comparators.
- If both **SPNIDEN** and **SPIDEN** are LOW, all secure transfers on the AHB, indicated by a LOW **HPROT[6]**, are not traced.

  The four TrustZone configuration pins are synchronized to the **HCLK** domain before they are used by internal logic, so from the time these two pin change their state, the actual behavior change takes place two **HCLK** cycles later.
- When **DBGEN** is HIGH, it has the same effect as setting **NIDEN** HIGH, that is, non-secure trace is enabled because **DBGEN**, invasive debug enable, implies non-invasive debug enable.
- When **SPIDEN** is HIGH, it has the same effect as setting **SPNIDEN** HIGH because **SPIDEN** also implies **SPNIDEN**.

Table 1-2 lists the security pin values and HTM tracing behavior.

Table 1-2 Security pin values and HTM tracing behavior

NIDEN	DBGEN	SPNIDEN	SPIDEN	Trace of non-secure transfers, HPROT[6] = 1	Trace of secure transfers, HPROT[6] = 0
0	0	X	X	Not allowed	Not allowed
1	X	0	0	Allowed	Not allowed
X	1	0	0	Allowed	Not allowed
1	0	1	X	Allowed	Allowed
1	0	X	1	Allowed	Allowed
0	1	1	X	Allowed	Allowed
0	1	X	1	Allowed	Allowed

See *HTM64 and HTM32 features summary* on page 5-2 for details of supported security features in HTM64 and HTM32 versions of HTM.

See Systems IP ARM11AMBA (Rev 2.0) AHB Extensions, ARM IHI 0023 for more information about **HPROT**.See Appendix A *Signal Descriptions* for a description of all HTM signals.

1.4.3 Security and software protection

To ensure that user software does not modify the behavior of privileged software, system designers must ensure that all unprivileged accesses are blocked from the Debug APB. Software running in user mode must not have access to any of the debug and trace peripherals.

1.4.4 ASIC control output

The HTM provides an ASIC control output port. This port is a general purpose output port. Use this port to control user-defined test logic such as a dummy bus master that creates additional AHB traffic to emulate different AHB traffic loading situations. It can also be used to stop a specified bus master from generating transfers on the AHB system.See *HTM ASIC Control Register, HTMASICCTRL* on page 3-26 for details of the register and an example configuration.

1.4.5 Bus select output

The HTM provides a 3-bit bus select output port. It selects different buses for tracing by the HTM as shown in Figure 1-3 on page 1-11.

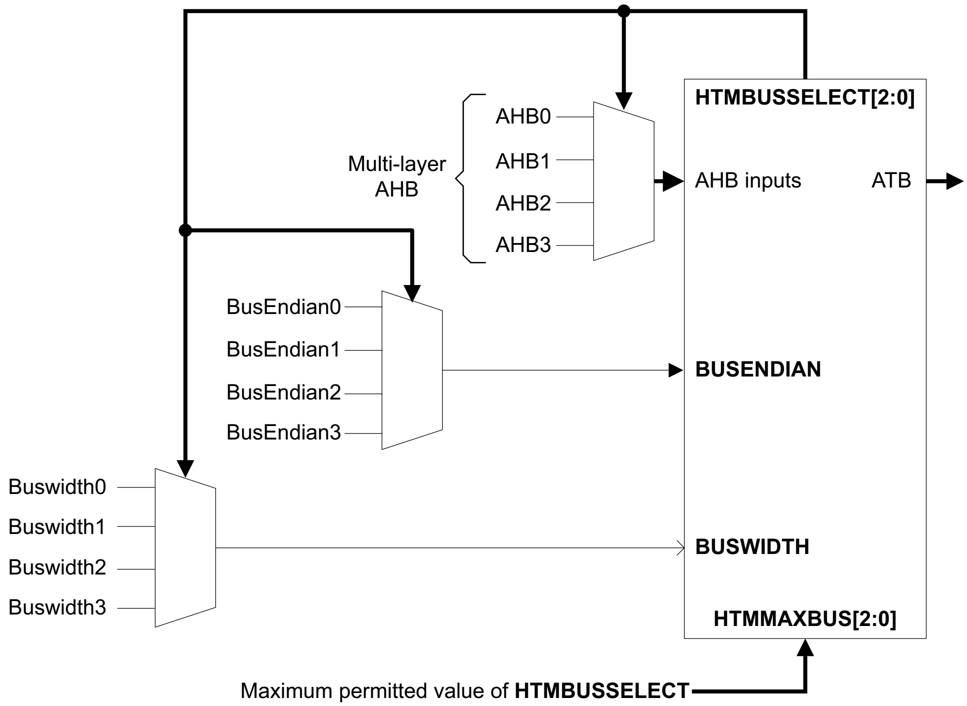


Figure 1-3 Example use of HTMBUSSELECT

To ensure that the bus multiplexer selects only existing buses, another 3-bit input port, **HTMMAXBUS**, is provided and must be used to indicate the maximum permitted value of **HTMBUSSELECT**. Debugger software must read the status of **HTMMAXBUS** from the HTMCFGCODE2 Register to determine how many AHB buses are connected to the HTM. A maximum of eight AHB buses can be monitored by bus multiplexing but the buses cannot be monitored simultaneously.

To switch trace from one AHB bus to another, the debug software must set the PROG bit to disable the trace, program the HTMBUSSELECT Register, and then clear the PROG bit. The bus multiplexer must not be switched during trace because this can create invalid bus transfer activities.

See *HTM Bus Select Register, HTMBUSSELECT* on page 3-27 for more information on the PROG bit.

See *HTM Configuration Code 2 Register, HTMCFGCODE2* on page 3-12 for details of **HTMBUSELECT**.

1.4.6 ARM11 AHB extensions

The ARM11 AHB extensions support unaligned accesses and exclusive accesses. They also support the use of multiple AHB bus masters.

See *HTM AMBA Control Select Registers, HTMHCTRLSEL0-7* on page 3-34.

See *HTM64 and HTM32 features summary* on page 5-2 and *HTM restrictions* on page 5-8 for details of HTM64 and HTM32 versions of the HTM.

1.4.7 Unsupported signals

The following signals are not supported:

AHB signals

HSPLIT[15:0] (Split complete bus for split-capable AHB systems).

ARM11 sideband signals

HSIDEBAND[3:0] (ARM1136 inner memory access attribute).

WRITEBACK (ARM1136 cache memory access attribute).

1.4.8 Power saving and clamping logic

The clamping logic is designed for systems with separate power domains for AHB and debug logic.

The source code of the HTM provides optional support for clamping logic to isolate circuit in different power domain. The clamping logic is controlled by two power down indication signals, **nCSOCPWRDN** and **nCDBGPWRDN**. By default, these two signals are only connected to the HTM status register and the register access interface, and the optional clamping logic is not enabled in the source code. You must edit the source code to enable the clamping logic instantiation and replace the clamping gate with a suitable gate in the technology library.

For information on power down and clamping support see *Power-down indication signals and clamping logic* on page A-9.

1.5 HTM functional description

The following sections describe functional operation of the HTM:

- *HTM operations*
- *FIFO operation*
- *ASIC control output* on page 1-10
- *External trace disabling* on page 1-15.

1.5.1 HTM operations

A number of AHB transfers are selected based on trace filter settings. Trace data is output continuously but, because bandwidth is limited on the ATB and the TPIU, only a small portion of transfers can be traced. If the FIFO is almost full, data suppression is activated. If the FIFO is completely full, trace is lost until the FIFO is free again. In both cases, the HTM outputs a warning message in the form of a packet, indicating the errors.

The HTM can be disabled by the PROG bit the in the HTM Control Register. When this bit is set, the resources (counters, comparators, and sequencer for example) and the trace operation are stopped so that resources are not activated accidentally, or trace is not accidentally generated during the programming of the HTM.

A trigger event generator is included in the HTM design. When a trigger event occurs, a trigger packet is inserted into the ATB output. See *HTM Trigger Event Register, HTMTRIGEVT* on page 3-14 for details of the Trigger Event Register.

1.5.2 FIFO operation

During a single cycle the FIFO can receive up to 18 bytes, and sends out up to four bytes. The input and output ports are controlled by two separate clock signals that can be asynchronous to each other. During operation, address, auxiliary, and data packets are appended to the current contents of the FIFO. If the FIFO is almost full, just lower than the programmed FIFO level, only address packets are stored. Data suppression is deactivated when the space left in the FIFO is larger than the programmed level.

The implementation of the FIFO is based on a circular buffer, but the operation can be illustrated with a simple FIFO model as shown in Figure 1-4 on page 1-14.

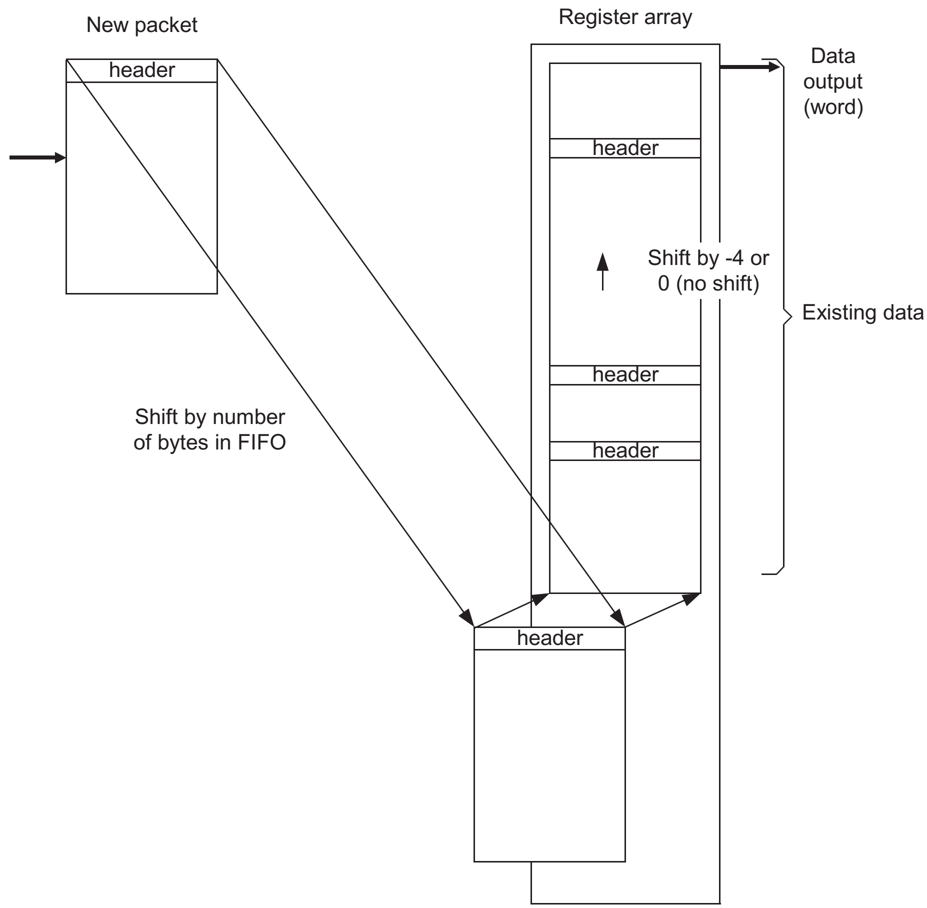


Figure 1-4 HTM FIFO operation

Data is continuously shifted out by the ATB interface, four bytes at a time. If there are less than four bytes in the FIFO, the FIFO waits for more data before transmitting data, except for the case of a flush operation on the ATB bus, when the remaining data in the FIFO is output. Flushing the FIFO can only be done through the ATB interface or by setting the PROG bit in the HTMCONTROL register.

If there is not enough room in the FIFO for additional bytes, the packet is dropped and an overflow packet is output to indicate a loss of trace.

When the PROG bit is set in the HTMCONTROL Register, the HTM trace operation is stopped but the FIFO contents and ATB operation are unaffected.

See Chapter 4 *Protocol Details* for details of packets.

1.5.3 External trace disabling

You can disable AHB trace temporarily without affecting the contents of the FIFO, using the **HTMTRACEDISABLE** external pin. You can disable AHB trace by:

- Setting the **HTMTRACEDISABLE** external pin in the HTM Control Register, HTMCONTROL. This can only be done when bit [6], EXTDISABLE, is HIGH.
- Setting the SWTRACEDISABLE bit in the HTM Control Register.

Unlike setting the PROG bit in the HTM Control Register, using the external trace disable or SWTRACEDISABLE does not affect the operation of the resources such as the counter and sequencer.

In single-core systems, the **HTMTRACEDISABLE** input signal can be connected to the **DBACK** output of the ARM processor. However, because **DBACK** is not closely coupled to AHB transfers, the switching of **DBACK** might not be an accurate indication of whether the current AHB transfer is generated by a debug process. If possible, instead of using **DBACK** for trace filtering, debug software must use the software disable function, the SWTRACEDISABLE bit, in the HTM Control Register to filter out accesses in debug mode.

For multi-core systems, the designer of the SoC must ensure the correct **DBACK** is used if trace must be disabled during debug mode. It is up to the designer of the SoC to determine how the correct **DBACK** is routed to the HTM. In some situations, for example, in a multi-layer AHB system with an HTM connected to an AHB slave at the output port of an AHB bus matrix, it might not be possible to determine which **DBACK** must be used. In this case the use of **DBACK** for trace disabling is not recommended.

Chapter 2
Functional Description

The HTM is a real-time trace module capable of address and data tracing of the AHB bus. This chapter contains the following sections:

2.1 HTM structure

The design of the HTM can be divided into:

- APB (AMBAv3) interface, including the APB register block
- *Trace Control Block* (TCB), including the AHB register block, comparators and resources
- Trace generation block, including the AHB sampler, packet generation, FIFO and ATB interface.

Figure 2-2 on page 2-3 shows a detailed block diagram of the HTM with clock domains marked. For this figure:

- Figure 2-1shows the security and control signals
- Figure 2-3 on page 2-4shows the APB interface signals
- Figure 2-4 on page 2-4shows the AHB interface signals
- Figure 2-5 on page 2-5shows the ATB interface signals

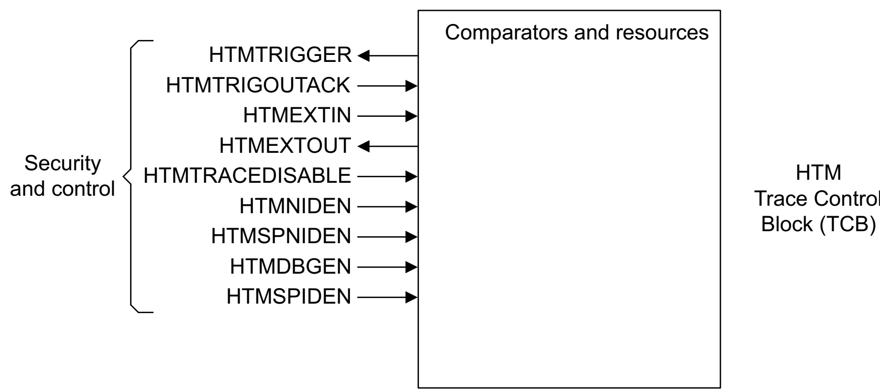


Figure 2-1 HTM security and control signals

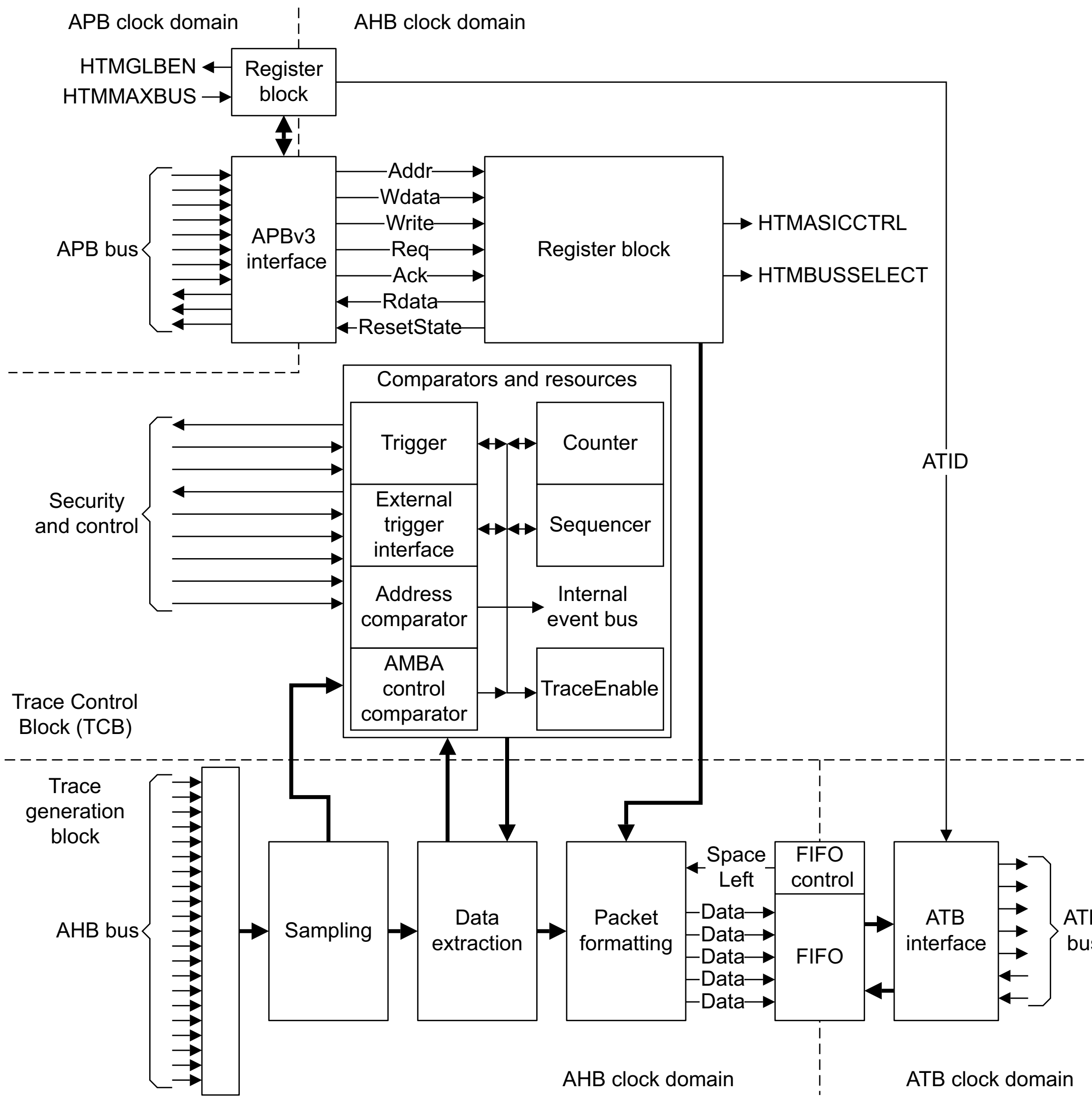


Figure 2-2 HTM detailed block diagram with clock domains

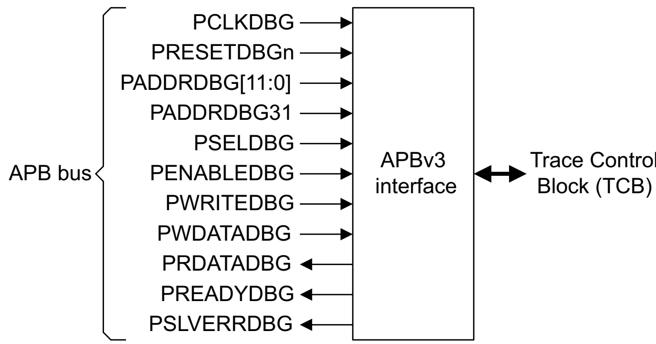


Figure 2-3 HTM APB interface signals

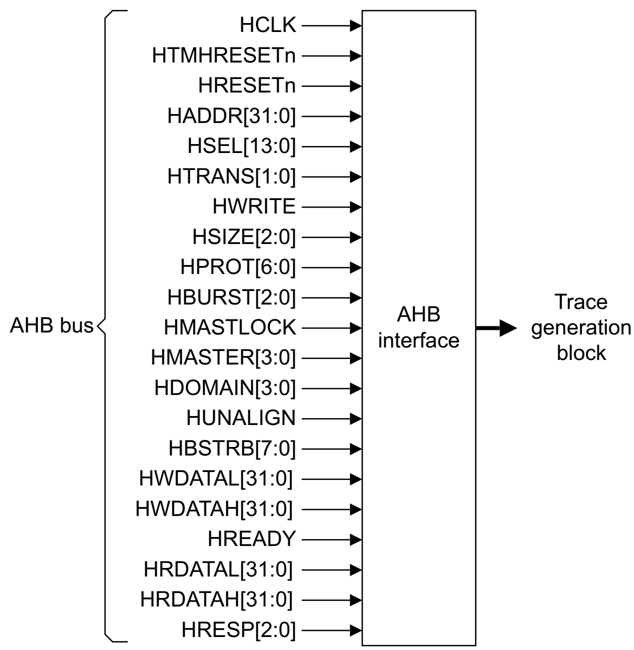


Figure 2-4 HTM AHB interface signals

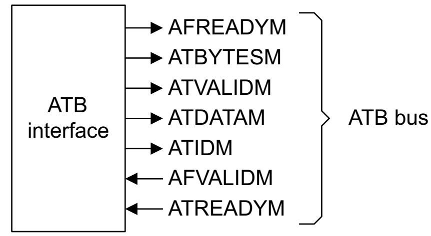


Figure 2-5 HTM ATB interface signals

2.2 HTM trace control block

The TCB contains the address and control comparators, and other logic blocks that can generate events. These logic blocks are called resources. In addition, the AHB register block is also a part of the trace control because most of these registers define the behavior of the HTM resources. Figure 2-9 on page 2-11 shows the full range of HTM resources.

For details of HTM64 and HTM32 resources see *HTM64 and HTM32 features summary* on page 5-2.

2.3 HTM idle status

The idle status bit indicates if the whole HTM is in an idle state, enabling the trace output interface to be switched to test mode, or switching off the trace system safely. The idle status bit is implemented as bit [12] in the HTMSTATUS register. See *HTM Status Register, HTMSTATUS* on page 3-8.

The idle status bit is controlled by a simple finite state machine, see Figure 2-6.

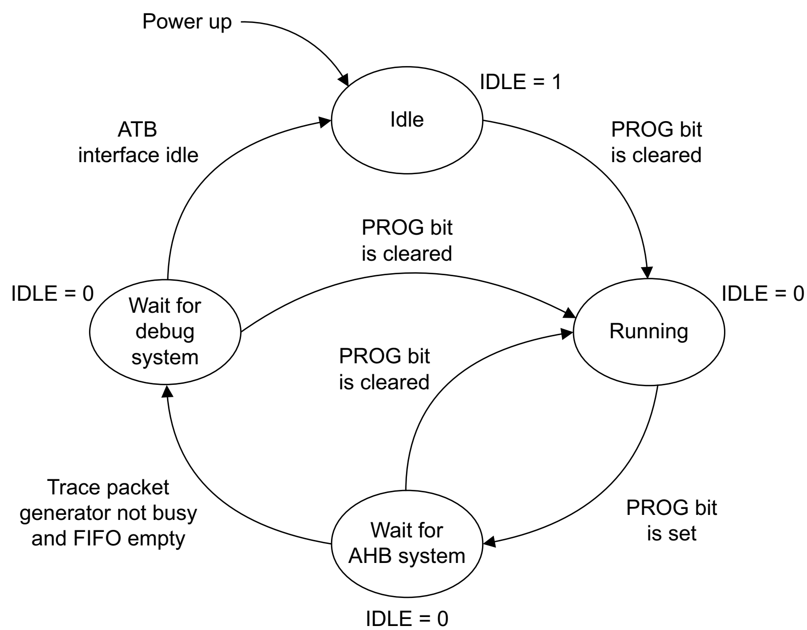


Figure 2-6 Idle status state machine

The idle status bit is set to 1 only if the state machine is in idle state. The design of the state machine reflects the structure of the HTM, as shown in Figure 2-7 on page 2-8.

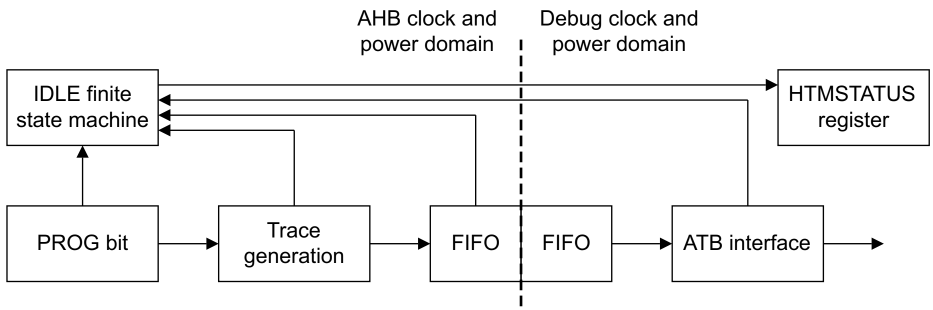


Figure 2-7 State machine structure in HTM

When the trace is started, the state machine can only return to idle state if:

- the PROG bit is set, that is, trace is stopped
- trace packet generator is stopped, and the FIFO is empty
- the remaining data on the ATB interface is outputted.

If power down signal is asserted and signal clamping option is set in the Verilog code, the IDLE bit reads as 0.

There are several situations where the IDLE bit must be checked before a required operation is carried out:

- integration testing
- topology detection
- power down of trace system.

If these operations are carried out while the HTM is not idle, that is, data is still on the ATB interface, loss or corruption of trace data can result.

When programming the HTM registers you must enable all the changes at the same time. For example, if the counter is reprogrammed, it might start to count based on incorrect events, before the trigger condition has been correctly set up.

You can use the HTM programming bit in the HTM Control Register (see *HTM Control Register, HTMCONTROL* on page 3-12) to disable all operations during programming. To do this you must follow the procedure shown in Figure 2-8 on page 2-9. When the Idle Status bit is clear (b0) you must not change the HTM control settings, because this can lead to Unpredictable behavior.

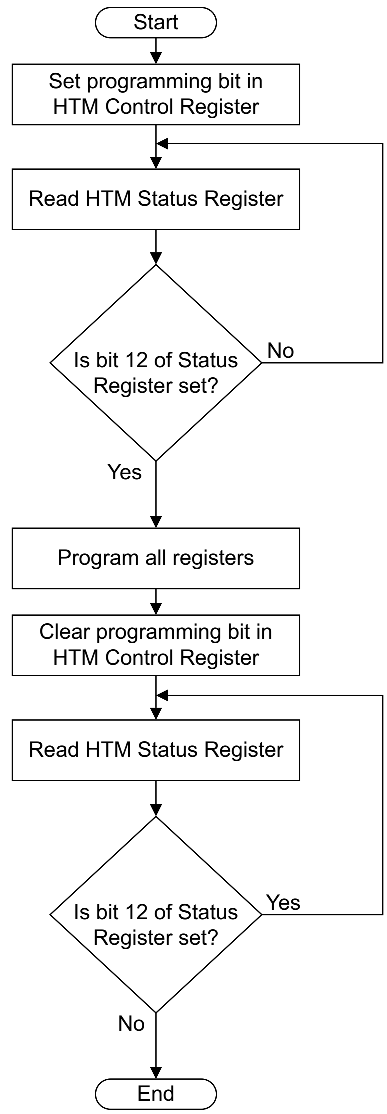


Figure 2-8 Programming HTM registers

It is not necessary for the core to be in debug state while the registers are being programmed.

2.4 HTM trace generation blocks

The trace generation blocks include:

Sampling registers for AHB signals

The sampling logic registers all the signals from the traced AHB.

Packet generator

The packet generator filters the AHB transfer based on the **TraceEnable** signal from the Comparators and Resources Block, and then packs the AHB transfer information into an address packet, auxiliary packet, and data packet. Data suppression is also done here.

FIFO The FIFO controller monitors the space remaining in the FIFO and controls the shift in and shift out of data.

ATB interface

The ATB interface outputs the trace packets, and interfaces with the FIFO for ATB flush functions.

2.5 HTM resources

Figure 2-9 shows the full range of HTM resources.

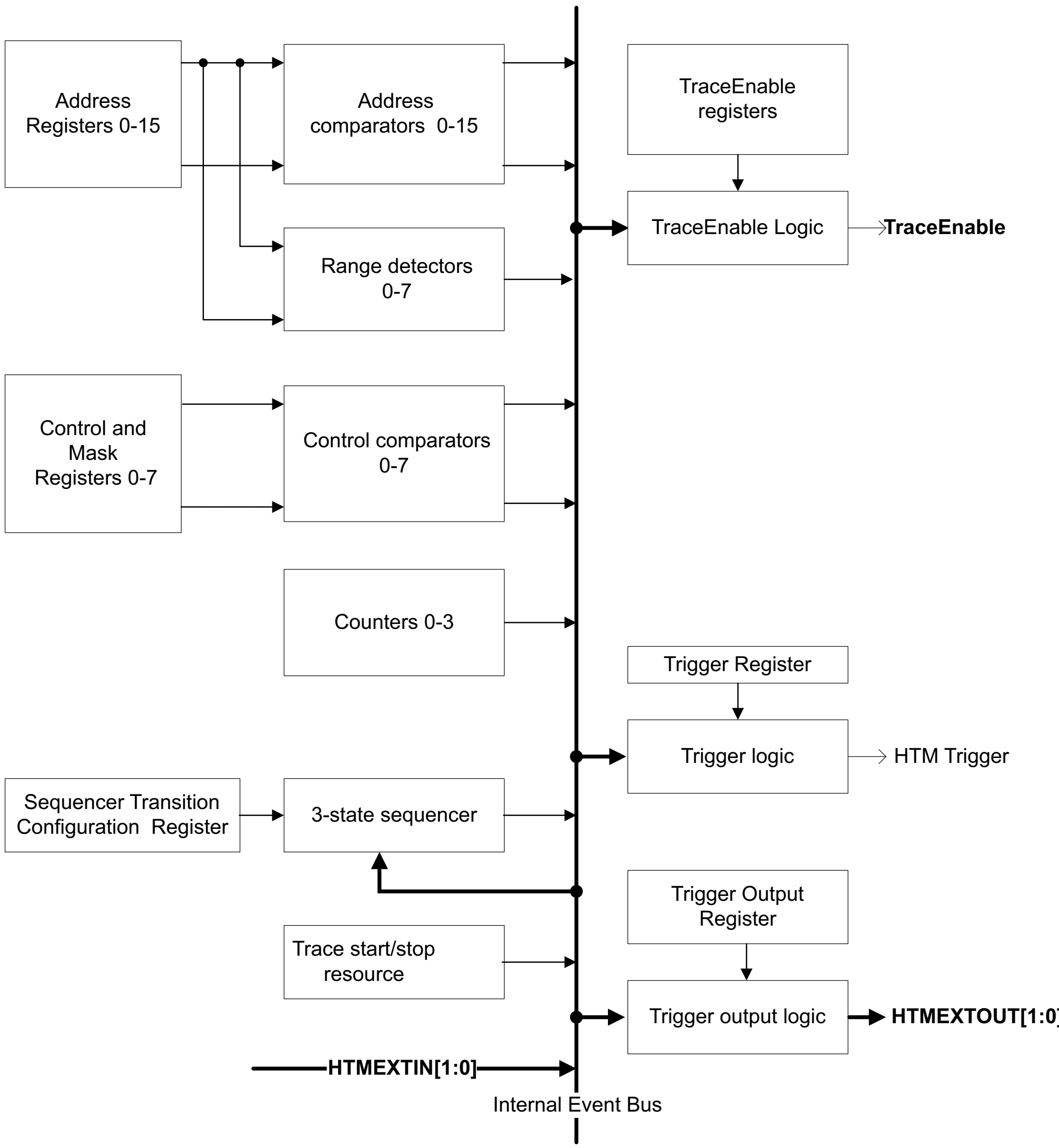


Figure 2-9 HTM resources

HTM resources are described in the following sections:

- *Internal event bus*
- *Boolean combinations for defining events*
- *Resource identification* on page 2-13.

The resources are described in general terms in this chapter. *HTM64 and HTM32 features summary* on page 5-2 provides details of the HTM64 and HTM32 resources.

2.5.1 Internal event bus

Some of the resources only require signals from AHB or control registers to generate output, and are called primary resources. Other resources require outputs from other resources, and are called derived resources. To simplify the design, all resource outputs are connected to an internal event bus. Derived resources operate on the signal values on this internal event bus, and various resource control registers in the register blocks.

Derived resources operate on a Boolean function of two different resources in the internal event bus. This is typically specified by a 17-bit register. The encoding of the event and resources is described in *HTM derived resources* on page 2-18.

2.5.2 Boolean combinations for defining events

If A is defined as the first resource match and B as the second match, an event is defined as a function of A and B. The functions and their bit encodings are listed in Table 2-1. The encoding is different from that found in ETMs because in HTM "A" and "NOT A" functions can be replaced by other functions. For example, by setting resource B to 7'b1101111 (type = 3'b110, index = 15), resource B is always true and thus function "A" can be represented by function "A AND B". Similarly, function "NOT A" can be represented by function "NOT A AND B".

Table 2-1 Boolean function encoding for events

Encoding	Function
b000	Reserved
b001	Reserved
b010	A AND B
b011	NOT (A) AND B
b100	NOT (A) AND NOT (B)

Table 2-1 Boolean function encoding for events (continued)

Encoding	Function
b101	A OR B
b110	NOT (A) OR B
b111	NOT (A) OR NOT (B)

A and B are identified with two seven-bit fields. See *Resource identification* for the exact resource encoding.

An event is encoded in three fields using 17 bits in total, as shown in Table 2-2. The first two fields encode the two event resources (see Table 2-4 on page 2-14 and Table 2-3) and the third field encodes the Boolean operation to be applied to them (see Table 2-1 on page 2-12).

Table 2-2 Event encoding

Bit	Description
[16:14]	Boolean function
[13:7]	Resource B
[6:0]	Resource A

Event and resource encoding is shown in Figure 2-10 on page 2-18.

2.5.3 Resource identification

To identify a resource requires seven bits:

- three bits for the resource type
- four bits for the index.

Table 2-3 describes the resource encoding.

Table 2-3 Resource encodings

Bit	Description
[6:4]	Resource type
[3:0]	Resource index

Table 2-4 defines the available resource types and lists the bit encodings used to identify them.

Table 2-4 HTM resource identification encoding

Resource type	Index range	Description
b000	0-15	Single address comparator 0-15
b001	0-7	Address range comparator 0-7 Represents the range between two single address comparators
b010	0-7	AMBA Control comparator 0-7
b011	-	Reserved
b100	0-3	Counter 0-3 at zero
b101	0-2 3-14 15	Sequencer in states1-3 Reserved Trace enable start/stop resource
b110	0-3 4-14 15	External inputs 1-4 Reserved Hard-wired input (always true)
b111	-	Reserved

When a particular resource is active, its output is a logical 1.

Note

- To permanently enable an event, you can specify the hard-wired input to A, using either function "A OR B" or "A OR NOT B" to produce the always 1 result.
- To permanently disable an event, you can specify the hard-wired input to A, using either function "NOT A AND B" or "NOT A AND NOT B" to produce the always 0 result.

2.6 HTM primary resources

Primary resources are described in:

- *Single address comparators*
- *Address range comparators* on page 2-16
- *AMBA control comparators* on page 2-16
- *External inputs* on page 2-17.

2.6.1 Single address comparators

The programmer's model supports up to 16 address comparators. See *HTM64 and HTM32 features summary* on page 5-2 for details of address comparators for HTM64 and HTM32. Each address comparator is composed of two registers: an address value register and an address access type register. The address access type register determines if the trace address must be instruction fetch or data accesses.

The address comparators have two operation modes:

Normal mode

An address comparator asserts its output when an address match takes place and the access type matches the address type register. The outputs from address comparators are asserted for only one cycle. This ensures that counters and sequencer only trigger once per transfer.

Sticky mode An address comparator updates its output when an active traced transfer is detected (**HREADY**=1 and **HTRANS[1]**=1) and trace is not prohibited for security. If the address matches, the output stays HIGH until the next active traced transfer is detected.

The single address comparators are controlled by the HTMADDR*x* and the HTMADDRTYPE*x* registers. These registers are shared with address range comparators.

Note

The *x* in the register name represents the applicable register range, for example, HTMADDR*x* represents any one of HTMADDR0-15. See *HTM register summary* on page 3-3 for the possible register ranges.

2.6.2 Address range comparators

The HTM architecture supports up to eight address range comparators. Each comparator uses a pair of address value registers to define address ranges. For example, Address Range comparator 0 uses HTMADDR0 and HTMADDR1, and Address Range comparator 1 uses HTMADDR2 and HTMADDR3.

The comparison output is high if Value_A <= Current Address <= Value_B

This behavior is different from ETM, because it allows address `0xFFFFFFFF` to be included in the address range. An address value register can be used by a single address comparator and an address range comparator at the same time.

The output from the address range comparators has the same operation modes as single address comparators, namely normal and sticky modes. When using address range comparators, the address type setting in the HADDRTYPE*x* registers of the two single address comparators involved must be of the same address type to ensure correct operation.

2.6.3 AMBA control comparators

The programmer's model supports up to eight AMBA control comparators. Each AMBA control comparator contains three registers:

Control Selection Register, HTMHCTRLSEL*x*

selects the AMBA control signals to be compared.

Control Value Register, HTMHCTRLVAL*x*

reference compare value for the comparisons.

Control Mask Register, HTMHCTRLMASK*x*

a mask of the comparisons.

The AMBA control comparators have two operation modes:

Normal mode

An AMBA control comparator asserts its output when a control signal match takes place. The outputs from the AMBA control comparators are asserted for a single cycle (when **HREADY**=1). This ensures that the counters and sequencers are only triggered once per transfer.

Sticky mode

An AMBA control comparator updates its output when an active transfer is detected (**HREADY**=1 and **HTRANS[1]**=1) and trace is not prohibited by security settings. If the control signals match, the output stays HIGH until the next active traced transfer is detected.

2.6.4 External inputs

External inputs enable activities from other devices to trigger the HTM.

Note

In a typical CoreSight system, the external inputs are connected to the ECT. Alternatively, you can add extra trace filtering resources by connecting the output of the additional resources to the external inputs.

2.7 HTM derived resources

Derived resources operate on a Boolean function of two different resources in the internal event bus. This is typically specified by a 17-bit register. See *Boolean combinations for defining events* on page 2-12 and *Resource identification* on page 2-13 for details of the encoding. Figure 2-10 shows the encoding of the event and resources.

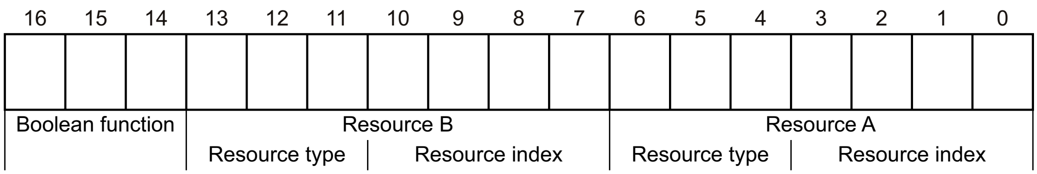


Figure 2-10 Internal event bus resource encoding

The derived resources use the signal from the internal event bus, perform the required function and then send output back to the bus.

HTM derived resources are described in:

- *Counter resources*
- *External outputs* on page 2-19
- *Sequencer* on page 2-19
- *Trace start/stop control* on page 2-20.

2.7.1 Counter resources

The HTM programmer's model supports up to four counter resources. The counter is a 16-bit down counter. It decrements when an enable event takes place, which is defined by the Counter Enable Register.

The following registers define the operation of the counter:

Counter Enable Register, HTMCNTENABLE*x*

This 17-bit register specifies the condition for the counter to decrement.

Counter Reload Value Register, HTMCNTRELDVAL*x*

This 16-bit wide register specifies the starting value of the counter. The counter is automatically loaded with this value when this register is programmed. The counter is then reloaded with this value when a counter reload event occurs. The counter reload event is specified by the Counter Reload Event Register.

Counter Reload Event Register, HTMCNTRELDEVT*x*

The Counter Reload Event Register specifies the condition for the counter to reload. This 17-bit wide register defines the standard event expression.

Counter Value Register, HTMCNTVALUE*x*

This 16-bit wide register stores the current value of the counter.

The counter outputs an event when the counter value is 0.

2.7.2 External outputs

External outputs enable HTM activities to trigger other devices. Each external output is controlled by a 17-bit register using the standard resource encoding format described in Table 2-3 on page 2-13.

2.7.3 Sequencer

The HTM programmer's model supports one sequencer. The sequencer has three states. This produces six different state transition possibilities. These six state transitions are defined by six transition event registers, HTMSEQEVT*x*, which are 17-bit. The encoding is based on the resource event encoding described in *HTM resources* on page 2-11. In addition, a status register, HTMSEQSTATE, is also provided to indicate current state. If you require multiple-stage trigger schemes, the trigger event is usually based on a sequencer state. If the trigger is derived from a single event, the sequencer is not required.

Note

If the current state of the sequencer is written with a value other than that for states 1, 2, or 3, behavior is undefined.

Figure 2-11 on page 2-20 shows the sequencer state diagram.

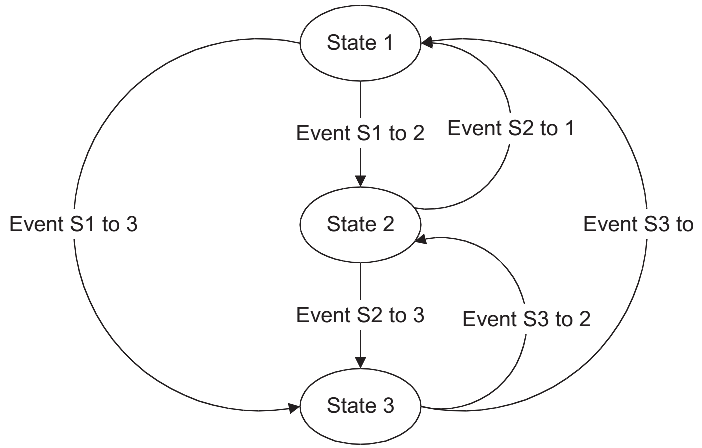


Figure 2-11 Sequencer state diagram

The sequencer has three possible next states (itself and two others), and can change state on every clock cycle. The state transitions are controlled with events.

On every cycle the sequencer does one of the following:

- remains in the current state
- moves to one of the other two states.

On debug reset, the sequencer goes to State 1.

You can read and write the current state of the sequencer.

The behavior of the sequencer is Unpredictable if it reaches a state where either of the two state transition events has not been programmed. If two contradictory state transition events are active, or neither event is active, no state transition occurs.

2.7.4 Trace start/stop control

The use of start/stop control can be enabled or disabled by setting/clearing of the SSENABLE bit in the HTMTRACECTRL register. If the SSENABLE bit is zero, the trace operates regardless of the status of the trace start/stop control. The trace function can be started or stopped automatically when a particular address is detected. The Trace Start/Stop Resource Control Register, HTMSTARTSTOP, identifies the single address comparators that hold trace start and stop addresses. This 32-bit wide register is divided into:

Lower half Indicates which single address comparators are used for start addresses.

Upper half Indicates which single address comparators are used for stop addresses.

2.8 HTM trace filtering

The trace filtering mechanism is controlled by a signal called **TraceEnable**. Operation of this signal is defined by the following registers:

TraceEnable Event Register, HTMTRACEEVT

The TraceEnable Event Register is used to define if the current AHB activities can be traced based on the current status of resources. This 17-bit wide register uses the resource and event encoding described in *Resource identification* on page 2-13. and *HTM resources* on page 2-11.

Trace Start/Stop Control Register, HTMSTARTSTOP

See *Trace start/stop control* on page 2-20.

TraceEnable Control Register, HTMTRACECTRL

The TraceEnable Control Register determines the generation of the **TraceEnable** signal according to:

- whether the Trace Start/Stop Control Register is used,
- whether all addresses must be included and trace filtering is purely based on address range exclusion
- the definition of include and exclude address range comparators, and the selection of address range comparators.

TraceEnable Control2 Register, HTMCTRL2

This register specifies whether certain addresses are to be included or excluded from the address comparison. Each bit represents an address stored in each single address comparator register. The register is 32 bits wide and is divided into:

Lower half defines included address comparisons.

Upper half defines the excluded address comparisons.

Figure 2-12 on page 2-22 shows **TraceEnable** signal generation logic.

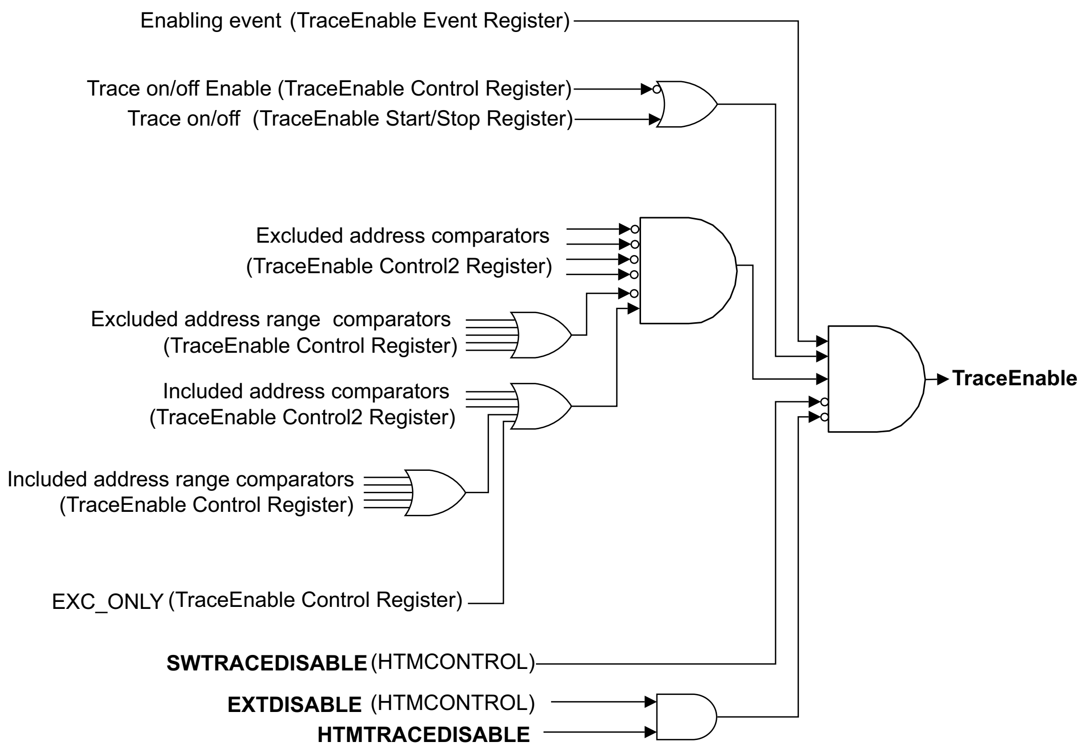


Figure 2-12 TraceEnable signal generation logic

2.8.1 External Trace Disable and Software Trace Disable

It is possible to temporarily disable trace generation by an external signal, **HTMTRACEDISABLE**, or by writing to the SWTRACEDISABLE bit in **HTMCONTROL**. This can be used, for example, to disable trace when the CPU is in debug mode. This trace disable feature does not affect the operation of resources. See *External trace disabling* on page 1-15.

2.9 HTM trigger unit

The HTM contains a trigger unit that inserts a trigger packet into the ATB data stream. Figure 2-13 shows the trigger unit. The Trigger Event Register, HTMTRIGEVT, is a 17-bit register using the resource encoding described in *HTM resources* on page 2-11.

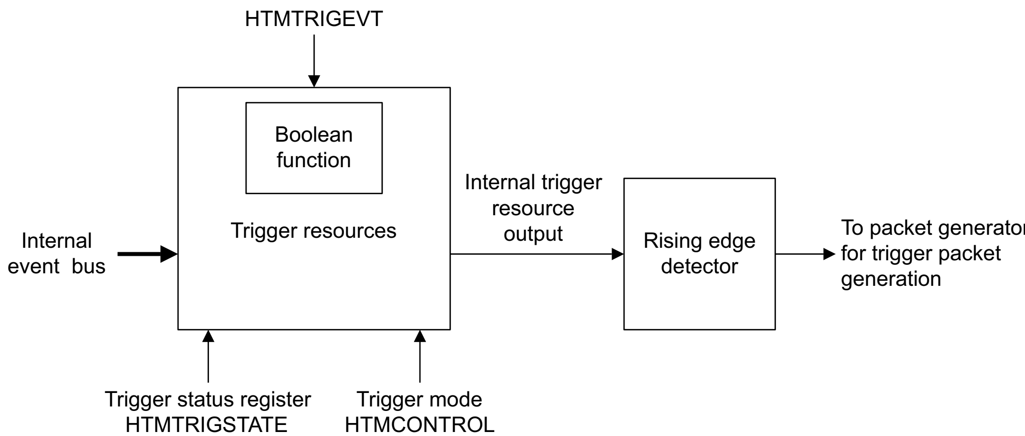


Figure 2-13 HTM trigger unit

After a trigger occurs, the debugger can determine if the trace must be stopped, and when it must be stopped.

If the trigger event is multicycle, only the first cycle causes the trigger packet to be inserted into the data stream.

The trigger has two operation settings:

Single Trigger

After the trigger is asserted, the Trigger Status Register, HTMTRIGSTATE, is set. This prevents further triggers from taking place.

Multi-trigger

The trigger can be output multiple times without clearing the HTMTRIGSTATE Register.

A trigger output is provided at the top level, indicating a nonsuppressed trigger has taken place. After a trigger takes place, the trigger output stays asserted until the trigger acknowledge signal is asserted, or when the Trigger Generation Event Register is reprogrammed.

2.10 Endianness support and bus width support

This section describes:

- *Endianness support*
- *Bus width support*.

2.10.1 Endianness support

HTM supports big-endian and little-endian modes. The input signal **BUSENDIAN** determines the mode. Based on this signal, HTM extracts the data from the data bus and packages the data correctly.

The **BUSENDIAN** input is a one-bit signal. Table 2-5 lists the encoding for the supported endian modes.

Table 2-5 BUSENDIAN values and endianness

BUSENDIAN value	Endian mode	Examples
`0x0`	Little-endian or byte invariant big-endian mode (BE-8). BE-8 is supported on ARM11.	Little endian 32-bit bus [31:0] = {byte3,…, byte0} 64-bit bus [63:0] = {byte7,… , byte0} BE-8 (not supported on ARM7, ARM9, or ARM10) 64-bit bus [63:0] = {byte0,… , byte7}
`0x1`	Big-endian (BE-32), supported on ARM10 and ARM11	32-bit bus [31:0] = {byte0,…, byte3} 64-bit bus [63:0] = {byte4-byte7, byte0-byte3} Supported on ARM10 and ARM11

Set **BUSENDIAN** HIGH if you are using word invariant big-endian mode (BE-32). Do not change the setting of **BUSENDIAN** during trace operation.

2.10.2 Bus width support

HTM supports both 64-bit data buses and 32-bit data buses. An input signal, **BUSWIDTH**, indicates the bus width for current transfers. You can change this signal so that HTM can be shared between a 32-bit bus and a 64-bit bus. However, the change must only be performed when HTM trace is disabled.

When HTM64 is connected to a 32-bit bus, the higher 32 bits of the data inputs must be tied LOW. The **BUSWIDTH** input signal and the higher 32 bits of data bus are not available on HTM32.

See *Bus select output* on page 1-10 for a description of the signals BUSENDIAN and BUSWIDTH. See *HTM64 and HTM32 features summary* on page 5-2 for bus width information about HTM64 and HTM32 versions of HTM.

Chapter 3
Programmer's Model

The HTM registers are all accessible from the Debug APB bus. This chapter contains the following sections:

3.1 About the programmer's model

When programming the HTM registers you must enable all the changes at the same time. For example, if the counter is reprogrammed, it might start to count based on incorrect events, before the trigger condition has been correctly set up.

You can use the programming bit, PROG, in the HTM Control Register, HTMCONTROL, to disable all resources and trace operations during programming.

3.1.1 APB interface

APB is a simple low-cost interface used to provide access to the programmable control registers of peripheral devices. It has the following features:

- unpipelined protocol, that is, a second transfer cannot start before the first transfer completes
- every transfer takes at least two cycles.

3.1.2 Programming resources

HTM programming resources are described in Chapter 2 *Functional Description*.

3.2 HTM register summary

The following applies to the registers used in the HTM:

- The base address of the HTM is not fixed, and can be different for any particular system implementation. However, the offset of any particular register from the base address is fixed.
- All registers are word-aligned and must be accessed as 32-bit.
- Reserved or unused address locations must not be accessed because this can result in Unpredictable behavior.
- Reserved or unused bits of registers must be written as zero, and ignored on read unless otherwise stated in the relevant text.
- All register bits are reset to a logic 0 by a system or power-on reset unless otherwise stated in the relevant text.
- All registers support read and write accesses (R/W) unless otherwise stated in the relevant text. A write (W) updates the contents of a register and a read (R) returns the contents of the register.

The following sections describe the HTM registers:

- *HTM registers*
- *HTM detailed register descriptions* on page 3-8.

3.2.1 HTM registers

Table 3-1 lists the HTM registers in base offset order.

Table 3-1 Summary of HTM registers

Address offset	Name	Type	Width (bits)	Reset value	Description
`0x000`	HTMGLBCTRL[a]	R/W	1	0'b0	*HTM Global Control Register, HTMGLBCTRL* on page 3-8
`0x004`	HTMSTATUS[a]	RO	13	13'bxxxxxx11	*HTM Status Register, HTMSTATUS* on page 3-8
`0x008`	HTMCFGCODE[a]	RO	32	`0xC84A4404` or `0x08484402`	*HTM Configuration Code Register, HTMCFGCODE* on page 3-10
`0x00C`	HTMCFGCODE2[a]	RO	11	`0x-40` or `0x-20`	*HTM Configuration Code 2 Register, HTMCFGCODE2* on page 3-12

Table 3-1 Summary of HTM registers (continued)

Address offset	Name	Type	Width (bits)	Reset value	Description
`0x010`	HTMCONTROL	R/W	9	`0x001`	*HTM Control Register, HTMCONTROL* on page 3-12
`0x014`	HTMTRIGEVT	R/W	17	`0x00000`	*HTM Trigger Event Register, HTMTRIGEVT* on page 3-14
`0x018`	HTMTRIGSTATE	R/W	1	`0x0`	*HTM Trigger Status Register, HTMTRIGSTATE* on page 3-15
`0x01C`	HTMAUXSEL	R/W	4	`0x0`	*HTM AUX Select Register, HTMAUXSEL* on page 3-16
`0x020`	HTMSYNCRELOAD	R/W	12	`0x000`	*HTM Synchronization Counter Reload Register, HTMSYNCRELOAD* on page 3-20
`0x024`	HTMSYNCCOUNT	RO	12	`0x000`	*HTM Synchronization Counter Value Register, HTMSYNCCOUNT* on page 3-21
`0x028`	HTMFIFOLEVEL	R/W	6	`0x00`	*HTM FIFO Level Register, HTMFIFOLEVEL* on page 3-22
`0x030`	HTMSTARTSTOP	R/W	32	`0x00000000`	*HTM Trace Enable START STOP Register, HTMSTARTSTOP* on page 3-22
`0x034`	HTMCTRL2	R/W	32	`0x00000000`	*HTM TraceEnable Control 2 Register, HTMTCTRL2* on page 3-23
`0x038`	HTMTRACEEVT	R/W	17	`0x00000`	*HTM TraceEnable Event Register, HTMTRACEEVT* on page 3-23
`0x03C`	HTMTRACECTRL	R/W	18	`0x00000`	*HTM TraceEnable Control Register, HTMTRACECTRL* on page 3-24
`0x040`	HTMSSTATE	R/W	1	`0x0`	*HTM Start/Stop Status Register, HTMSSTATE* on page 3-25
`0x044`	HTMASICCTRL	R/W	8	`0x00`	*HTM ASIC Control Register, HTMASICCTRL* on page 3-26
`0x048`	HTMBUSSELECT	R/W	3	`0x0`	*HTM Bus Select Register, HTMBUSSELECT* on page 3-27
`0x080-0x0BC`	HTMADDR0-15	R/W	32	`0x00000000`	*HTM Address Comparator Value Registers, HTMADDR0-15* on page 3-28

Table 3-1 Summary of HTM registers (continued)

Address offset	Name	Type	Width (bits)	Reset value	Description
`0x0C0-0x0FC`	HTMADDRTYPE0-15	R/W	12	`0x000`	*HTM Address Type Registers, HTMADDRTYPE0-15* on page 3-28
`0x100-0x1FF`	-	-	-	-	Reserved
`0x200-0x21C`	HTMHCTRLSEL0-7	R/W	5	`0x00`	*HTM AMBA Control Select Registers, HTMHCTRLSEL0-7* on page 3-34
`0x220-0x23C`	HTMHCTRLVAL0-7	R/W	8	`0x00`	*HTM AMBA Control Compare Value Registers, HTMHCTRLVAL0-7* on page 3-36
`0x240-0x25C`	HTMHCTRLMASK0-7	R/W	8	`0x00`	*HTM AMBA Control Compare Mask Registers, HTMHCTRLMASK0-7* on page 3-37
`0x280-0x28C`	HTMCNTRELDVAL0-3	R/W	16	`0x0000`	*HTM Counter Reload Value Registers, HTMCNTRELDVAL0-3* on page 3-37
`0x290-0x29C`	HTMCNTENABLE0-3	R/W	17	`0x00000`	*HTM Counter Enable Registers, HTMCNTENABLE0-3* on page 3-38
`0x2A0-0x2AC`	HTMCNTRELDEVT0-3	R/W	17	`0x00000`	*HTM Counter Reload Event Registers, HTMCNTRELDEVT0-3* on page 3-39
`0x2B0-0x2BC`	HTMCNTVALUE0-3	RO	16	`0x0000`	*HTM Counter Value Registers, HTMCNTVALUE0-3* on page 3-39
`0x300-0x314`	HTMSEQEVT0-5	R/W	17	`0x00000`	*HTM Sequencer Transition Event Registers, HTMSEQEVT0-5* on page 3-40
`0x31C`	HTMSEQSTATE	RO	2	`0x0`	*HTM Sequencer State Register, HTMSEQSTATE* on page 3-41
`0x380-0x38C`	HTMEXTOUTEVT0-3	WO	17	`0x0000`	*HTM External Output Event Registers, HTMEXTOUTEVT0-3* on page 3-42
`0x400`	HTMATIDOUT	R/W	7	`0x00`	*HTM ATB ID Register, HTMATIDOUT* on page 3-42

a. The number of wait states is 1, unless otherwise specified. The number of wait states depends on synchronization between **PCLK** and **HCLK** unless specified. Registers with a fixed number of wait states can be accessed even if AHB is in reset state.

Table 3-2 lists the CoreSight management registers.

Table 3-2 CoreSight management registers

Address offset	Name	Type	Width (bits)	Reset value	Description
`0xFA0`	HTMCLAIMTAGSET	R/W	4	`0x0F`	*HTM Claim Tag Set Register, HTMCLAIMTAGSET* on page 3-43
`0xFA4`	HTMCLAIMTAGCLR	R/W	4	`0x00`	*HTM Claim Tag Clear Register, HTMCLAIMTAGCLR* on page 3-44
`0xFB0`	HTMLOCK_ACCESS	WO	32	-	*HTM Lock Access Register, HTMLOCK_ACCESS* on page 3-44
`0xFB4`	HTMLOCK_STATUS	RO	3	`0x3`	*HTM Lock Status Register, HTMLOCK_STATUS* on page 3-45
`0xFB8`	HTMAUTHSTATUS	RO	8	8'b1-001-00	*HTM Authentication Status Register, HTMAUTHSTATUS* on page 3-47
`0xFC8`	HTMDEVID	RO	32	`0x00000000`	*HTM Device CoreSight ID Register, HTMDEVID* on page 3-48
`0xFCC`	HTMDEV_TYPE	RO	8	`0x43`	*HTM ATB Device Type Register, HTMDEV_TYPE* on page 3-48

Table 3-3 lists the peripheral and component identification registers.

Table 3-3 Peripheral and component identification registers

Address offset	Name	Type	Width (bits)	Reset value	Description
`0xFD0`	HTMPERIPHID4	RO	8	`0x04`	*Peripheral Identification Registers, HTMPERIPHID0-7* on page 3-49
`0xFD4`	HTMPERIPHID5	RO	8	`0x00`	Reserved
`0xFD8`	HTMPERIPHID6	RO	8	`0x00`	Reserved
`0xFDC`	HTMPERIPHID7	RO	8	`0x00`	Reserved
`0xFE0`	HTMPERIPHID0	RO	8	`0x17`	*Peripheral Identification Registers, HTMPERIPHID0-7* on page 3-49
`0xFE4`	HTMPERIPHID1	RO	8	`0xB9`	*Peripheral Identification Registers, HTMPERIPHID0-7* on page 3-49

Table 3-3 Peripheral and component identification registers (continued)

Address offset	Name	Type	Width (bits)	Reset value	Description
`0xFE8`	HTMPERIPHID2	RO	8	`0x4B`	*Peripheral Identification Registers, HTMPERIPHID0-7* on page 3-49
`0xFEC`	HTMPERIPHID3	RO	8	`0x00`	*Peripheral Identification Registers, HTMPERIPHID0-7* on page 3-49
`0xFF0`	HTMCOMPONID0	RO	8	`0x0D`	*Identification Registers, HTMPCOMPONID0-3* on page 3-55
`0xFF4`	HTMCOMPONID1	RO	8	`0x90`	*Identification Registers, HTMPCOMPONID0-3* on page 3-55
`0xFF8`	HTMCOMPONID2	RO	8	`0x05`	*Identification Registers, HTMPCOMPONID0-3* on page 3-55
`0xFFC`	HTMCOMPONID3	RO	8	`0xB1`	*Identification Registers, HTMPCOMPONID0-3* on page 3-55

3.3 HTM detailed register descriptions

This section describes the HTM registers.

3.3.1 HTM Global Control Register, HTMGLBCTRL

The HTMGLBCTRL Register enables the HTM block.

Figure 3-1 shows the bit assignments.

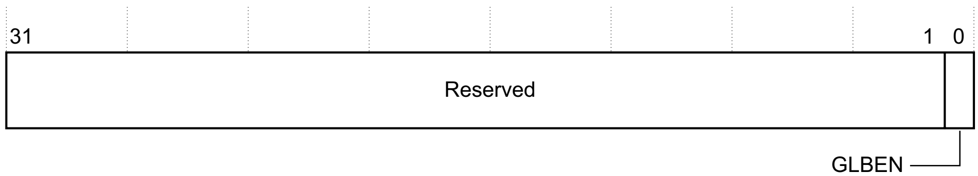


Figure 3-1 HTMGLBCTRL Register bit assignments

Table 3-4 shows the bit assignments.

Table 3-4 HTMGLBCTRL Register bit assignments

Bits	Name	Function
[31:1]	Reserved	Reserved. Read as zero, do not modify.
[0]	GLBEN	Global enable: 1 = enabled 0 = disabled.

When GLBEN is cleared, the ATB interface and trace operations are disabled. The GLBEN register value is also output at the top level so that system designers can use it to turn off the HTM circuits that are in the AHB clock domain. When the GLBEN bit is cleared, the existing states of the HTM resources, FIFO contents, and settings in the HTM can be lost.

3.3.2 HTM Status Register, HTMSTATUS

The HTMSTATUS Register provides the status, FIFO status, and TrustZone status of the HTM.

Figure 3-2 on page 3-9 shows the bit assignments.

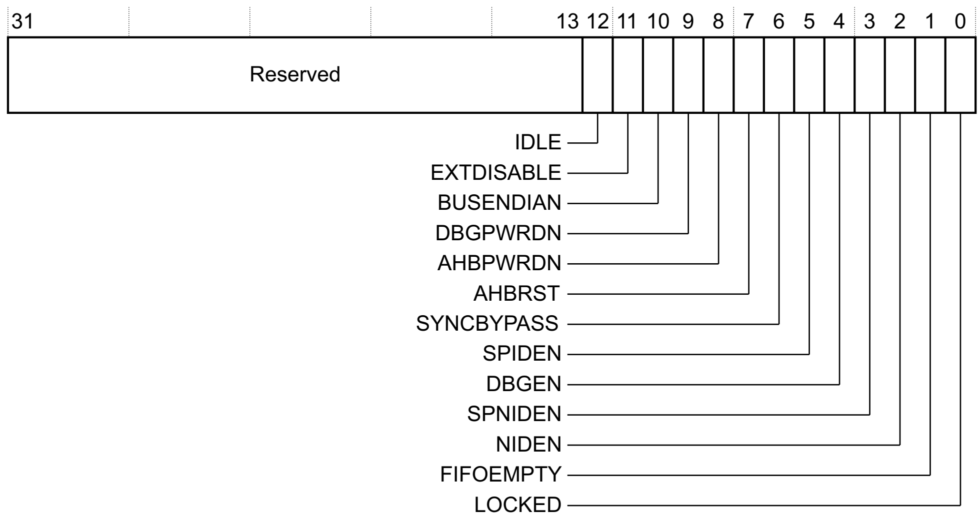


Figure 3-2 HTMSTATUS Register bit assignments

Table 3-5 shows the bit assignments.

Table 3-5 HTMSTATUS Register bit assignments

Bits	Name	Function
[31:13]	Reserved	Reserved. Read as zero.
[12]	IDLE	HTM Idle status.
[11]	EXTDISABLE	External Trace Disable status.
[10]	BUSENDIAN	Current **BUSENDIAN** input status.
[9]	DBGPWRDN	1 when **nCDBGPWRDN** = 0.
[8]	AHBPWRDN	1 when **nCSOCPWRDN** = 0.
[7]	AHBRST	AHB Reset state: 1 = **HRESETn** is LOW (reset) 0 = **HRESETn** is HIGH
[6]	SYNCBYPASS	**SYNCBYPASS** signal status: 1 = **SYNCBYPASS** is HIGH (synchronization bypass enabled) 0 = **SYNCBYPASS** is LOW (synchronization bypass disabled)

Table 3-5 HTMSTATUS Register bit assignments (continued)

Bits	Name	Function
[5]	SPIDEN	TrustZone **SPIDEN** signal status: 1 = **SPIDEN** is HIGH 0 = **SPIDEN** is LOW
[4]	DBGEN	TrustZone **DBGEN** signal status: 1 = **DBGEN** is HIGH 0 = **DBGEN** is LOW
[3]	SPNIDEN	TrustZone **SPNIDEN** signal status: 1 = **SPNIDEN** is HIGH 0 = **SPNIDEN** is LOW
[2]	NIDEN	TrustZone **NIDEN** signal status: 1 = **NIDEN** is HIGH 0 = **NIDEN** is LOW
[1]	FIFOEMPTY	Status of FIFO: 1 = FIFO is empty 0 = FIFO is not empty
[0]	LOCKED	Shows the locked status of the HTM: 1 = Access to the HTM is locked (reset) 0 = Access to the HTM is not locked. Use the HTMLOCK Register to get access. See *HTM Lock Access Register, HTMLOCK_ACCESS* on page 3-44.

3.3.3 HTM Configuration Code Register, HTMCFGCODE

The HTMCFGCODE Register provides the implementation configuration of the HTM.

Figure 3-3 on page 3-11 shows the bit assignments.

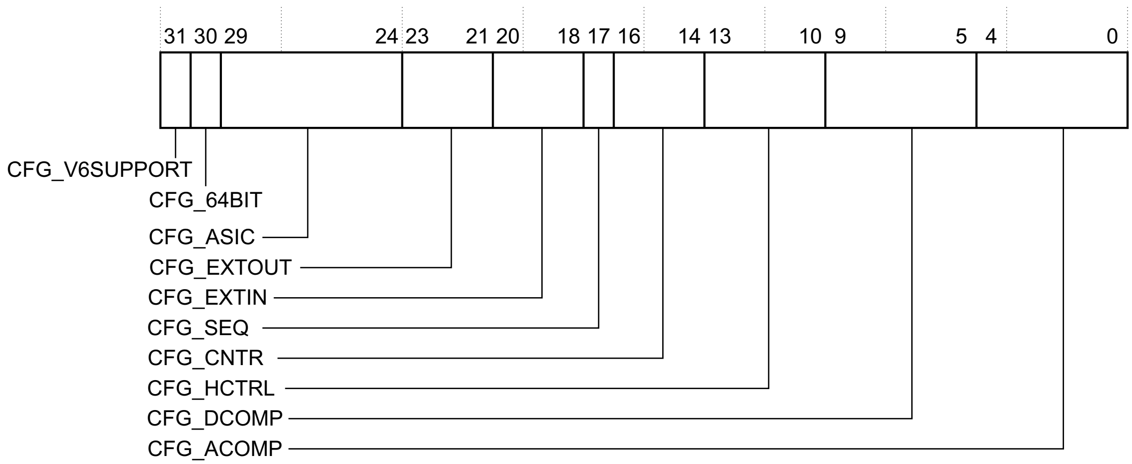


Figure 3-3 HTMCFGCODE Register bit assignments

Table 3-6 shows the bit assignments

Table 3-6 HTMCFGCODE Register bit assignments

Bits	Name	Function
[31]	CFG_V6SUPPORT	ARMv6 AHB extension support: 1 = supported 0 = not supported.
[30]	CFG_64BIT	64-bit AHB support: 1 = supported 0 = not supported.
[29:24]	CFG_ASIC	Width of ASICCTRL register.
[23:21]	CFG_EXTOUT	Number of External Outputs.
[20:18]	CFG_EXTIN	Number of External Inputs.
[17]	CFG_SEQ	Number of Sequencers.
[16:14]	CFG_CNTR	Number of Counters.
[13:10]	CFG_HCTRL	Number of AMBA control comparators.
[9:5]	CFG_DCOMP	Number of data comparators (0).
[4:0]	CFG_ACOMP	Number of pairs of address comparators.

3.3.4 HTM Configuration Code 2 Register, HTMCFGCODE2

The HTMCFGCODE2 register indicates the maximum permitted value of **HTMBUSSELECT** and size of the FIFO. See *HTM Bus Select Register, HTMBUSSELECT* on page 3-27.

Figure 3-4 shows the bit assignments.

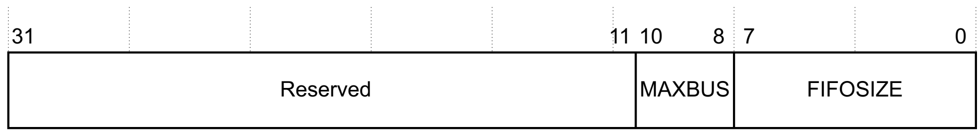


Figure 3-4 HTMCFGCODE2 Register bit assignments

Table 3-7 shows the bit assignments

Table 3-7 HTMCFGCODE2 Register bit assignments

Bits	Name	Function
[31:11]	Reserved	Reserved. Read as zero.
[10:8]	MAXBUS	**HTMMAXBUS** value.
[7:0]	FIFOSIZE	FIFO size in bytes.

3.3.5 HTM Control Register, HTMCONTROL

The HTMCONTROL Register controls the trace operations of the HTM.

Figure 3-5 on page 3-13 shows the bit assignments.

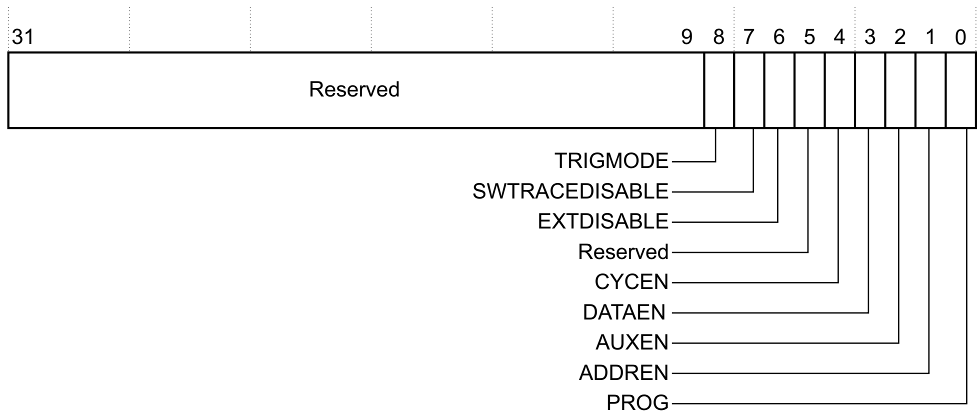


Figure 3-5 HTMCONTROL Register bit assignments

Table 3-8 shows the bit assignments.

Table 3-8 HTMCONTROL Register bit assignments

Bits	Name	Function
[31:9]	Reserved	Reserved. Read as zero, do not modify.
[8]	TRIGMODE	Enable/Disable multiple triggers: 1 = Enable multiple triggers. 0 = Single trigger only. A trigger cannot take place when trigger status is already 1.
[7]	SWTRACEDISBALE	Software Trace Off: 1 = **TraceEnable** disable (existing FIFO content unaffected). 0 = Normal operation for **TraceEnable**.
[6]	EXTDISABLE	Use external trace disable: 1 = **TraceEnable** affected by top level pin **HTMTRACEDISABLE**. 0 = **HTMTRACEDISABLE** has no effect.
[5]	Reserved	Reserved. Read as zero, do not modify.
[4]	CYCEN	CycleCount packet enable: 1 = Enable CycleCount packets. 0 = Disable CycleCount packets.

Table 3-8 HTMCONTROL Register bit assignments (continued)

Bits	Name	Function
[3]	DATAEN	Enable data packet: 1 = Enable data packets. 0 = Disable data packets.
[2]	AUXEN	Enable auxiliary packet: 1 = Enable AUX packets. 0 = Disable AUX packets.
[1]	ADDREN	Enable address packet: 1 = Enable address packets. 0 = Disable address packets.
[0]	PROG	Enables or disables the HTM function during programming: 1 = Programming mode, HTM function disabled. 0 = Normal mode.

The PROG bit setting does not stop the register accesses. It is used to ensure that the HTM is not activated accidentally during the programming process. When the PROG bit is set it has the following effects:

- Trace is disabled. No more trace is produced. Existing data in the FIFO is not destroyed, and is output to the ATB interface when possible.
- The remaining data in the FIFO is output to the ATB even if there are less than four bytes of data held in the FIFO. This is the same as an ATB flush operation.
- The counters, sequencer, and start/stop block are held in their current state.
- The external outputs are forced LOW.

Unlike the *Embedded Trace Macrocell* (ETM), the register values do not change when the PROG bit is cleared from 1 to 0.

3.3.6 HTM Trigger Event Register, HTMTRIGEVT

The HTMTRIGEVT Register defines the trigger event. When the trigger event matches, a trigger packet is generated. The trigger packet generator produces one trigger packet for each rising edge of trigger event.

Figure 3-6 on page 3-15 shows the bit assignments.

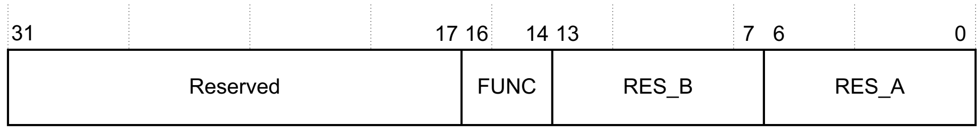


Figure 3-6 HTMTRGEVT Register bit assignments

Table 3-9 shows the bit assignments.

Table 3-9 HTMTRIGEVT Register bit assignments

Bits	Name	Function
[31:17]	Reserved	Reserved. Read as zero, do not modify.
[16:14]	FUNC	Boolean Function between Resource A and B.
[13:7]	RES_B	Resource B.
[6:0]	RES_A	Resource A.

3.3.7 HTM Trigger Status Register, HTMTRIGSTATE

The HTMTRIGSTATE Register indicates if a trigger event has taken place since the trigger event was programmed. It is cleared automatically when HTMTRIGEVT is programmed, or by writing to the trigger status bit directly. The value of the TRIGSTATE bit indicates if a trigger has occurred. If multiple triggers occur, the value stays unchanged (HIGH).

Figure 3-7 shows the bit assignments.

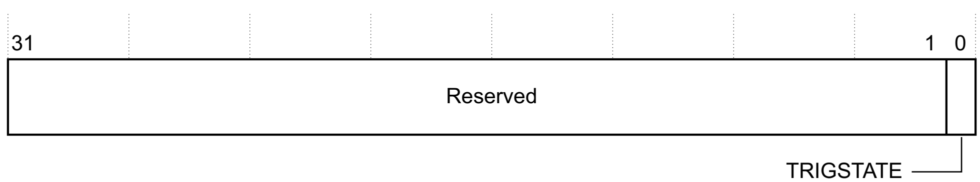


Figure 3-7 HTMTRIGSTATE Register bit assignments

Use the value of the trigger status bit to suppress multiple trigger packets from being generated. This is done by programming the TRIGMODE bit in the HTMCONTROL register:

- If TRIGMODE is set to 0, only the first trigger event causes a trigger packet to be generated. Additional trigger events are suppressed, because TRIGSTATE is set to 1.

- If TRIGMODE is set to 1, a trigger packet is generated for each trigger event regardless of the TRIGSTATE value.

Table 3-10 shows the bit assignments.

Table 3-10 HTMTRIGSTATE Register bit assignments

Bits	Name	Function
[31:1]	-	Reserved. Read as zero, do not modify.
[0]	TRIGSTATE	Trigger status. 1 when trigger event has taken place.

3.3.8 HTM AUX Select Register, HTMAUXSEL

The HTMAUXSEL Register controls the signal to be included in the auxiliary packet, which can contain 12 bits of information.

Figure 3-8 shows the bit assignments.

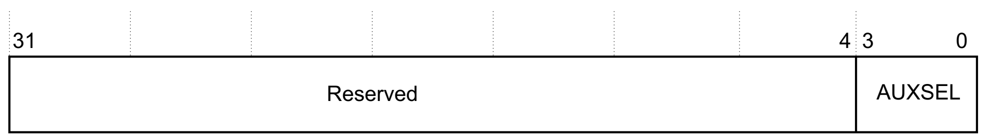


Figure 3-8 HTMAUXSEL Register bit assignments

Table 3-11 shows the bit assignments.

Table 3-11 HTMAUXSEL Register bit assignments

Bits	Name	Function
[31:4]	-	Reserved. Read as zero, do not modify.
[3:0]	AUXSEL	Auxiliary packet selection

Table 3-12 shows the auxiliary packet key.

Table 3-12 Auxiliary packet key

Abbreviation	Meaning
HB	**HBSTRB**
HBR	**HBurst**
HD	**HDOMAIN**
HM	**HMASTER**
HML	**HMASTLOCK**
HP	**HPROT**
HS	**HSIZE**
HT	**HTRANS**
HUN	**HUNALIGN**
HW	**HWRITE**
Res	Response Response is encoded **HRESP**: 00=OKAY 01=ERROR 10=EXCLUSIVE FAILED 11=SPLIT/RETRY
SC	SELCODE is encoded **HSEL** as listed in Table 3-13.
WS	Wait states

Table 3-13 lists the encodings of SELCODE.

Table 3-13 SELCODE encodings

SELCODE	Description
`0x0`	**HSEL[0]** selected
`0x1`	**HSEL[1]** selected
`0x2`	**HSEL[2]** selected

Table 3-13 SELCODE encodings (continued)

SELCODE	Description
`0x3`	**HSEL[3]** selected
`0x4`	**HSEL[4]** selected
`0x5`	**HSEL[5]** selected
`0x6`	**HSEL[6]** selected
`0x7`	**HSEL[7]** selected
`0x8`	**HSEL[8]** selected
`0x9`	**HSEL[9]** selected
`0xA`	**HSEL[10]** selected
`0xB`	**HSEL[11]** selected
`0xC`	**HSEL[12]** selected
`0xD`	**HSEL[13]** selected
`0xE`	No **HSEL** asserted
`0xF`	Error case: more than one **HSEL** asserted

The number of wait states of a transfer is indicated by a 6-bit WS field or a 4-bit WS field:

- If HTMAUXSEL is equal to `0x0`-`0x3`, and the number of wait states exceeds 63 (`0x3F`), the value stays at 63.
- If HTMAUXSEL is equal to `0xC` or `0xE`, when a 4-bit wait state is used, and the number of wait states exceeds 15 (`0xF`), the value stays at 15.

Table 3-14 shows the AUXSEL second byte values.

Table 3-14 Auxiliary packet second byte values

AUXSEL	Debug scenario	AUXSEL second byte HCTRL[11:5]						
		Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
`0x0`	General	HP[0]	HML	HT[0]	Res[1]	Res[0]	HW	WS[5]
`0x1`	General	HP[1]	HP[0]	HT[0]	Res[1]	Res[0]	HW	WS[5]
`0x2`	General	HP[0]	HM[3]	HM[2]	HM[1]	HM[0]	HW	WS[5]
`0x3`	General	HP[1]	HM[3]	HM[2]	HM[1]	HM[0]	HW	WS[5]
`0x4`	Unaligned	HM[2]	HM[1]	HM[0]	HUN	HB[7]	HB[6]	HB[5]
`0x5`	Unaligned	HP[4]	HP[3]	HP[0]	HUN	HB[7]	HB[6]	HB[5]
`0x6`	Unaligned	HP[3]	HP[2]	HP[0]	HUN	HB[7]	HB[6]	HB[5]
`0x7`	Unaligned	HP[5]	HP[1]	HP[0]	HUN	HB[7]	HB[6]	HB[5]
`0x8`	Exclusive	HT[0]	HD[3]	HD[2]	HD[1]	HD[0]	HP[6]	HP[5]
`0x9`	Exclusive	HT[0]	HM[3]	HM[2]	HM[1]	HM[0]	HP[6]	HP[5]
`0xA`	Lock and cache	HML	HD[3]	HD[2]	HD[1]	HD[0]	HP[6]	HP[5]
`0xB`	Lock and cache	HML	HM[3]	HM[2]	HM[1]	HM[0]	HP[6]	HP[5]
`0xC`	Profiling	HP[0]	Res[1]	Res[0]	SC[3]	SC[2]	SC[1]	SC[0]
`0xD`	Profiling	HP[0]	HS[1]	HS[0]	SC[3]	SC[2]	SC[1]	SC[0]
`0xE`	Profiling	HT[0]	HS[1]	HS[0]	HW	HP[3]	HP[2]	HP[1]
`0xF`	Profiling	HBR[2]	HBR[1]	HBR[0]	HUN	HP[3]	HP[2]	HP[1]

Table 3-15 shows the AUXSEL first byte values.

Table 3-15 Auxiliary packet first byte values

AUXSEL	Debug scenario	AUXSEL first byte HCTRL[4:0]				
		Bit 6	Bit 5	Bit 4	Bit 3	Bit 2
`0x0`	General	WS[4]	WS[3]	WS[2]	WS[1]	WS[0]
`0x1`	General	WS[4]	WS[3]	WS[2]	WS[1]	WS[0]
`0x2`	General	WS[4]	WS[3]	WS[2]	WS[1]	WS[0]
`0x3`	General	WS[4]	WS[3]	WS[2]	WS[1]	WS[0]
`0x4`	Unaligned	HB[4]	HB[3]	HB[2]	HB[1]	HB[0]
`0x5`	Unaligned	HB[4]	HB[3]	HB[2]	HB[1]	HB[0]
`0x6`	Unaligned	HB[4]	HB[3]	HB[2]	HB[1]	HB[0]
`0x7`	Unaligned	HB[4]	HB[3]	HB[2]	HB[1]	HB[0]
`0x8`	Exclusive	HW	Res[1]	Res[0]	HP[1]	HP[0]
`0x9`	Exclusive	HW	Res[1]	Res[0]	HP[1]	HP[0]
`0xA`	Lock and cache	HP[4]	HP[3]	HP[2]	HP[1]	HP[0]
`0xB`	Lock and cache	HP[4]	HP[3]	HP[2]	HP[1]	HP[0]
`0xC`	Profiling	HW	WS[3]	WS[2]	WS[1]	WS[0]
`0xD`	Profiling	HW	HM[3]	HM[2]	HM[1]	HM[0]
`0xE`	Profiling	HP[0]	WS[3]	WS[2]	WS[1]	WS[0]
`0xF`	Profiling	HP[0]	HS[1]	HS[0]	HW	HT[0]

3.3.9 HTM Synchronization Counter Reload Register, HTMSYNCRELOAD

The HTMSYNCRELOAD Register holds the reload value of the synchronization down counter. The counter determines how often the address packet and auxiliary packets must be output in full and when an A-SYNC packet must be issued. The counter decrements for each byte of data output to ATB. The counter automatically reloads with this value when reaching 0, or when this register is programmed.

Figure 3-9 on page 3-21 shows the bit assignments.

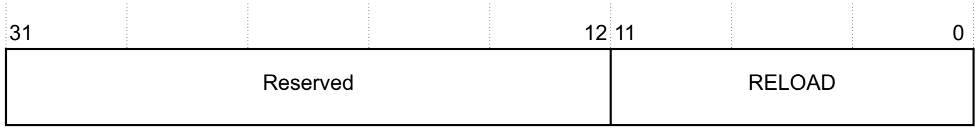


Figure 3-9 HTMSYNCRELOAD Register bit assignments

Table 3-16 shows the bit assignments

Table 3-16 HTMSYNCRELOAD Register bit assignments

Bits	Name	Function
[31:12]	Reserved	Reserved. Read as zero, do not modify.
[11:0]	RELOAD	Reload value.

3.3.10 HTM Synchronization Counter Value Register, HTMSYNCCOUNT

The HTMSYNCCOUNT Register indicates the current value of the down counter. The counter forces the output of complete address or auxiliary packets, and the A-SYNC packet regularly so that external debug hardware can decompress trace information even if part of the trace is lost or unavailable. See *Synchronizing trace* on page 4-31 for more information on synchronization trace.

Figure 3-10 shows the bit assignments.

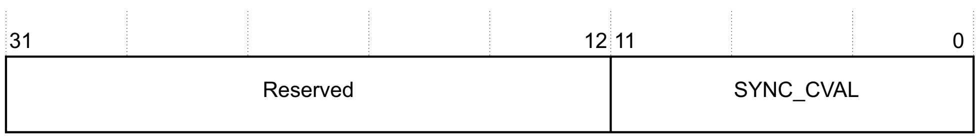


Figure 3-10 HTMSYNCCOUNT Register bit assignments

Table 3-17 shows the bit assignments

Table 3-17 HTMSYNCCOUNT Register bit assignments

Bits	Name	Function
[31:12]	Reserved	Reserved. Read as zero.
[11:0]	SYNC_CVAL	Current down counter value.

3.3.11 HTM FIFO Level Register, HTMFIFOLEVEL

The HTMFIFOLEVEL Register defines the data suppress trigger level of the FIFO. When the remaining space in the FIFO is reached or is lower than the HTMFIFOLEVEL, data suppression begins. Set this register to 0 to disable data suppression.

Figure 3-11 shows the bit assignments.

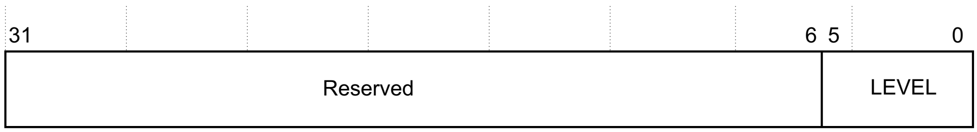


Figure 3-11 HTMFIFOLEVEL Register bit assignments

Table 3-18 shows the bit assignments.

Table 3-18 HTMCONTROL Register bit assignments

Bits	Name	Function
[31:6]	Reserved	Reserved. Read as zero, do not modify.
[5:0]	LEVEL	FIFO data suppress trigger level. Reset at `0x0`.

3.3.12 HTM Trace Enable START STOP Register, HTMSTARTSTOP

The HTMSTARTSTOP Register controls the trace start/stop of the TraceEnable signal. This register identifies the comparators that hold start/stop addresses.

Figure 3-12 shows the bit assignments.

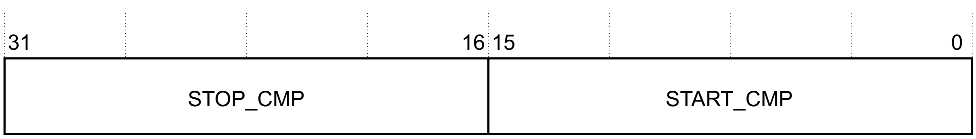


Figure 3-12 HTM STARTSTOP Register bit assignments

Table 3-19 shows the bit assignments

Table 3-19 HTMSTART STOP Register bit assignments

Bits	Name	Function
[31:16]	STOP_CMP	When HIGH, selects single address comparators 15-0 as stop addresses. For example, bit 16 HIGH selects single address comparator 0.
[15:0]	START_CMP	When HIGH, selects single address comparators 15-0 as start addresses. For example, bit 0 HIGH selects single address comparator 0.

3.3.13 HTM TraceEnable Control 2 Register, HTMTCTRL2

The HTMTCTRL2 Register determines which single addresses are excluded from the trace.

Figure 3-13 shows the bit assignments.

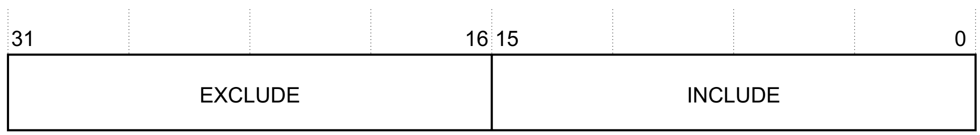


Figure 3-13 HTMCTRL2 Register bit assignments

Table 3-20 shows the bit assignments.

Table 3-20 HTMCTRL2 Register bit assignments

Bits	Name	Function
[31:16]	EXCLUDE	When HIGH, selects single address comparators 15-0 for exclude control. For example, bit 16 selects single address comparator 0.
[15:0]	INCLUDE	When HIGH, selects single address comparators 15-0 for include control. For example, bit 0 selects single address comparator 0.

3.3.14 HTM TraceEnable Event Register, HTMTRACEEVT

The HTMTRACEEVT Register determines the event that is used to generate TraceEnable output.

Figure 3-14 on page 3-24 shows the bit assignments.

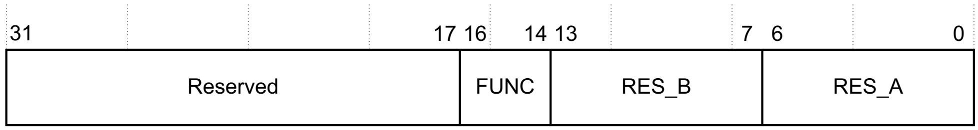


Figure 3-14 HTMTRACEEVT Register bit assignments

Table 3-21 shows the bit assignments.

Table 3-21 HTMTRACEEVT Register bit assignments

Bits	Name	Function
[31:17]	Reserved	Reserved. Read as zero, do not modify.
[16:14]	FUNC	Boolean Function between Resource A and B.
[13:7]	RES_B	Resource B.
[6:0]	RES_A	Resource A.

3.3.15 HTM TraceEnable Control Register, HTMTRACECTRL

The HTMTRACECTRL Register controls the generation of **TraceEnable** signals.

Figure 3-15 shows the bit assignments.

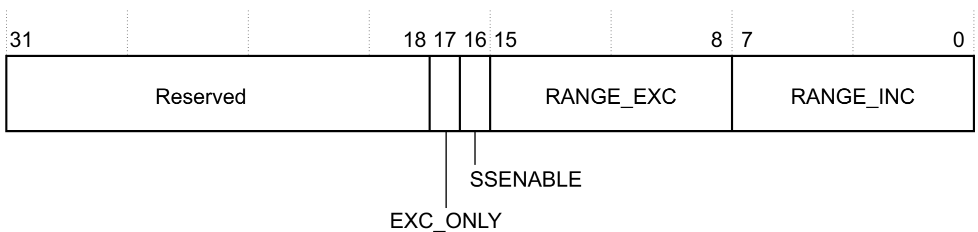


Figure 3-15 HTMTRACECTRL Register bit assignments

Table 3-22 shows the bit assignments.

Table 3-22 HTMTRACECONTROL Register bit assignments

Bits	Name	Function
[31:18]	Reserved	Reserved. Read as zero, do not modify.
[17]	EXC_ONLY	Exclude only. Include all address range and use only address exclusion for address filtering.
[16]	SSENABLE	Trace Start/Stop enable: 1 = Tracing is controlled by trace on and off addresses. 0 = Tracing is unaffected by the trace start/stop logic. The trace start/stop resource is unaffected by the value of this bit.
[15:8]	RANGE_EXC	When HIGH, select address range comparators 7-0 for exclude control. For example, bit 8 HIGH selects address range comparator 0.
[7:0]	RANGE_INC	When HIGH, select address range comparators 7-0 for include control. For example, bit 0 HIGH selects address range comparator 0.

3.3.16 HTM Start/Stop Status Register, HTMSSTATE

The HTMSSTATE register indicates the current status of Trace Start/Stop resource.

Figure 3-16 shows the bit assignments.

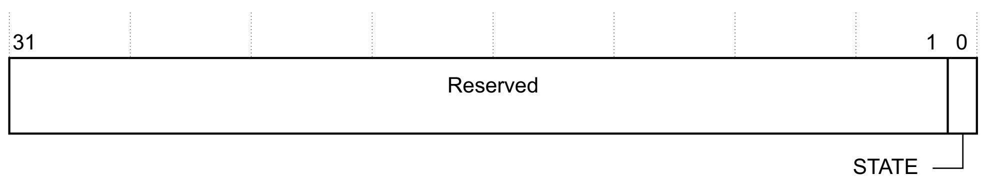


Figure 3-16 HTMSSTATE Register bit assignments

Table 3-23 shows the bit assignments.

Table 3-23 HTMSSTATE Register bit assignments

Bits	Name	Function
[31:1]	Reserved	Reserved. Read as zero, do not modify.
[0]	STATE	Trace Start/Stop state. Only used when SSENABLE in HTMTRACECTRL is HIGH. 1 = trace started. 0 = trace stopped.

3.3.17 HTM ASIC Control Register, HTMASICCTRL

The HTMASICCTRL Register controls the ASIC control output port. The actual usage is implementation-dependent. Most systems might not implement all bits. The number of bits implemented can be determined by reading the HTMCFGCODE Register.

Note

The HTM only implements ASICCTRL[7:0], which is connected to the **HTMASICCTRL[7:0]** port.

Figure 3-17 shows the bit assignments.

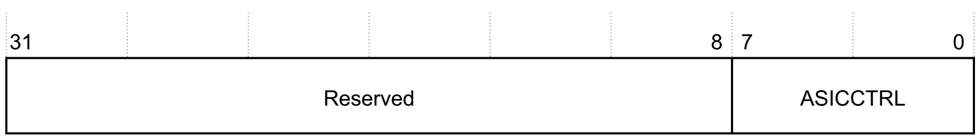


Figure 3-17 HTMASICCTRL Register bit assignments

Table 3-24 shows the bit assignments.

Table 3-24 HTMASICCTRL Register bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved. Read as zero, do not modify.
[7:0]	ASICCTRL	ASIC Control

Figure 3-18 shows an example where **HTMASICCTRL** controls a dummy bus master that is used to inject additional bus traffic to assist in analyzing bus behavior.

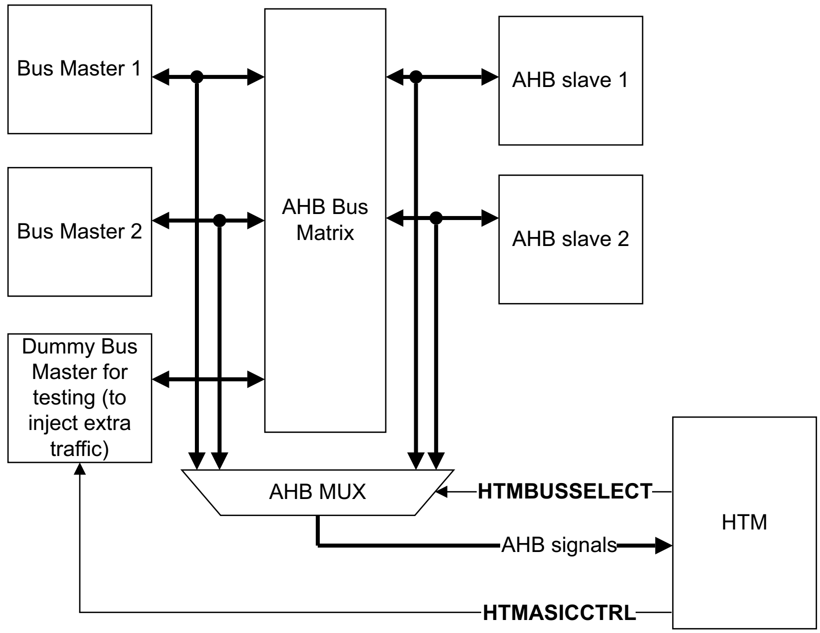


Figure 3-18 HTMASICCTRL example use

3.3.18 HTM Bus Select Register, HTMBUSSELECT

The HTMBUSSELECT Register controls external bus multiplexers if the HTM is to be shared between multiple AHB buses.

Figure 3-19 shows the bit assignments.

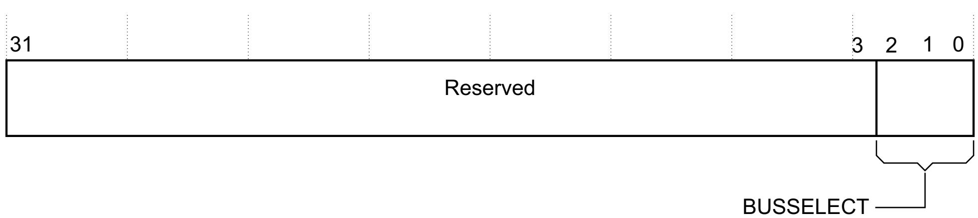


Figure 3-19 HTMBUSSELECT Register bit assignments

Table 3-25 shows the bit assignments.

Table 3-25 HTMBUSSELECT Register bit assignments

Bits	Name	Function
[31:3]	Reserved	Reserved. Read as zero, do not modify.
[2:0]	BUSSELECT	Bus select output value.

3.3.19 HTM Address Comparator Value Registers, HTMADDR0-15

The HTMADDR Registers hold the address values for single address comparators and address range comparators. When used as an address range comparator, two address comparator value registers are used. For example, HTMADDR0 and HTMADDR1 are used for address range comparator 0, and HTMADDR2 and HTMADDR3 are used for address range comparator 1.

Table 3-26 shows the bit assignments.

Table 3-26 HTMADDR Register bit assignments

Bits	Name	Function
[31:0]	ADDR	Address value.

3.3.20 HTM Address Type Registers, HTMADDRTYPE0-15

The HTMADDRTYPE Registers hold the type of access to be detected by the address comparator. When the addresses are used as address range comparators, the setting of HTMADDRTYPE of both addresses must be the same or Unpredictable behavior can occur.

For unaligned transfers, the size must be set to *any* and must use address range comparators to handle address range.

Figure 3-20 on page 3-29 shows the bit assignments.

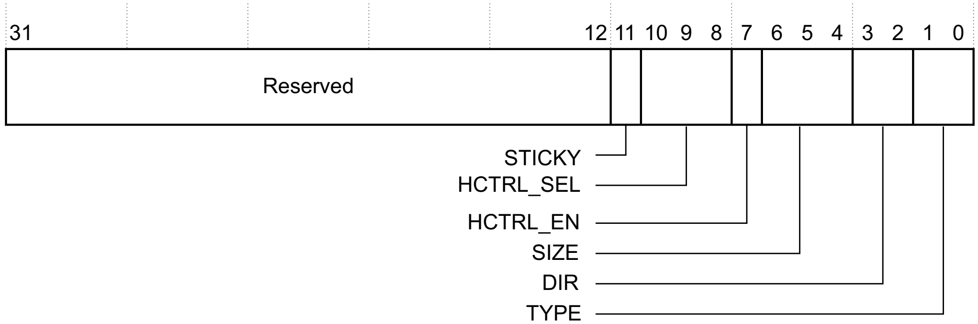


Figure 3-20 HTMADDRTYPE Register bit assignments

Table 3-27 shows the bit assignments.

Table 3-27 HTMADDRTYPE Register bit assignments

Bits	Name	Function
[31:12]	Reserved	Reserved. Read as zero, do not modify.
[11]	STICKY	Output mode: 0 = Output only high when **HREADY**=1 1 = Output stays unchanged until next active traced transfer detected (**HREADY**=1 and **HTRANS[1]** =1 and trace not prohibited by security setting)
[10:8]	HCTRL_SEL	AMBA control filtering 7 = Use AMBA control comparator 7 ... 1 = Use AMBA control comparator1 0 = Use AMBA control comparator0
[7]	HCTRL_EN	Enable AMBA control filtering
[6:4]	SIZE	Size - Address window size Others = reserved 011 = Doubleword 010 = Word 001 = Halfword 000 = Byte

Table 3-27 HTMADDRTYPE Register bit assignments (continued)

Bits	Name	Function
[3:2]	DIR	Access Direction 11 = Reserved 10 = Any 01 = Write only 00 = Read only
[1:0]	TYPE	Access Type 11 = Reserved 10 = Any 01 = Data only 00 = Instructions only

HTM address matching

This section describes:

- *Single address matching*
- *Address range matching* on page 3-31.

Single address matching

The size field is not used to compare directly with HSIZE. It is used to define a window size for address compare. Table 3-28 gives an example.

Table 3-28 SIZE field with HSIZE

Signal	Address	Size field in HTMADDRTYPE0	HSIZE	Address window used for compare
HTMADDR0	`0x1000`	WORD (3'b010)	-	`0x1000` - `0x1003`
HADDR	`0x1002`	-	HWORD (3'b001)	`0x1002` - `0x1003`

As shown in Figure 3-21 on page 3-31 this results in a match even if the address value is not exactly the same because the address windows overlap.

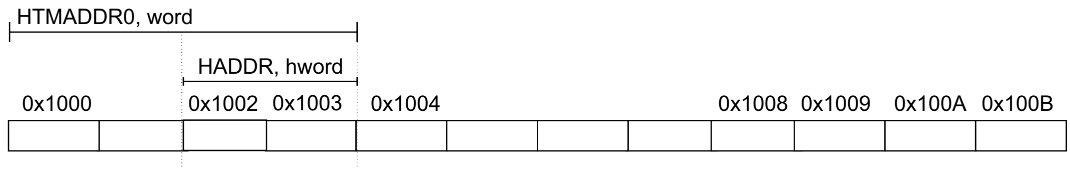


Figure 3-21 Single address match with overlapping address windows

Similarly, with the following values there is also a match as shown in Figure 3-22:

HTMADDR0 value `0x1002`

size word

HADDR value `0x1004`

HSIZE value word

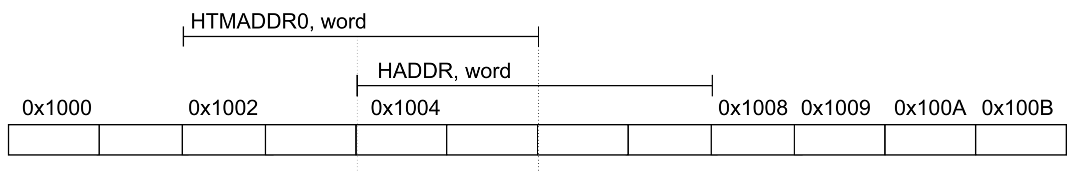


Figure 3-22 Second single address matching example

In contrast, with the following values there is no match, as shown in Figure 3-23:

HTMADDR0 value `0x1004`

size hword

HADDR value `0x1003`

HSIZE value byte

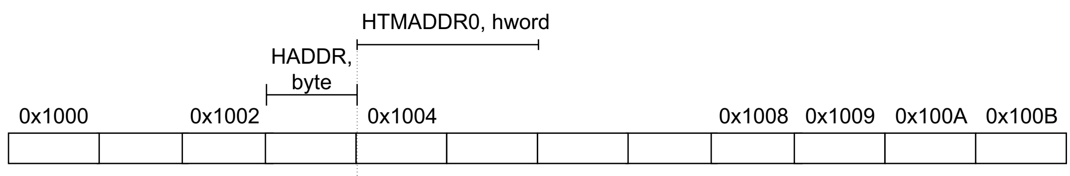


Figure 3-23 Single address non-matching example

Address range matching

For address range comparisons, the address windows include the address or addresses between the two single address comparators and the window size of each single address comparator. So, in Figure 3-24 on page 3-32, the range is `0x1000-0x100D`.

As shown in Figure 3-24, with the following values there is a successful match:

HTMADDR0 value `0x1004`
size hword
HTMADDR1 value `0x100C`
size hword
HADDR value `0x1008`
HSIZE value word

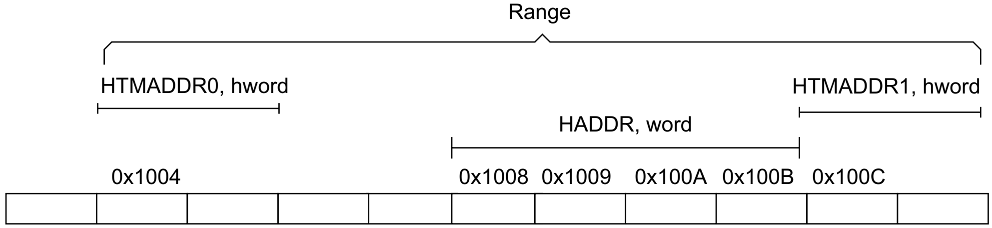


Figure 3-24 Address range comparison successful match example 1

Similarly, there is a successful match with the following values, as shown in Figure 3-25:

HTMADDR0 value `0x1004`
size hword
HTMADDR1 value `0x100C`
size hword
HADDR value `0x100D`
HSIZE value byte

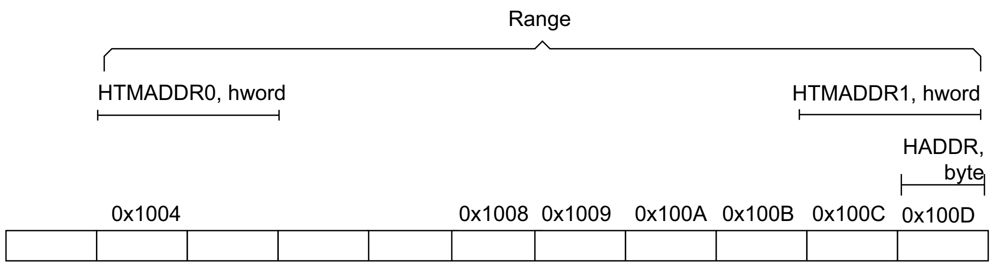


Figure 3-25 Address range comparison successful match example 2

In contrast, with the following values there is no match, as shown in Figure 3-26:

HTMADDR0 value `0x1004`
size hword
HTMADDR1 value `0x100C`
size hword
HADDR value `0x1003`
HSIZE value byte

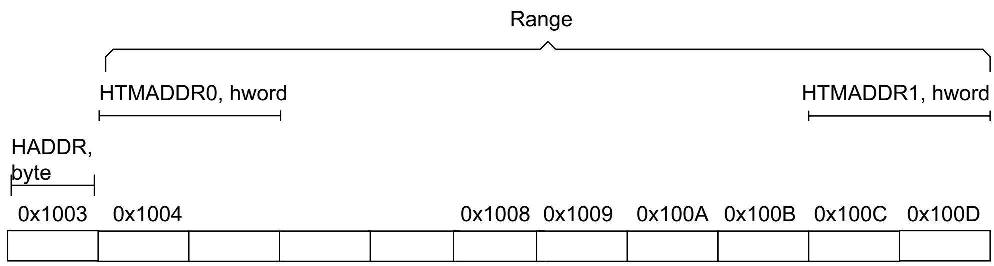


Figure 3-26 Address range comparison unsuccessful match example 1

Similarly, with the following values there is no match, as shown in Figure 3-27:

HTMADDR0 value `0x1004`
size hword
HTMADDR1 value `0x100C`
size hword
HADDR value `0x100E`
HSIZE value hword

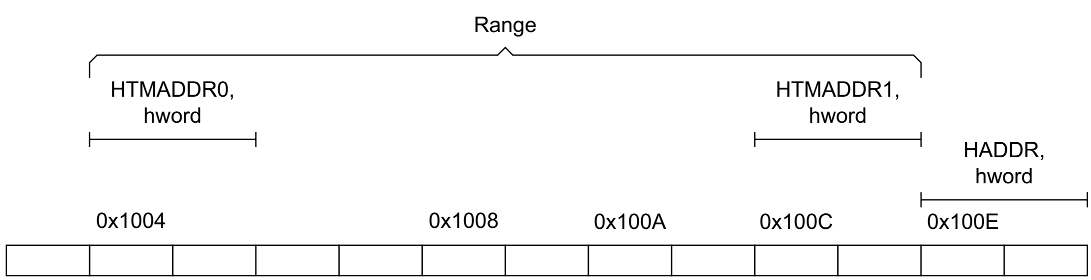


Figure 3-27 Address range comparison unsuccessful match example 2

3.3.21 HTM AMBA Control Select Registers, HTMHCTRLSEL0-7

The HTMHCTRLSEL Registers select the values for AMBA control comparators.

Figure 3-28 shows the bit assignments.

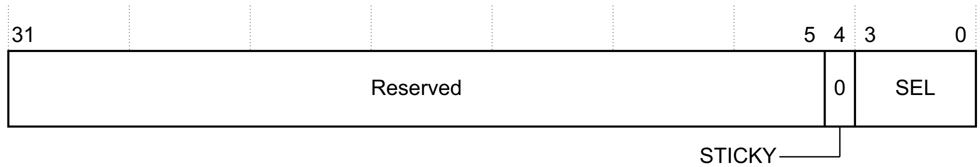


Figure 3-28 HTMHCTRLSEL Register bit assignments

Table 3-29 shows the bit assignments.

Table 3-29 HTMHCTRLSEL Register bit assignments

Bits	Name	Function
[31:5]	Reserved	Reserved. Read as zero, do not modify.
[4]	STICKY	Output mode. 0 = Output only high when **HREADY**=1 1 = Output stays unchanged until next active traced transfer is detected (**HREADY**=1, **HTRANS[1]** =1 and trace is allowed by security settings).
[3:0]	SEL	Select AMBA control signal for compare.

Table 3-30 shows the SEL key to Figure 3-29 on page 3-35.

Table 3-30 SEL values key

Abbreviation	Meaning	Remarks
D2-D0	Data	Lowest three bits of the extracted data on the data bus
HB	**HBSTRB**	-
HBR	**HBurst**	-
HD	HDOMAIN	-
HM	HMASTER	-
HML	**HMASTLOCK**	-

Table 3-30 SEL values key (continued)

Abbreviation	Meaning	Remarks
HP	HPROT	-
HS	HSIZE	-
HT	**HTRANS**	-
HUN	HUNALIGN	-
HW	**HWRITE**	-
Res	Response	See Table 3-12 on page 3-17 for encoding
SC	SELCODE	See Table 3-13 on page 3-17 for encoding
WR[1:0]	Wait range	Wait state range: Bit 0 is set to 1 if the number of wait state is over 15 Bit 1 is set to 1 if the number of wait state is over 31

Figure 3-29 shows the compared signals for the values of SEL.

	Compared signal							
SEL	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0	HS[1]	HS[0]	HP[0]	HD[3]	HD[2]	HD[1]	HD[0]	HW
0x1	HS[1]	HS[0]	HP[0]	HM[3]	HM[2]	HM[1]	HM[0]	HW
0x2	HS[1]	HS[0]	HP[0]	HP[1]	HBR[2]	HBR[1]	HBR[0]	HW
0x3	HB[7]	HB[6]	HB[5]	HB[4]	HB[3]	HB[2]	HB[1]	HB[0]
0x4	HP[1]	HP[5]	HML	HM[3]	HM[2]	HM[1]	HM[0]	HW
0x5	HP[1]	HP[5]	HML	HD[3]	HD[2]	HD[1]	HD[0]	HW
0x6	HP[4]	HP[3]	HP[2]	HP[1]	HBR[2]	HBR[1]	HBR[0]	HW
0x7	HP[4]	HP[3]	HP[2]	HP[1]	HP[6]	HP[5]	HP[0]	HW
0x8	HS[1]	HS[0]	HML	HP[1]	HP[6]	HUN	HP[0]	HW
0x9	HS[1]	HS[0]	HT[1]	HT[0]	HBR[2]	HBR[1]	HBR[0]	HW
0xA	HS[1]	HS[0]	HP[0]	HW	SC[3]	SC[2]	SC[1]	SC[0]
0xB	HP[6]	HP[5]	HP[1]	HW	SC[3]	SC[2]	SC[1]	SC[0]
0xC	HM[3]	HM[2]	HM[1]	HM[0]	SC[3]	SC[2]	SC[1]	SC[0]
0xD	HD[3]	HD[2]	HD[1]	HD[0]	SC[3]	SC[2]	SC[1]	SC[0]
0xE	Res[1]	Res[0]	WR[1]	WR[0]	SC[3]	SC[2]	SC[1]	SC[0]
0xF	Res[1]	Res[0]	HML	HP5	D2	D1	D0	HW

Figure 3-29 SEL values and AMBA control comparators

The various SEL combinations are used as follows:

Combinations `0x0`-`0xD`

The AMBA control comparators can be used for event generations and can work with address comparators (see *HTM Address Type Registers, HTMADDRTYPE0-15* on page 3-28). Eleven possible combinations are provided to enable comparison of multiple signals at the same time.

Combinations `0xE` and `0xF`

The AMBA control comparators can be used for trigger generation when the HTMHCTRLSEL register is set to `0xE` or `0xF`. When these combinations are used, the output result from the AMBA control comparator is not timing-accurate and therefore must not be used for trace filtering, otherwise the traced filtering behavior will be unpredictable. In addition, do not enable the use of these control comparison in address comparator setting. However, these combinations can be used for generating triggers in trace or cross trigger to other trace devices such as ETM:

- Combination `0xE` is useful to detect when a certain slave is selected, or when an error response is generated, or to detect when wait states exceed a certain range of clock cycles.
- Combination `0xF` is useful for debugging semaphore operation.

3.3.22 HTM AMBA Control Compare Value Registers, HTMHCTRLVAL0-7

The HTMHCTRLVAL Registers hold the compare value for AMBA control comparators.

Figure 3-30 shows the bit assignments.

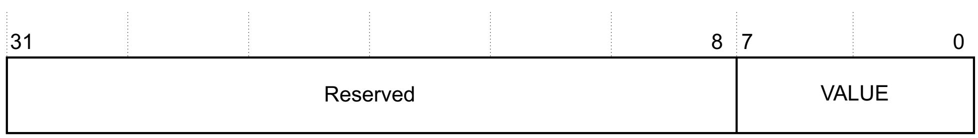


Figure 3-30 HTMHCTRLVAL Register bit assignments

Table 3-31 shows the bit assignments.

Table 3-31 HTMHCTRLVAL Registers bit assignments

Bits	Name	Function
[31:8]	-	Reserved. Read as zero, do not modify.
[7:0]	VALUE	Compare Value.

3.3.23 HTM AMBA Control Compare Mask Registers, HTMHCTRLMASK0-7

The HTMHCTRLMASK Registers hold the compare mask for AMBA control comparators

Figure 3-31 shows the bit assignments.

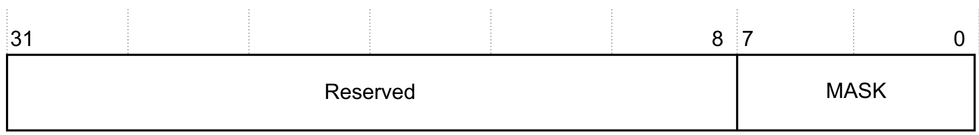


Figure 3-31 HTMHCTRLMASK Register bit assignments

Table 3-32 shows the bit assignments.

Table 3-32 HTMHCTRLMASK Register bit assignments

Bits	Name	Function
[31:8]	-	Reserved. Read as zero, do not modify.
[7:0]	MASK	Compare mask. For each bit: 1 = compare enabled 0 = compare disabled.

3.3.24 HTM Counter Reload Value Registers, HTMCNTRELDVAL0-3

The HTMCNTRELDVAL Registers hold the starting and reload value for counters. The counter automatically loads with this value when this register is programmed. It is then reloaded when the counter reload event occurred, defined by the Counter Reload Event Register.

Figure 3-32 on page 3-38 shows the bit assignments.

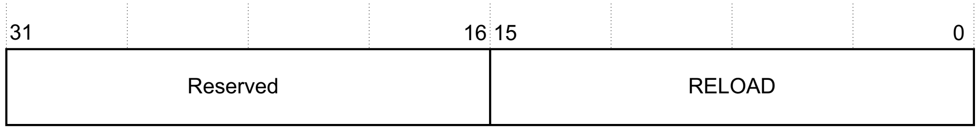


Figure 3-32 HTMCNTRELDVAL Register bit assignments

Table 3-33 shows the bit assignments.

Table 3-33 HTMCNTRELDVAL Register bit assignments

Bits	Name	Function
[31:16]	-	Reserved. Read as zero, do not modify.
[15:0]	RELOAD	Reload Value.

3.3.25 HTM Counter Enable Registers, HTMCNTENABLE0-3

The HTMCNTENABLE registers define the enable event for the counters.

Figure 3-33 shows the bit assignments.

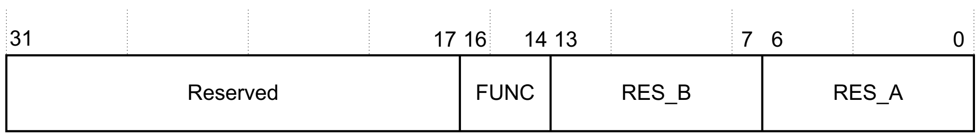


Figure 3-33 HTMCNTENABLE Register bit assignments

Table 3-34 shows the bit assignments.

Table 3-34 HTMCNTENABLE Register bit assignments

Bits	Name	Function
[31:17]	Reserved	Reserved. Read as zero, do not modify.
[16:14]	FUNC	Boolean Function between Resource A and B.
[13:7]	RES_B	Resource B.
[6:0]	RES_A	Resource A.

3.3.26 HTM Counter Reload Event Registers, HTMCNTRELDEVT0-3

The HTMCNTRELDEVT Registers define the reload event for the counters.

Figure 3-34 shows the bit assignments.

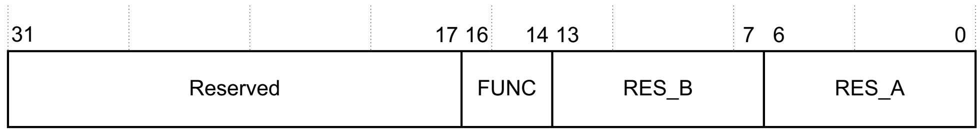


Figure 3-34 HTMCNTRELDEVT Register bit assignments

Table 3-35 shows the bit assignments

Table 3-35 HTMCNTRELDEVT Register bit assignments

Bits	Name	Function
[31:17]	Reserved	Reserved. Read as zero, do not modify.
[16:14]	FUNC	Boolean Function between Resource A and B.
[13:7]	RES_B	Resource B.
[6:0]	RES_A	Resource A.

3.3.27 HTM Counter Value Registers, HTMCNTVALUE0-3

The HTMCNTVALUE Registers hold the current values of the counters. The counter value can be changed by writing to this register.

Figure 3-35 shows the bit assignments.

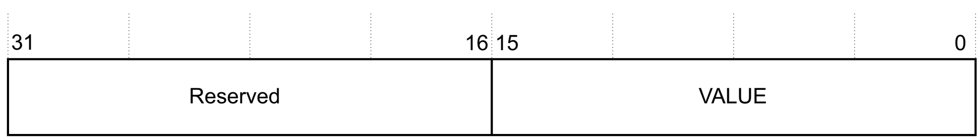


Figure 3-35 HTMCNTVALUE Register bit assignments

Table 3-36 shows the bit assignments.

Table 3-36 HTMCNTVALUE Register bit assignments

Bits	Name	Function
[31:16]	Reserved	Reserved. Read as zero, do not modify.
[15:0]	VALUE	Current counter value.

3.3.28 HTM Sequencer Transition Event Registers, HTMSEQEVT0-5

The HTMSEQEVT Registers determine the transition events for the sequencers.

Table 3-37 defines the HTMSEQEVT0-5 Registers.

Table 3-37 HTMSEQEVT0-5 Register bit assignments

Address	Name	Function
`0x300`	HTMSEQEVT0	State 1-State 2 transition event register
`0x304`	HTMSEQEVT1	State 2-State 1 transition event register
`0x308`	HTMSEQEVT2	State 2-State 3 transition event register
`0x30C`	HTMSEQEVT3	State 3-State 1 transition event register
`0x310`	HTMSEQEVT4	State 3-State 2 transition event register
`0x314`	HTMSEQEVT5	State 1-State 3 transition event register

Figure 3-36 shows the bit assignments.

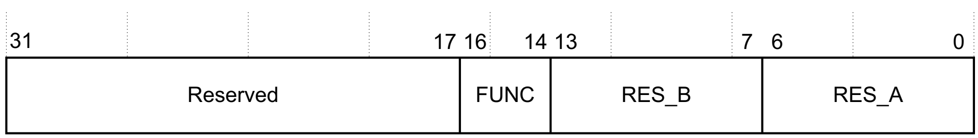


Figure 3-36 HTMSEQEVT Register bit assignments

Table 3-38 shows the bit assignments.

Table 3-38 HTMSEQEVT Register bit assignments

Bits	Name	Function
[31:17]	Reserved	Reserved. Read as zero, do not modify.
[16:14]	FUNC	Boolean Function between Resource A and B.
[13:7]	RES_B	Resource B.
[6:0]	RES_A	Resource A.

3.3.29 HTM Sequencer State Register, HTMSEQSTATE

The HTMSEQSTATE registers holds the current state of the HTM sequencers.

Figure 3-37 shows the bit assignments.

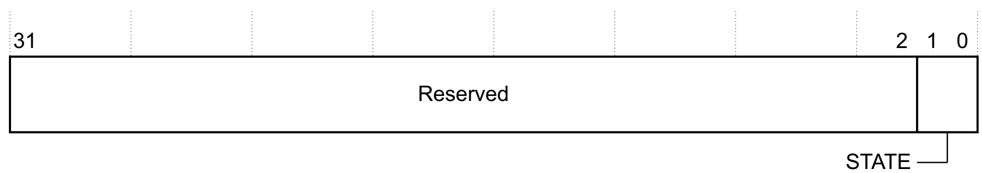


Figure 3-37 HTMSEQSTATE Register bit assignments

Table 3-39 shows the bit assignments.

Table 3-39 HTMSEQSTATE Register bit assignments

Bits	Name	Function
[31:2]	-	Reserved. Read as zero, do not modify.
[1:0]	STATE	Current state: 0 = state 1 1 = state 2 2 = state 3.

3.3.30 HTM External Output Event Registers, HTMEXTOUTEVT0-3

The HTMEXTOUTEVT Registers define the enable event for the external outputs.

Figure 3-38 shows the bit assignments.

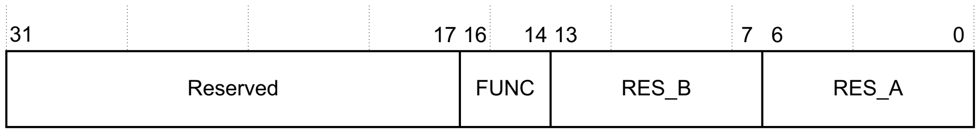


Figure 3-38 HTMEXTOUTEVT Register bit assignments

Table 3-40 shows the bit assignments.

Table 3-40 HTMEXTOUTEVT Register bit assignments

Bits	Name	Function
[31:17]	Reserved	Reserved. Read as zero, do not modify.
[16:14]	FUNC	Boolean Function between Resource A and B.
[13:7]	RES_B	Resource B.
[6:0]	RES_A	Resource A.

3.3.31 HTM ATB ID Register, HTMATIDOUT

The HTMATIDOUT Register determines the ATB ID output value.

Figure 3-39 shows the bit assignments.

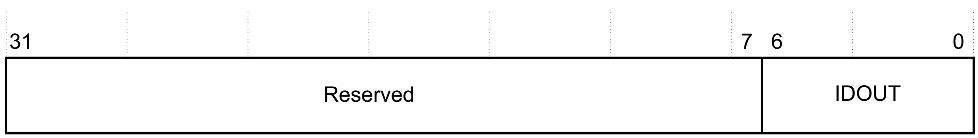


Figure 3-39 HTMATIDOUT Register bit assignments

Table 3-41 shows the bit assignments.

Table 3-41 HTMATIDOUT Register bit assignments

Bits	Name	Function
[31:7]	Reserved	Reserved. Read as zero, do not modify.
[6:0]	IDOUT	ATB ID output value.

3.3.32 HTM Claim Tag Set Register, HTMCLAIMTAGSET

The HTMCLAIMTAGSET Register indicates how many bits are implemented in the claim tag register, and is used for setting the claim tag. The claim tag register is typically used for any interrogating tools to determine if the device is being programmed or has been programmed.

Figure 3-40 shows the bit assignments.

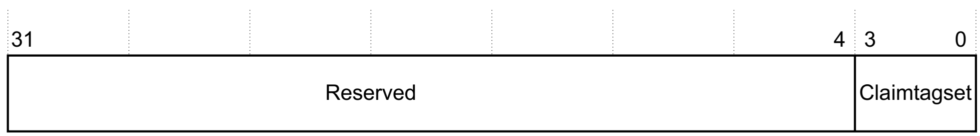


Figure 3-40 HTMCLAIMTAGSET Register bit assignments

Table 3-42 shows the bit assignments.

Table 3-42 HTMCLAIMTAGSET Register bit assignments

Bits	Name	Function
[31:4]	Reserved	Reserved. Read as zero, do not modify.
[3:0]	Claimtagset	Read: 1 = Claim tag bit is implemented 0 = Claim tag bit is not implemented. Write: 1 = Set claim tag bit 0 = No effect.

3.3.33 HTM Claim Tag Clear Register, HTMCLAIMTAGCLR

The HTMCLAIMTAGCLR Register indicates the current status of the claim tag bit, and is used for clearing the claim tag.

Figure 3-41 shows the bit assignments.

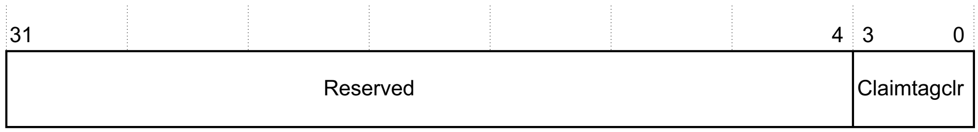


Figure 3-41 HTMCLAIMTAGCLR Register bit assignments

Table 3-43 shows the bit assignments.

Table 3-43 HTMCLAIMTAGCLR Register bit assignments

Bits	Name	Function
[31:4]	Reserved	Reserved. Read as zero, do not modify.
[3:0]	Claimtagclr	Read: Current value of claim tag. Write: 1 = Clear claim tag bit 0 = No effect.

3.3.34 HTM Lock Access Register, HTMLOCK_ACCESS

The HTMLOCK_ACCESS Register locks the write access to registers. After power up the HTM is locked and therefore write access to registers is ignored. To unlock the HTM, the word `0xC5ACCE55` (CoreSight ACCESS) must be written to this register location. After required register accesses is done, it is possible to lock the HTM by writing any number except the key access word to the HTMLOCK_ACCESS register. The lock status is indicated in the HTMSTATUS and HTMLOCK_STATUS Registers.

Note

When **PADDRDBG31** is HIGH (lock bypassed), writes to the HTMLOCK_ACCESS Register are ignored.

Table 3-44 shows the bit assignments.

Table 3-44 HTMLOCK_ACCESS Register bit assignments

Bits	Name	Function
[31:0]	KEY	Write Access Code. The word `0xC5ACCE55` enables further write access to this device.

3.3.35 HTM Lock Status Register, HTMLOCK_STATUS

The HTMLOCK_STATUS Register indicates if the lock feature is implemented, and the current status of the lock. If this register is accessed from upper 2GB of memory (**PADDRDBG31** is HIGH during access), the lock mechanism is bypassed, so both bit 0 and bit 1 are 0. This indicates that the lock mechanism is not implemented and the device is unlocked. Otherwise, bit 0 is 1 and bit 1 depends on the current status of the lock. Bit 2 is low in both cases.

Figure 3-42 shows the bit assignments.

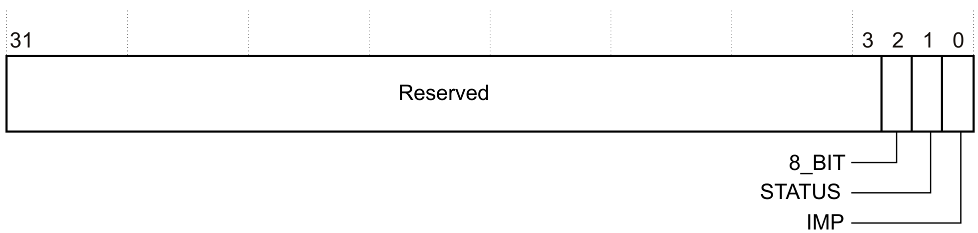


Figure 3-42 HTMLOCK_STATUS Register bit assignments

Table 3-45 shows the bit assignments.

Table 3-45 HTMLOCK_STATUS Register bit assignments

Bits	Name	Function
[31:3]	Reserved	Reserved. Read as zero.

Table 3-45 HTMLOCK_STATUS Register bit assignments (continued)

Bits	Name	Function
[2]	8_BIT	Set to 0 because the HTM implements a 32-bit lock register.
[1]	STATUS	Lock status: 1 = Access has been locked. 0 = Device write access is unlocked.
[0]	IMP	Lock implemented: 1 = Lock mechanism implemented. 0 = Lock mechanism not implemented.

Table 3-46 shows the effect of **PADDRDBG31** on the registers.

Table 3-46 Effect of PADDRDBG31 on registers

Register	Access with PADDRDBG31= 0	Access with PADDRDBG31= 1
Functional registers	Read always allowed. Write depends on lock status.	Read always allowed. Write always allowed.
HTMSTATUS	Read always allowed. Lock status depends on actual lock status.	Read always allowed. Lock status is unlocked.
HTMLOCK_ACCESS	Write always allowed so that it can be unlocked.	Write ignored.
HTMLOCK_STATUS	Read shows lock implemented. Lock status depends on actual lock status.	Read shows lock not implemented. Lock status is unlocked.
HTMCLAIMTAGSET	Read always allowed. Write depends on lock status.	Read always allowed. Write always allowed.
HTMCLAIMTAGCLR	Read always allowed. Write depends on lock status.	Read always allowed. Write always allowed.
Integration registers	Read always allowed. Write depends on lock status.	Read always allowed. Write always allowed.

3.3.36 HTM Authentication Status Register, HTMAUTHSTATUS

The HTMAUTHSTATUS Register reports the current required security levels. Where functionality changes because a security level changes, the values in this register must also change.

Figure 3-43 shows the bit assignments.

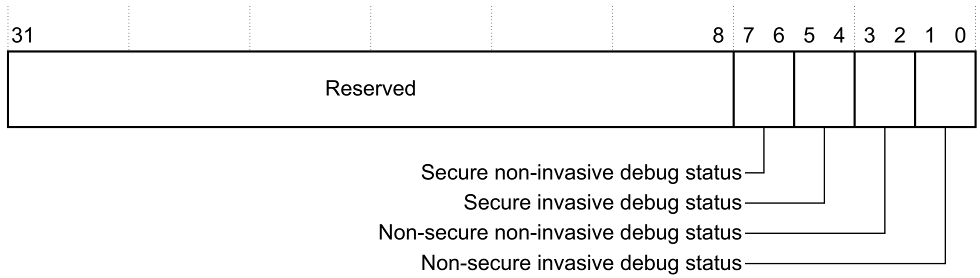


Figure 3-43 HTMAUTHSTATUS Register bit assignments

Table 3-47 shows the bit assignments.

Table 3-47 HTMAUTHSTATUS Register bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved. Read as zero.
[7:6]	Secure non-invasive debug	If ((**SPNIDEN** or **SPIDEN**) and (**NIDEN** or **DBGEN**)) is 1 this field equals 2'b11, indicating the functionality is implemented and enabled. Otherwise, this field equals 2'b10 (implemented but disabled)
[5:4]	Secure invasive debug	Equals 2'b00. This functionality is not implemented.
[3:2]	Non-secure non-invasive debug	If **NIDEN** or **DBGEN** is 1 this field equals 2'b11, indicating the functionality is implemented and enabled. Otherwise, this field equals 2'b10 (implemented but disabled)
[1:0]	Non-secure invasive debug	Equals 2'b00. This functionality is not implemented.

3.3.37 HTM Device CoreSight ID Register, HTMDEVID

The HTMDEVID register defines the CoreSight ID.

Table 3-48 shows the bit assignments.

Table 3-48 HTMDEVID Register bit assignments

Bits	Name	Function
[31:0]	Reserved	CoreSight Device identification

3.3.38 HTM ATB Device Type Register, HTMDEV_TYPE

The HTMDEV_TYPE Register indicates the type of device in terms of CoreSight class.

Figure 3-44 shows the bit assignments.

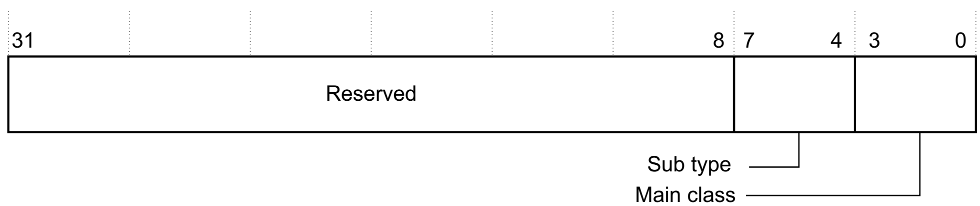


Figure 3-44 HTMDEV_TYPE Register bit assignments

Table 3-49 shows the bit assignments.

Table 3-49 HTMDEV_TYPE Register bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved. Read as zero.
[7:4]	Sub type	Equals `0x4`. Bus trace.
[3:0]	Main class	Equals `0x3`. Trace source.

3.4 Peripheral Identification Registers, HTMPERIPHID0-7

The HTMPERIPHID0-7 Registers are eight 8-bit registers, that span address locations `0xFD0-0xFEC`. The read-only registers provide the following options of the peripheral:

- part number
- designer (JEP106 code)
- revision information
- customer modified field
- memory footprint size.

See *Identification fields* on page 3-53 for more information.

The peripheral identification registers are described in the following sections:

- *HTM Peripheral ID0 Register, HTMPERIPHID0*
- *HTM Peripheral ID1 Register, HTMPERIPHID1* on page 3-50
- *HTM Peripheral ID2 Register, HTMPERIPHID2* on page 3-51
- *HTM Peripheral ID3 Register, HTMPERIPHID3* on page 3-51
- *HTM Peripheral ID4 Register, HTMPERIPHID4* on page 3-52
- *HTM Peripheral ID5 Register, HTMPERIPHID5* on page 3-52
- *HTM Peripheral ID6 Register, HTMPERIPHID6* on page 3-53
- *HTM Peripheral ID7 Register, HTMPERIPHID7* on page 3-53.

3.4.1 HTM Peripheral ID0 Register, HTMPERIPHID0

The HTMPERIPHID0 Register is hard-coded and the fields in the register determine the reset value.

Figure 3-45 shows the bit assignments.

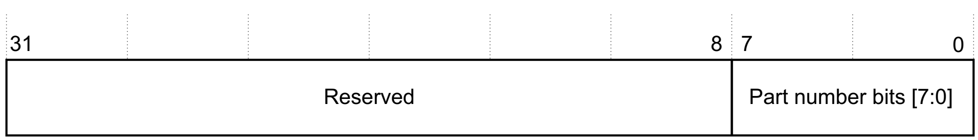


Figure 3-45 HTMPERIPHID0 Register bit assignments

Table 3-50 shows the bit assignments.

Table 3-50 HTMPERIPHID0 Register bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved. Read as zero.
[7:0]	Part number bits [7:0]	These bits read back as `0x17`.

3.4.2 HTM Peripheral ID1 Register, HTMPERIPHID1

The HTMPERIPHID1 Register is hard-coded and the fields in the register determine the reset value.

Figure 3-46 shows the bit assignments.

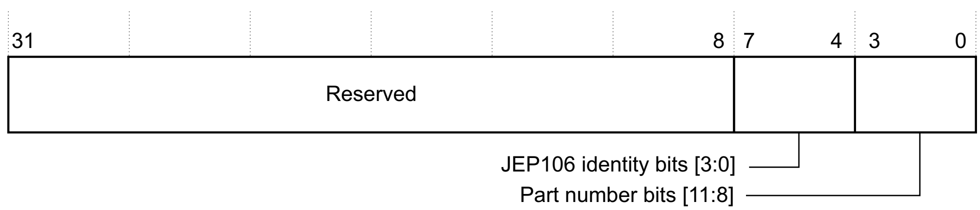


Figure 3-46 HTMPERIPHID1 Register bit assignments

Table 3-51 shows the bit assignments.

Table 3-51 HTMPERIPHID1 Register bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved. Read as zero.
[7:4]	JEP106 identity bits [3:0]	These bits read back as `0xB`.
[3:0]	Part number bits [11:8]	These bits read back as `0x9`.

3.4.3 HTM Peripheral ID2 Register, HTMPERIPHID2

The HTMPERIPHID2 Register is hard-coded and the fields in the register determine the reset value.

Figure 3-47 shows the bit assignments.

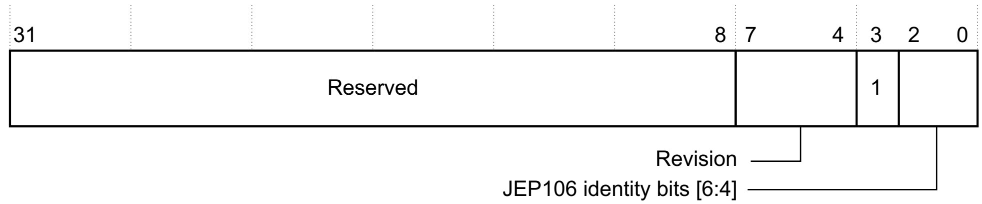


Figure 3-47 HTMPERIPHID2 Register bit assignments

Table 3-52 shows the bit assignments.

Table 3-52 HTMPERIPHID2 Register bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved. Read as zero.
[7:4]	Revision	These bits read back as `0x4`.
[3]	-	Always `0x1`, indicating JEP106 value is used.
[2:0]	JEP106 identity [6:4]	These bits read back as `0x3`.

3.4.4 HTM Peripheral ID3 Register, HTMPERIPHID3

The HTMPERIPHID3 Register is hard-coded and the fields in the register determine the reset value.

Figure 3-48 shows the bit assignments.

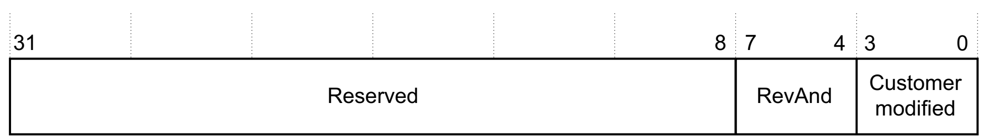


Figure 3-48 HTMPERIPHID3 Register bit assignments

Table 3-53 shows the bit assignments.

Table 3-53 HTMPERIPHID3 Register bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved. Read as zero.
[7:4]	RevAnd	These bits read back as `0x0`.
[3:0]	Customer modified	These bits read back as `0x0`.

3.4.5 HTM Peripheral ID4 Register, HTMPERIPHID4

The HTMPERIPHID4 Register is hard-coded and the fields in the register determine the reset value.

Figure 3-49 shows the bit assignments.

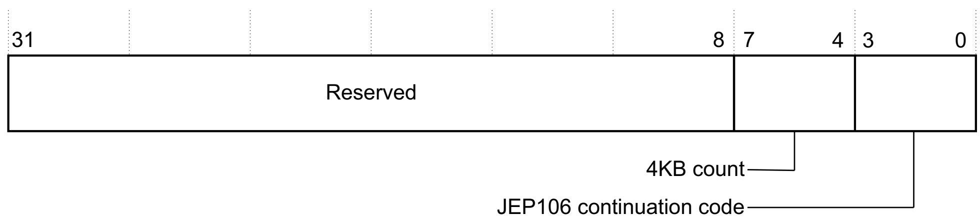


Figure 3-49 HTMPERIPHID4 Register bit assignments

Table 3-54 shows the bit assignments.

Table 3-54 HTMPERIPHID4 Register bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved. Read as zero.
[7:4]	4KB count	These bits read back as `0x0`.
[3:0]	JEP106 continuation code	These bits read back as `0x4`.

3.4.6 HTM Peripheral ID5 Register, HTMPERIPHID5

The HTMPERIPHID5 Register is unused and is reserved for future use.

3.4.7 HTM Peripheral ID6 Register, HTMPERIPHID6

The HTMPERIPHID6 Register is unused and is reserved for future use.

3.4.8 HTM Peripheral ID7 Register, HTMPERIPHID7

The HTMPERIPHID7 Register is unused and is reserved for future use.

3.4.9 Identification fields

The identification fields are used as follows:

Part number

This is selected by the designer of the component and must not conflict with any previously created components from that designer, unless this is an update to a component.

JEP106 This indicates the designer of the component and not the implementer, except where the two are the same. JEDEC is an organization that maintains a document that lists manufacturer codes (JEP106).

ARM has been allocated the 59th slot in bank 5, with `0x3B` being the JEDEC defined 8-bit representation. This appears as a JEP106 code of `0x7F7F7F7F3B`. This is represented as:

- Peripheral ID4[3:0] is `0x4`, indicating the 5th bank
- Peripheral ID2[2:0] is `0x3`, bits [6:4] of `0x3B`
- Peripheral ID1[7:4] is `0xB`, bits [3:0] of `0x3B`.

Revision The Revision field is an incremental value starting at `0x0` for the first design of this component. This only increases by 1 for both major and minor revisions and is a look-up to establish the exact major or minor revision.

Customer modified

This value must indicate whether the customer has modified the component from the delivered RTL. If no RTL modifications are allowed then this field is always zero. The HTM has a value of zero.

RevAnd The HTM uses the RevAnd bits only for the Tie-off RevAnd information. Modifications to this field reflect any minor changes, and not an RTL change. The HTM has a value of zero.

KByte count

This is a 4-bit value that indicates the total contiguous size of the memory window used by this device in powers of 2 from the standard 4KB. The HTM has a value of zero, indicating that its memory size is 4KB. For more explanation on the value of this field see the CoreSight Architecture Specification.

3.5 Identification Registers, HTMPCOMPONID0-3

HTMPCOMPONID0-3 are read-only registers. The HTMPCOMPONID0-3 Registers are four 8-bit registers that span address locations `0xFF0-0xFFC`. These read-only registers can conceptually be treated as a single 32-bit register. The register is used as a standard cross-peripheral identification system.

The identification registers are described in the following sections:

- *HTM Component ID0 Register, HTMCOMPONID0*
- *HTM Component ID1 Register, HTMCOMPONID1*
- *HTM Component ID2 Register, HTMCOMPONID2* on page 3-56
- *HTM Component ID3 Register, HTMCOMPONID3* on page 3-56.

3.5.1 HTM Component ID0 Register, HTMCOMPONID0

The HTMCOMPONID0 register is hard-coded and the fields in the register determine the reset value. This register can be accessed with three wait states.

Table 3-55 shows the bit assignments.

Table 3-55 HTMCOMPONID0 Register bit assignments

Bits	Name	Description
[31:8]	Reserved	Reserved. Read as zero.
[7:0]	HTMPCOMPONID0	These bits read back as `0x0D`.

3.5.2 HTM Component ID1 Register, HTMCOMPONID1

The HTMCOMPONID1 register is hard-coded and the fields in the register determine the reset value. This register can be accessed with three wait states.

Table 3-56 shows the bit assignments.

Table 3-56 HTMCOMPONID1 Register bit assignments

Bits	Name	Description
[31:8]	Reserved	Reserved. Read as zero.
[7:0]	HTMPCOMPONID1	Bits [7:4] read back as `0x9`. Bits [3:0] read back as `0x0`.

3.5.3 HTM Component ID2 Register, HTMCOMPONID2

The HTMCOMPONID2 register is hard-coded and the fields in the register determine the reset value. This register can be accessed with three wait states.

Table 3-57 shows the bit assignments.

Table 3-57 HTMCOMPONID2 Register bit assignments

Bits	Name	Description
[31:8]	Reserved	Reserved. Read as zero.
[7:0]	HTMPCOMPONID2	These bits read back as `0x05`.

3.5.4 HTM Component ID3 Register, HTMCOMPONID3

The HTMCOMPONID3 register is hard-coded and the fields in the register determine the reset value. This register can be accessed with three wait states.

Table 3-58 shows the bit assignments.

Table 3-58 HTMCOMPONID3 Register bit assignments

Bits	Name	Description
[31:8]	Reserved	Reserved. Read as zero.
[7:0]	HTMPCOMPONID3	These bits read back as `0xB1`.

Chapter 4
Protocol Details

This chapter describes the AMBA *AHB Trace Macrocell* (HTM) data packet protocol. It contains the following sections:

4.1 ATB interface outputs

The trace results are output using the 32-bit AMBA Trace Bus (ATB) interface. The information is packet based and byte oriented. An AHB transfer is typically represented by an address packet, a data packet and an auxiliary packet. The generation of data packets and auxiliary packets is optional, depending on the setting in the HTMCONTROL Register. Auxiliary packets are use to hold AMBA control information.

For AHB burst transfers, only the first transfer has the address packet and auxiliary packet. The rest of the burst only has data packets because the address can be calculated from the start of the burst, and the AMBA control information in the auxiliary packet does not change.

In addition to the address, data, and auxiliary packets, there are a number of control packets. They include:

- Trigger
- Data suppressed
- FIFO Overflow
- CycleCount
- AHB reset
- TraceOff.

4.2 Trace bandwidth reduction

The trace is compressed using the following techniques:

- Address information is compressed by transmitting only the *Least Significant Bits* (LSB) that are different from the previous address.
- Data packets are compressed by removing some of the leading zeros.
- Auxiliary packets are compressed by transmitting only bytes that are changed.

4.2.1 Data suppression

In normal mode, when the FIFO is almost full, that is, the available free space is less than the level specified in the FIFO Level Register, the data packet and auxiliary packets are not sent. The data suppressed packet is sent instead. If a data suppressed packet takes place during a burst, the rest of the data packet is not transmitted even if the FIFO has enough space to do so.

4.2.2 FIFO overflow

In normal mode, a packet is not stored into the FIFO if there is not enough free space. Although existing data in the FIFO is not lost, it results in lost trace. A FIFO overflow packet is used to indicate this so that the debug tool is able to determine that trace loss has occurred.

Normally during a burst transfer only the first address is sent. The rest of the burst transfers only produce data packets. If a loss of trace takes place during a burst, the rest of the data packet is not transmitted even if the FIFO has enough space to do so.

4.3 ATB packet format

With the exception of data packet and A-sync packet, all packets have a Cont bit in their *Most Significant Bit* (MSB) location. If this bit is '1', then it means another byte follows. If this bit is '0', it indicates the end of the packet. See Figure 4-1.

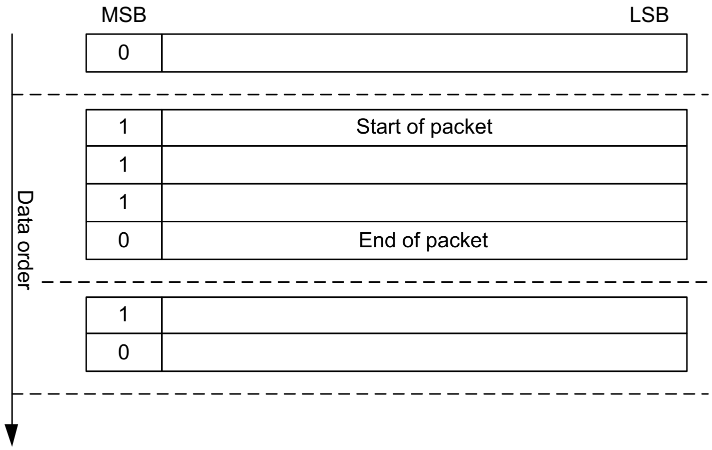


Figure 4-1 HTM packets

For data packets, payload information is included in the first byte of the packet so that the decompression engine can tell where the next header starts, as shown in Figure 4-2.

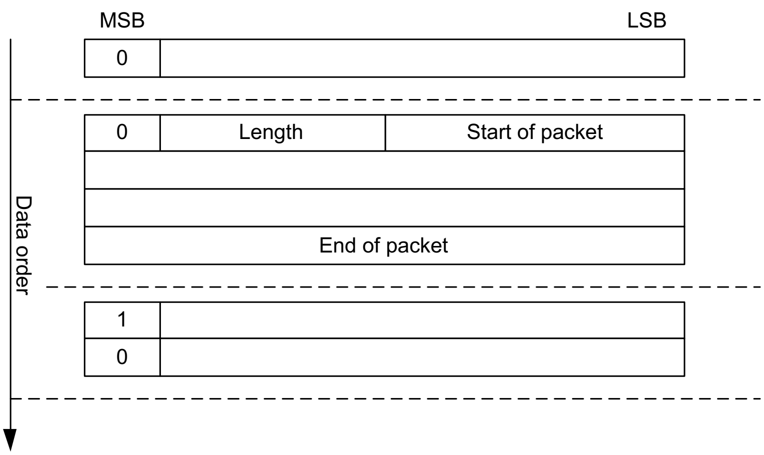


Figure 4-2 HTM data packet length

Data length is encoded using 3 bits as shown in Table 4-1

Table 4-1 HTM data length encodings

Length encoding	Number of bytes
000	0 byte
001	1 byte
010	2 bytes
011	4 bytes
100	6 bytes
101	8 bytes
110	Reserved
111	Reserved

4.3.1 Header encoding summary

Table 4-2 lists the HTM header encodings.

Table 4-2 HTM header encodings summary

Type	Bit 7	Bits 6-5	Bits 4-3	Bit 2	Bit 1	Bit 0
A-sync	0	00	000		0	0
Trigger	0	01	00	0	0	0
SEQ addr	0	11	00	0	0	0
Ignore	0	00	01	0	0	0
Trace off	0	01	01	0	0	0
Data suppressed	0	10	01	0	0	0
FIFO overflow	0	11	01	0	0	0
AHB reset on	0	00	10	0	0	0
AHB reset off	0	01	10	0	0	0
Reserved	X	10/11	10	0	0	0
Reserved	X	XX	11	0	0	0

Table 4-2 HTM header encodings summary (continued)

Type	Bit 7	Bits 6-5	Bits 4-3	Bit 2	Bit 1	Bit 0
Cycle count	Cont	Count[3:0]		1	0	0
Data	0	Length[2:0]	Resp[1:0]		1	0
Address	Cont	HADDR[3:0]		HWr	0	1
Auxiliary	Cont	HCTRL[4:0]			1	1

4.3.2 A-sync packet

The HTM regularly outputs an A-sync packet to enable the decompressor to find the start of the header. The A-sync sequence is eight `0x00`s followed by a `0x80` as shown in Figure 4-3. This pattern is unique, so the decompressor can determine a header location when this pattern is found.

0x00
0x00
0x00
0x00
0x00
0x00
0x00
0x00
0x80

Figure 4-3 HTM A-sync packet

4.3.3 Trigger packet

The trigger packet is inserted when the trigger event takes place. Figure 4-4 shows the bit values of the HTM trigger packet.

0	0	1	0	0	0	0	0

Figure 4-4 HTM trigger packet

4.3.4 Sequential address packet

This is used when a new sequential access is carried out but the data packet is disabled. Figure 4-5 shows the bit values of the sequential address packet.

0	1	1	0	0	0	0	0

Figure 4-5 HTM sequential address packet

4.3.5 Ignore packet

Ignore packets are not generated in normal situations but the format is defined to enable the data to be transferred on media without handshaking. Figure 4-6 shows the values of the HTM ignore packet.

0	0	0	0	1	0	0	0

Figure 4-6 HTM ignore packet

4.3.6 Trace off packet

This packet is used when the trace is turned off by the TraceStart/Stop control in TraceEnable generation, or when the trace operation is stopped by setting PROG bit in the HTMCONTROL Register. Figure 4-7 shows the values of the HTM trace off packet.

0	0	1	0	1	0	0	0

Figure 4-7 HTM trace off packet

4.3.7 Data suppressed packet

The data suppressed packet is output if a data/auxiliary packet is dropped because the FIFO reached the level specified by the HTMFIFOLEVEL Register. When a data suppressed packet is issued, if it is during a burst, the data for the rest of the burst is suppressed. The Data Suppress mode is clear if a data packet or auxiliary packet is output by the HTM. Figure 4-8 shows the values of the data suppressed packet.

0	1	0	0	1	0	0	0

Figure 4-8 HTM data suppressed packet

4.3.8 FIFO overflow packet

In normal mode, a packet is not stored in the FIFO if there is not enough free space in it. Although existing data in the FIFO is not lost, it results in lost trace. A FIFO overflow packet is used to indicate this situation so that the debug tool can determine when a loss of trace has taken place. Figure 4-9 shows the values of the HTM FIFO overflow packet.

0	1	1	0	1	0	0	0

Figure 4-9 HTM FIFO overflow packet

Normally, during a burst transfer, only the first address is sent. The rest of the burst transfers only produce data packets. If a lost trace takes place during a burst, the rest of the data packet is not transmitted even if the FIFO has enough space to do so.

4.3.9 AHB reset packets

An AHB reset on packet is generated when the **HRESETn** signal is asserted while the HTM is active. Figure 4-10 shows the values of the AHB reset on packet.

0	0	0	1	0	0	0	0

Figure 4-10 AHB reset on packet

An AHB reset off packet is generated when the **HRESETn** signal is deasserted while the HTM is active. Figure 4-11 shows the values of the AHB reset off packet.

0	0	1	1	0	0	0	0

Figure 4-11 AHB reset off packet

4.3.10 CycleCount packet

The CycleCount packet is used to enable timing reconstruction in the trace. When CycleCount packet generation is enabled, this is output before a new trace packet when there has been a gap in the trace, for example, wait states or trace filtering. The HTM stops outputting the Count value when the rest of the Count values are the same as the previous values. For example, if the count value is less than 16, only one byte is output. Figure 4-12 on page 4-9 shows the values of the CycleCount packet.

Cont	Count[3:0]	1	0	0
Cont	Count[10:4]			
Cont	Count[17:11]			
Cont	Count[24:18]			
0	Count[31:25]			

Figure 4-12 HTM CycleCount packet

4.3.11 Data packet

Figure 4-13 shows the bit values of the data packet. The MSB is reserved and reads as zero.

Reserved	Length[2:0]	Resp[1:0]	1	0
Data 0 (bits [7:0])				
Data 1 (bits [15:8])				
Data 2 (bits [23:16])				
Data 3 (bits [31:24])				
Data 4 (bits [39:32])				
Data 5 (bits [47:40])				
Data 6 (bits [55:48])				
Data 7 (bits [63:56])				

Figure 4-13 HTM data packet bit values

Table 4-3 shows how the Resp field is encoded.

Table 4-3 Resp encodings

Resp encoding	Meaning
00	Okay
01	Error
10	Exclusive failed
11	Split/Retry

4.3.12 Address packet

The total number of bytes in the address packet depends on how many MSBs of the new address are the same as the previous traced address. After all the address bits that are different from previous address are sent, the address packet terminates. Figure 4-14 shows the bit values of the address packet.

Cont	HADDR[3:0]				HWr	0	1
Cont	HADDR[8:4]					HSIZE[1:0]	
Cont	HADDR[12:9]				HBURST[2:0]		
Cont	HADDR[19:13]						
Cont	HADDR[26:20]						
0	0	HSIZE[2]	HADDR[31:27]				

Figure 4-14 HTM address packet bit values

4.3.13 Auxiliary packet

The **HCTRL** value is determined by the HTMAUXSEL Register. In the data protocol there is no limit on the maximum number of bytes, but the HTM hardware limits it to two bytes. Figure 4-15 shows the auxiliary packet format.

Cont	HCTRL[4:0]					1	1
0	HCTRL[11:5]						

Figure 4-15 HTM auxiliary packet

The control signals are only sampled when **HREADY** is HIGH. If the values of the control signals have changed when **HREADY** is LOW, the old value is ignored.

When the generation of address packets and data packets is disabled, and if auxiliary packets generation is enabled, the auxiliary packet is output for every traced transfer. Otherwise the auxiliary is output only when the control information changes.

The contents of the auxiliary packet are determined by the HTMAUXSEL register described in *HTM AUX Select Register, HTMAUXSEL* on page 3-16.

4.4 Data extraction

To reduce bandwidth usage, only valid data is captured from the data bus. For example, the HTM captures only 2 data bytes in a halfword transfer.

For unaligned transfers, the data bytes are masked with their corresponding byte lane strobe signal, **HBSTRB**.

Table 4-4 shows HTM data extraction when **HUNALIGN** is LOW and **BUSWIDTH** is HIGH.

Table 4-4 HTM data extraction with HUNALIGN 0 and BUSWIDTH 1

BUSENDIAN	HSIZE	HADDR[2:0]	HBSTRB	Captured data
1 (BE-32)	DWORD	3'b000	Don't care	Bits[63:0]
1	WORD	3'b000	Don't care	Bits[31:0]
1	WORD	3'b100	Don't care	Bits[63:32]
1	HWORD	3'b000	Don't care	Bits[31:16]
1	HWORD	3'b010	Don't care	Bits[15:0]
1	HWORD	3'b100	Don't care	Bits[63:48]
1	HWORD	3'b110	Don't care	Bits[47:32]
1	BYTE	3'b000	Don't care	Bits[31:24]
1	BYTE	3'b001	Don't care	Bits[23:16]
1	BYTE	3'b010	Don't care	Bits[15:8]
1	BYTE	3'b011	Don't care	Bits[7:0]
1	BYTE	3'b100	Don't care	Bits[63:56]
1	BYTE	3'b101	Don't care	Bits[55:48]
1	BYTE	3'b110	Don't care	Bits[47:40]
1	BYTE	3'b111	Don't care	Bits[39:32]

Table 4-5 shows HTM data extraction when **HUNALIGN** is LOW and **BUSWIDTH** is LOW.

Table 4-5 HTM data extraction with HUNALIGN 0 and BUSWIDTH 0

BUSENDIAN	HSIZE	HADDR[2:0]	HBSTRB	Captured data
1	WORD	3'bx00	Don't care	Bits[31:0]
1	HWORD	3'bx00	Don't care	Bits[31:16]
1	HWORD	3'bx10	Don't care	Bits[0:15]
1	BYTE	3'bx00	Don't care	Bits[31:24]
1	BYTE	3'bx01	Don't care	Bits[23:16]
1	BYTE	3'bx10	Don't care	Bits[15:8]
1	BYTE	3'bx11	Don't care	Bits[7:0]

Table 4-6 shows HTM data extraction values when **HUNALIGN** is HIGH and **BUSWIDTH** is HIGH.

Table 4-6 HTM data extraction with HUNALIGN 1 and BUSWIDTH 1

BUSENDIAN	HSIZE	HADDR[2:0]	HBSTRB	Captured data
Don't care	DWORD	Don't care	Don't care	Bits[63:0] masked with **HBSTRB**
Don't care	WORD	Don't care	8'b0000xxxx	Bits[31:0] masked with **HBSTRB**
Don't care	WORD	Don't care	8'bxxxx0000	Bits[63:32] masked with **HBSTRB**
Don't care	HWORD	Don't care	8'b000000xx	Bits[15:0] masked with **HBSTRB**
Don't care	HWORD	Don't care	8'b0000xx00	Bits[31:16] masked with **HBSTRB**
Don't care	HWORD	Don't care	8'b00xx0000	Bits[47:32] masked with **HBSTRB**
Don't care	HWORD	Don't care	8'bxx000000	Bits[63:48] masked with **HBSTRB**

Table 4-7 on page 4-13 shows HTM data extraction values when **HUNALIGN** is HIGH and **BUSWIDTH** is LOW.

Table 4-7 Data extraction with HUNALIGN 1 and BUSWIDTH 0

BUSENDIAN	HSIZE	HADDR[2:0]	HBSTRB	Captured data
Don't care	WORD	Don't care	8'b0000xxxx	Bits[31:0] masked with **HBSTRB**
Don't care	HWORD	Don't care	8'b000000xx	Bits[15:0] masked with **HBSTRB**
Don't care	HWORD	Don't care	8'b0000xx00	Bits[31:16] masked with **HBSTRB**

4.5 HTM trace data output rules

The trace output data consists of different packets. A transfer is typically indicated by three packets, an address packet followed by an auxiliary and then a data packet or data packets. All three packets can be enabled or disabled.

Only active transfers, non-sequential and sequential, are traced. Idle and busy cycles are not traced.

For burst transfers, the address packet is output for the first transfer. Only data packets are output for the rest of the burst. This is because the address can be calculated and the other AMBA control signals do not change.

If there is an excluded address in the burst, as specified by the EXCLUDE field in the HTMCTRL2 Register or the RANGE_EXC field in the HTMTRACCTRL Register, the address packet is sent when the trace resumes. The auxiliary packet is output only when the information is changed except in profiling modes (auxiliary packet enabled, but address packed and data packet disabled). In profiling mode the auxiliary packet is output for every traced transfer with a minimum of 1 byte.

If address packet generation is enabled and data packet is disabled, sequential transfers in the burst are indicated by sequential address packets.

If both address and data packets are disabled and auxiliary packet generation is enabled, an auxiliary packet is output for every traced AHB transfer even if the control information is the same as in previous transfer. This is particularly useful with the bus profiling function (auxiliary packet enabled, but address packed and data packet disabled). Data compression is still used to reduce the bandwidth by shortening auxiliary packets if possible.

When the TraceOn/Off resource is used, a TraceOff address match generates a TraceOff packet. In addition, when the PROG bit is asserted and when the global enable, **GLBEN**, is HIGH, the TraceOff packet is output.

During generation of data or auxiliary packets, if the remaining space in the FIFO is less than the specified data suppress trigger level, the data suppression packet is sent. If this happens during a burst transfer, the remaining burst data is not sent.

During generation of address packets or cycle count packets, if the remaining space in the FIFO is less than the required size for the packet, the data overflow packet is sent.

When a data suppressed packet is output, it does not have to be sent again during the next AHB transfer. The status remains valid until another auxiliary packet or data packet is sent.

When a FIFO overflow packet is sent, it does not have to be sent again during the next AHB transfer. The status remains valid until another packet (of any kind) is sent.

A data suppressed packet does not have to be sent if the overflow status is valid.

A trigger packet is sent at the rising edge of the triggering event. If an address comparator is used for trigger generation and multiple triggers are allowed, the address comparator must not operate in sticky mode because two successive transfers might be handled as one trigger event.

Data suppression can only be caused by generation of data packets and auxiliary packets. Data suppression status only suppresses data and auxiliary packets.

4.6 HTM packet generation

The HTM design is pipelined so address and data packets are delayed from the AHB activities. In addition, there is a one-cycle delay at the input capturing stage. Address packets, auxiliary packets and data packets are generated when the captured **HREADY** is high. Figure 4-16 shows typical operations over time and the HTM trace packets generated.

Note

The one cycle delay for the capture stage is not shown in the diagrams in this section.

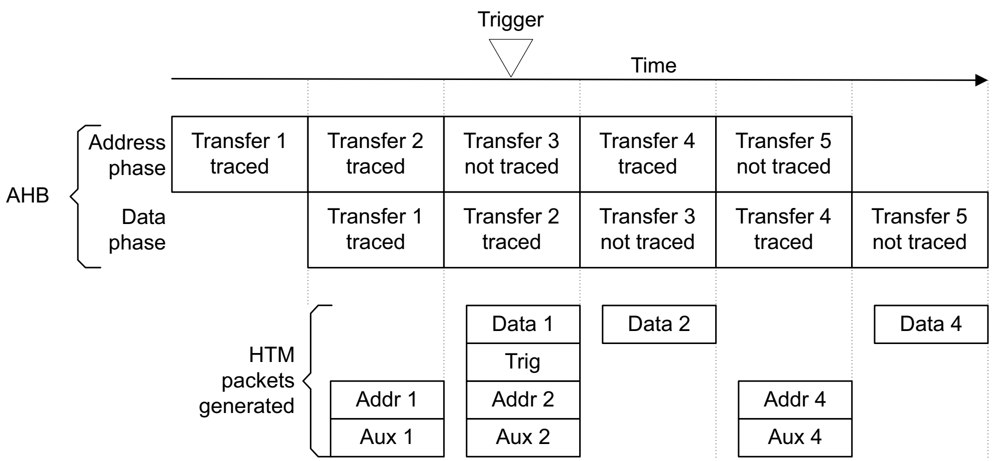


Figure 4-16 HTM typical operations and packets

The data packet is delayed because of the pipelined nature of AHB. Data packets and their corresponding address packets are rearranged at the FIFO input so that the address packets and data packets can be correlated as shown in Figure 4-17.

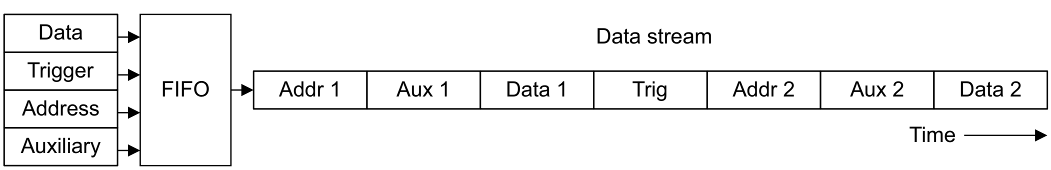


Figure 4-17 HTM packet rearrangement

4.6.1 HTM typical burst transfer

Figure 4-18 shows which HTM packets are generated for burst operations. Only data packets are output for the rest of the burst. This is because the address can be calculated and the other AMBA control signals do not change.

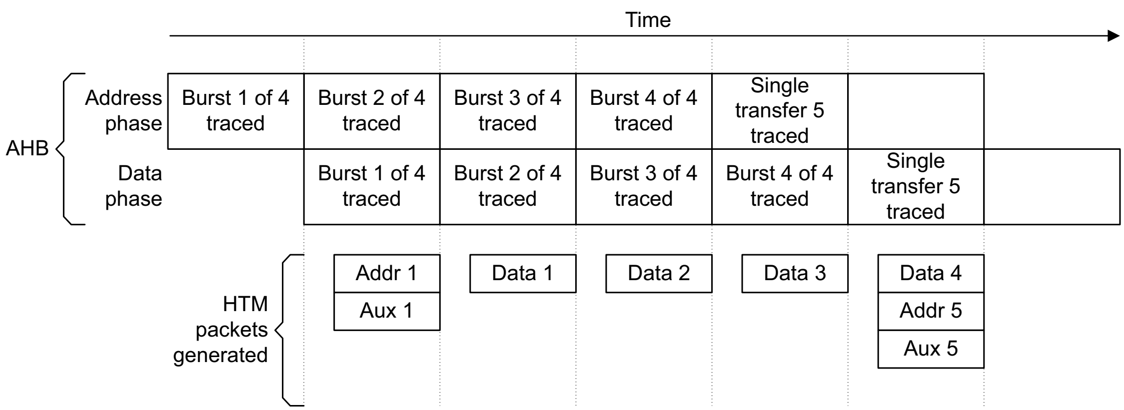


Figure 4-18 HTM typical burst operations and packets generated

4.6.2 HTM typical burst but data is not traced

Figure 4-19 shows tracing of a burst transfer without tracing the data. Instead of using data packets, SEQ packets are used to indicate that a sequential transfer has taken place.

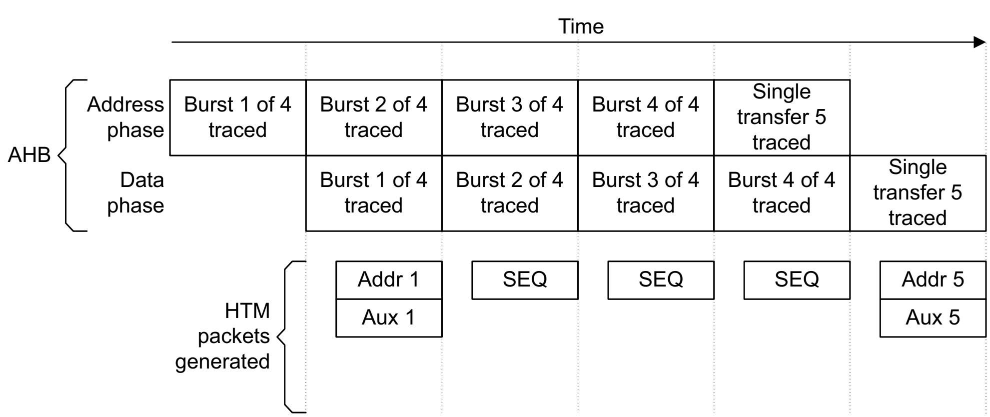


Figure 4-19 HTM typical burst operations where data is not traced

4.6.3 HTM address excluded during burst

If an address is excluded during a burst, the next traced transfer produces an address packet instead of a sequential packet. The Aux 4 packet in Figure 4-20 is not sent because the control information is the same as in the previous packet, Aux 1.

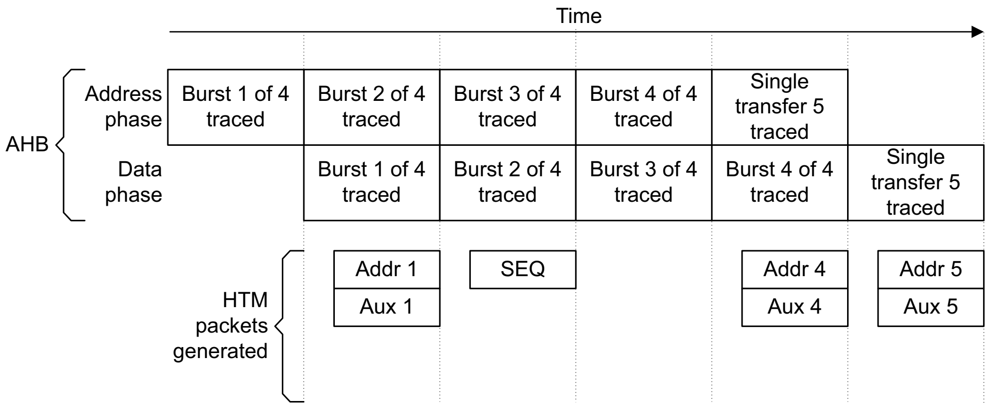


Figure 4-20 HTM address packet excluded during burst

4.6.4 Address wrap handling

Wrap bursts are handled in the same way as incrementing bursts. The decompressor can work out the wrap address from **HSIZE** and **HBURST** in the address packet so it is not necessary to send details of the wrapped address. For example, Addr4 is replaced with SEQ in Figure 4-21 on page 4-19. However, if one of the transfers is excluded, the next traced transfer must have its address packet sent. This is the case with Addr3 in Figure 4-21 on page 4-19. Aux 3 in Figure 4-21 on page 4-19 is not sent because the control information is unchanged.

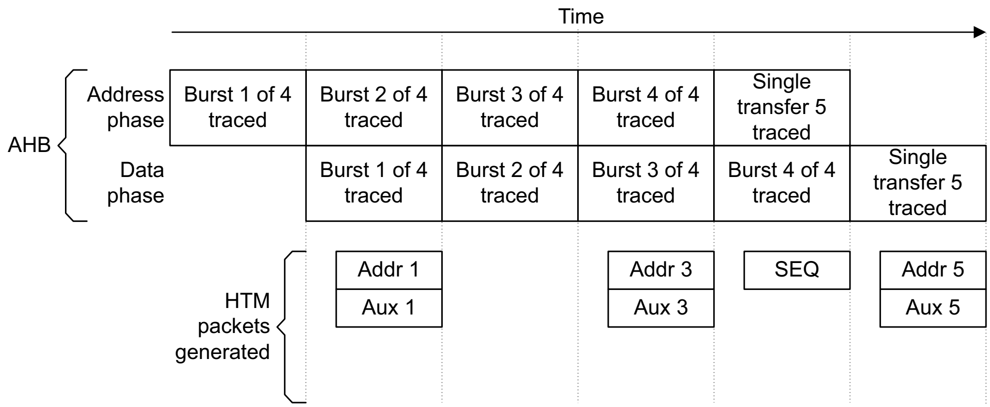


Figure 4-21 HTM wrap burst with address excluded

4.6.5 Data suppressed

When the space left in the FIFO is less than the value programmed for the data suppress triggering level, the data packet and the auxiliary packets are suppressed. A data suppress packet is used to indicate the occurrence of this situation, as shown by DS2 in Figure 4-22. After the data suppress packet is sent, the status remains until a data packet or auxiliary packet is output again (when the space in the FIFO rises above the data suppress triggering level). Address packets are not affected by the data suppression.

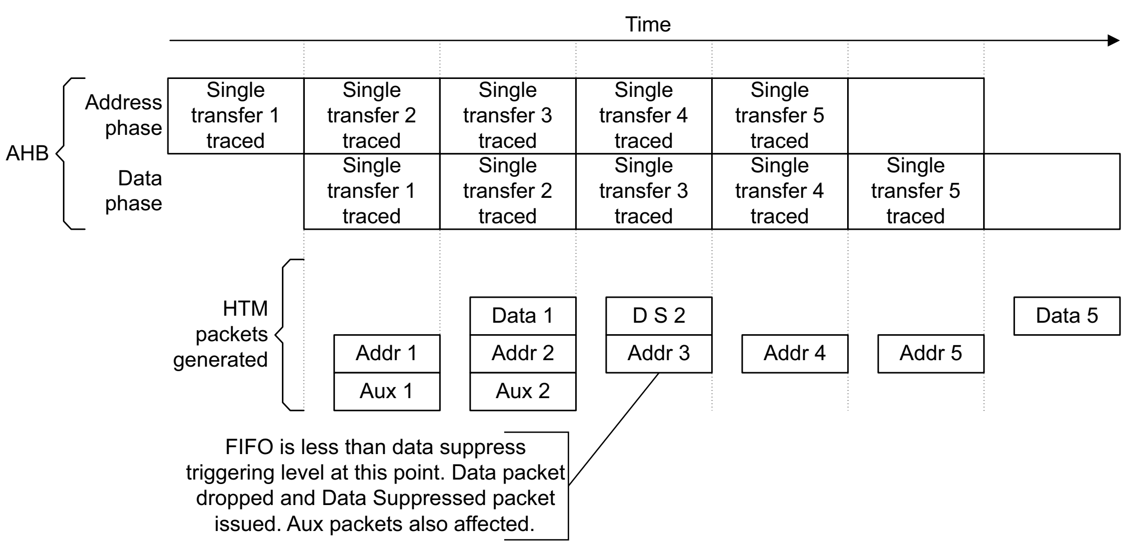


Figure 4-22 HTM use of data suppressed packets

4.6.6 Data suppress inside a burst

In burst transfer, the address packet is sent only in the first transfer. The remaining transfer is normally indicated by multiple data packets. However, if a data suppress packet is used during a burst, the remaining burst transfers become untraceable, as shown in Figure 4-23.

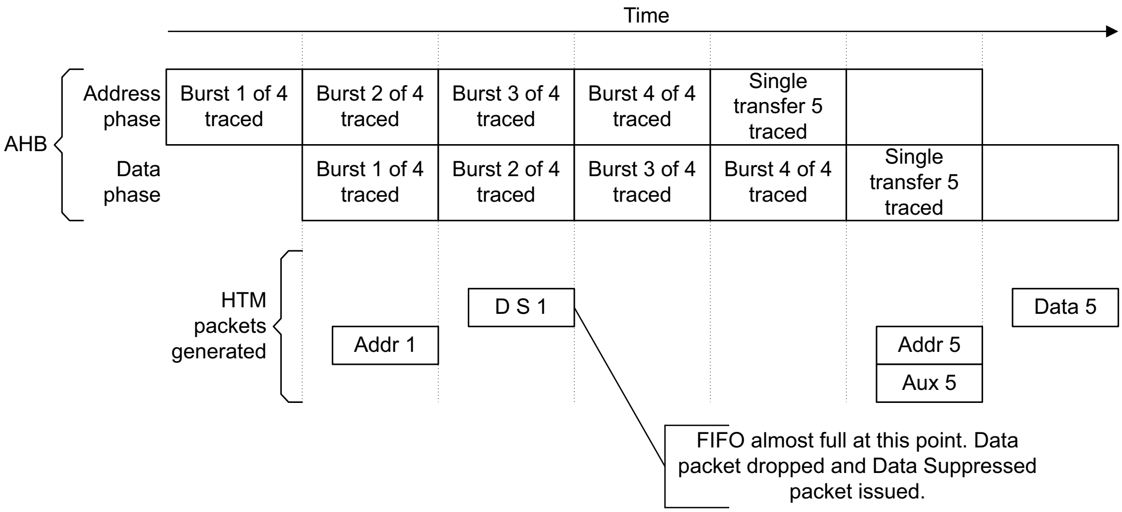


Figure 4-23 Data suppressed packet inside a burst

4.6.7 Data suppressed for single transfers

Data suppress status remains true until a new data packet or auxiliary packet is sent. If the space in FIFO is less than the data suppress triggering level, it is not necessary to resend data suppress packet even when there is a new single or burst transfer. In addition, data suppress can be caused by an auxiliary packet. After a Data Suppressed packet (DS1 in the diagram) is issued following packets do not have to issue Data Suppressed packets again, as shown in Figure 4-24 on page 4-21.

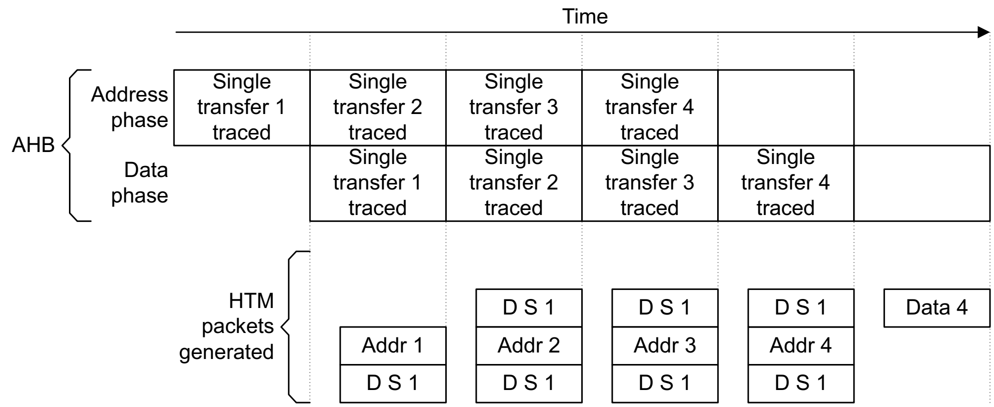


Figure 4-24 Data suppressed packets in a single transfer

4.6.8 FIFO overflow (trace lost)

After FIFO overflow, trace is lost and no more packets can be generated until the FIFO is free again, as shown in Figure 4-25.

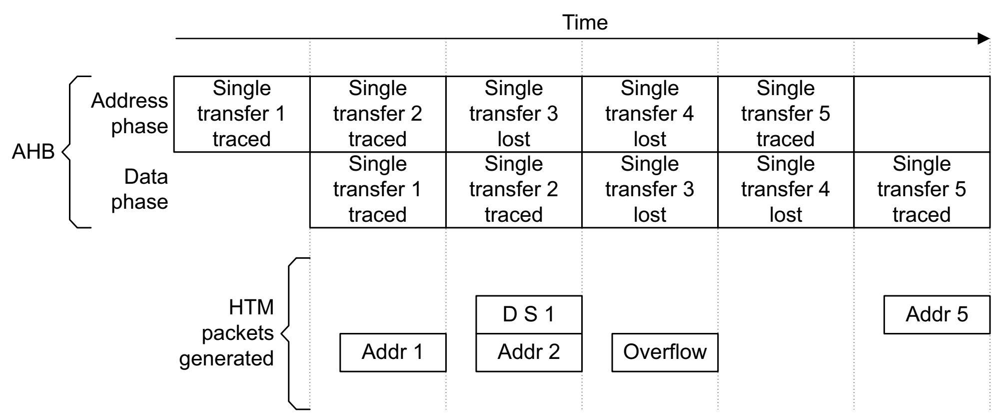


Figure 4-25 Lost trace and use of FIFO overflow packet

4.6.9 TraceOff control

The TraceOff packets are generated when:

- a stop address matches, as specified by the HTMSTARTSTOP register, and Trace Start/Stop control is used, as specified by the SSENABLE bit in the HTMTRRACECTRL register
- the trace is stopped by setting the PROG bit in the HTMCONTROL register.

Figure 4-26 shows that transfer 2 matches a stop address and transfer 5 matches a start address.

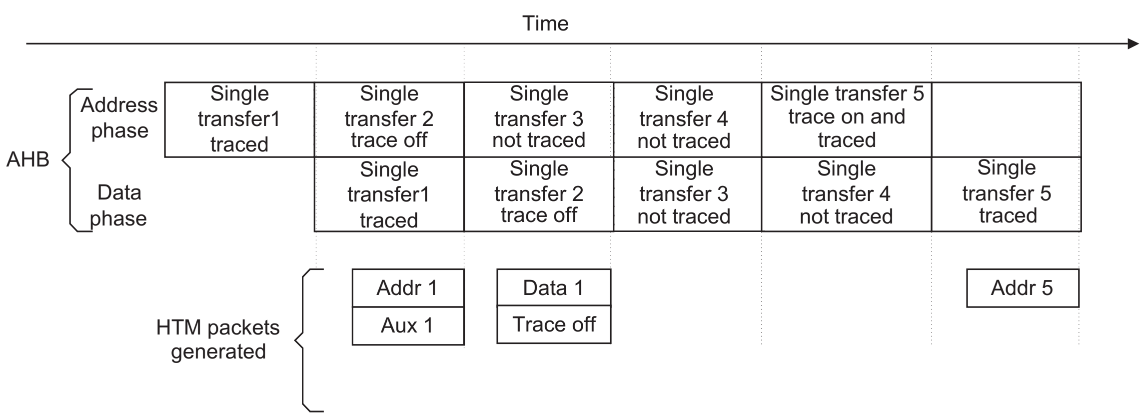


Figure 4-26 TraceOff operation

4.7 Cycle-accurate trace

The HTM provides the CycleCount packet for cycle-accurate trace and can link to other debug devices through external input/outputs for approximated timing correlation. The HTMCONTROL Register enables CycleCount packets. When CycleCount packets are enabled, they are inserted into the ATB data stream whenever a gap is present, as shown in Figure 4-27.

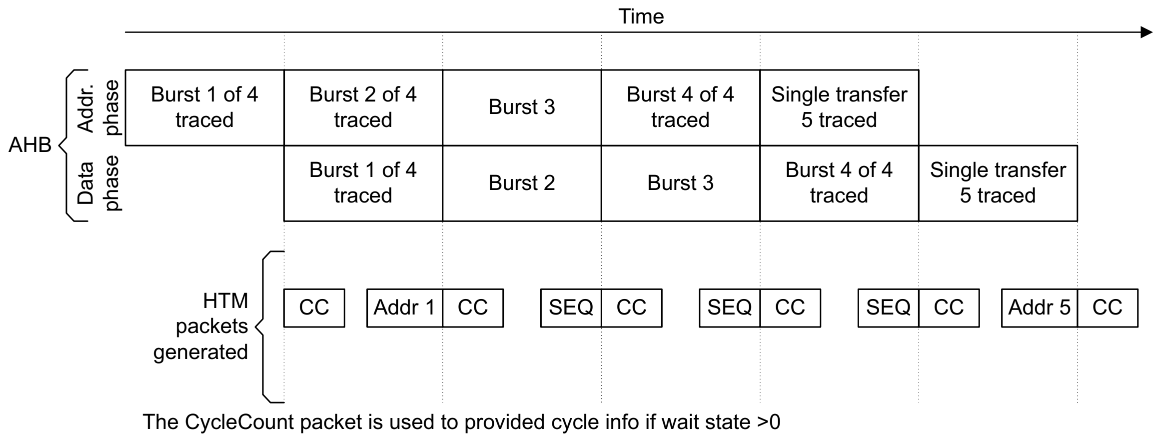


Figure 4-27 Use of CycleCount packet for timing-accurate trace

In the case where an AHB transfer is not traced, a CycleCount packet is inserted before the next traced address packet to indicate the time gap between the two traced packets, as shown in Figure 4-28.

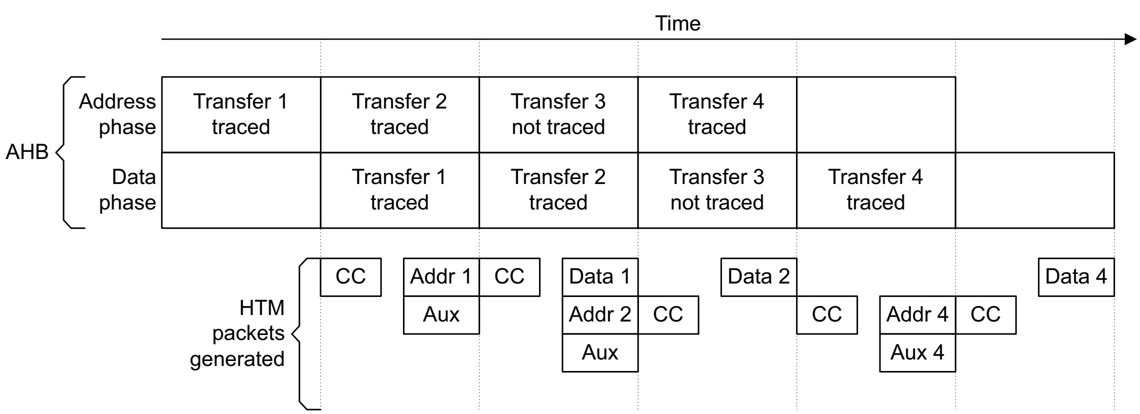


Figure 4-28 Additional CycleCount packet to show gaps in transfers

The gap in the trace operation can be caused by a transfer that is not traced because of an excluded address/transfer type, or because of trace being turned off by the Trace On/Off control, as shown in Figure 4-29.

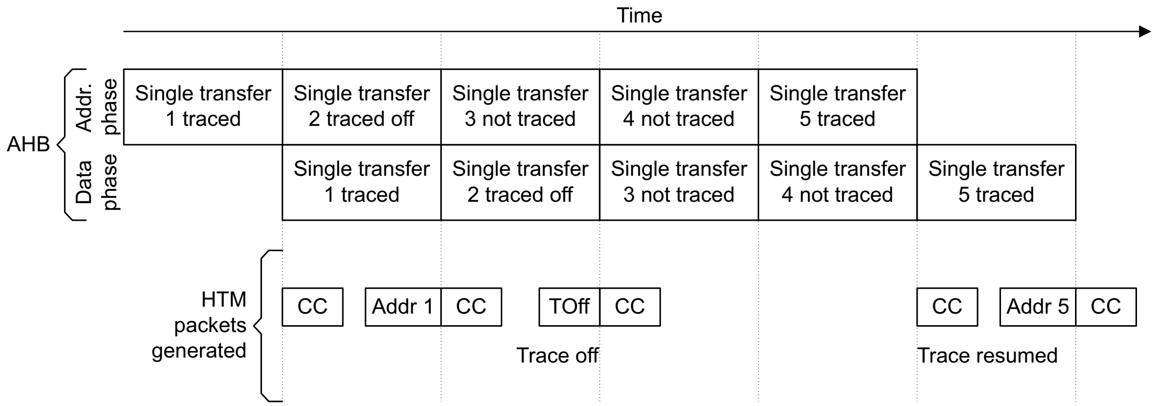


Figure 4-29 CycleCount packet after trace resumes

When there is no wait state between two traced transfers, the result is two address packets followed by a cycle count packet, as shown in Figure 4-30.

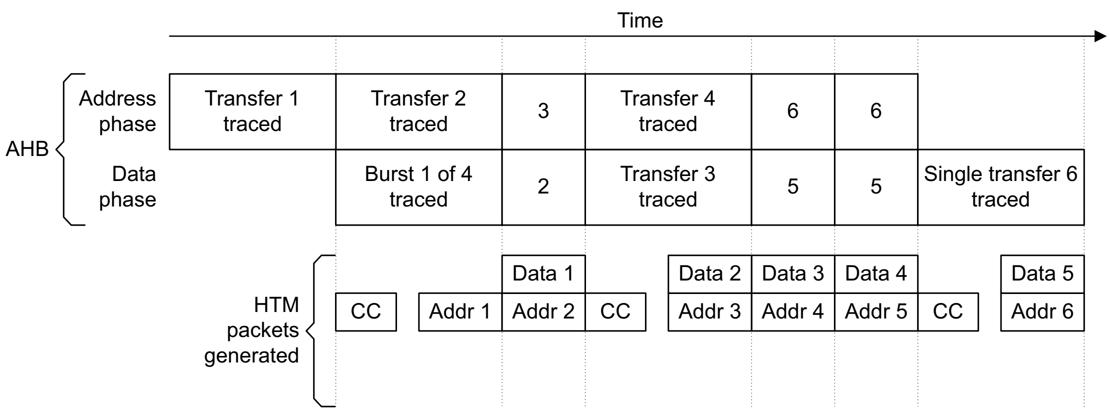


Figure 4-30 Use of CycleCount packet in transfers with no wait state

Cycle count packets can also be used as a way to measure time. When used with trigger packets, the time gap between two events can be measured as shown in Figure 4-31.

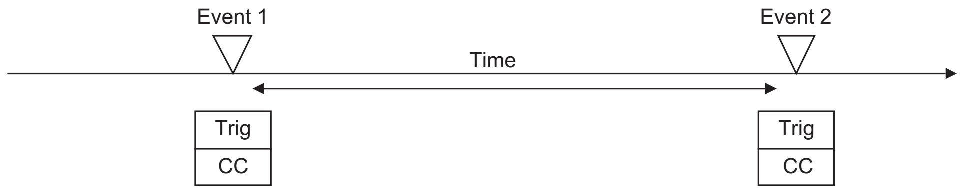


Figure 4-31 Time measurement with CycleCount packet

4.8 Reconstruction of timing information

You can use CycleCount packets to reconstruct timing in bus activities. In timing reconstruction the decompressor must take into account the various latencies in the system. This is described in:

- *Address, data, and auxiliary packets*
- *Reset On/Off packets and TraceOff packet* on page 4-28
- *Trigger packet* on page 4-28.

4.8.1 Address, data, and auxiliary packets

For address, data, and auxiliary packets the delay consists of a one-cycle delay for bus activities capture and another pipeline stage in the HTM as shown in Figure 4-32.

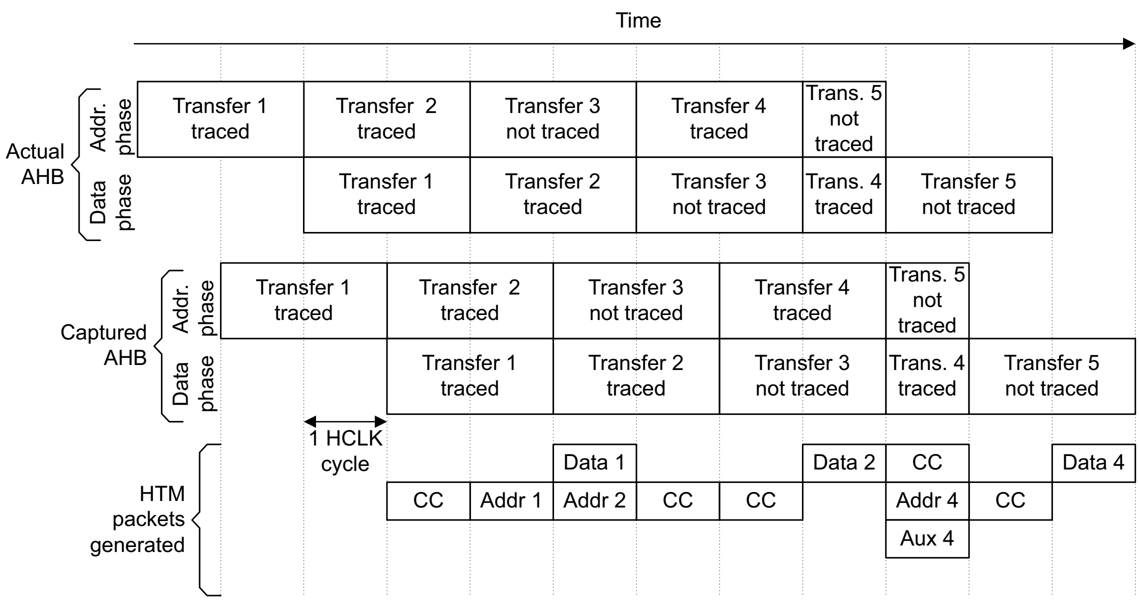


Figure 4-32 Delays from AHB bus activities to trace generation

A typical trace stream with wait states might contain the following ATB sequence. To simplify the examples, auxiliary packets are not listed.

```
<CC 0>
<Addr 1>
<CC 1>
<Data 1>
<Addr 2>
<CC 2>
```

```
<Data 2>
<Addr 3>
…
CC 0 = cycle from last Cycle Count packet to Address 1 accepted.
CC 1 = wait state of transfer 1
CC 2 = wait state of transfer 2
```

The correlated CycleCount packet is inserted after the transfer, when the captured **HREADY** signal changes from 1 to 0. However, if a transfer does not have any wait state, the following sequence is produced:

```
<Addr 1>
<Data 1>
<Addr 2>
<CC 2>
<Data 2>
…
```

There are two address packets before `<CC 2>`. In this case, the timing is interpreted as:

```
Wait state of transfer 2 = 0 (single cycle transfer)
Wait state of transfer 1 = CC2 - 1
```

Similarly, for three transfers with no wait state, a packet sequence might be:

```
<Addr 1>
<Data 1>
<Addr 2>
<Data 2>
<Addr 3>
<CC 3>
<Data 3>
```

The result is interpreted as:

```
Wait state of transfer 3 = 0 (single cycle transfer)
Wait state of transfer 2 = 0 (single cycle transfer)
Wait state of transfer 1 = CC3 - 2
```

4.8.2 Reset On/Off packets and TraceOff packet

For AHB reset packets and the TraceOff packet, the CycleCount packet is inserted one cycle after the event, as shown in Figure 4-33. However, AHB reset packet is not affected by the pipeline.

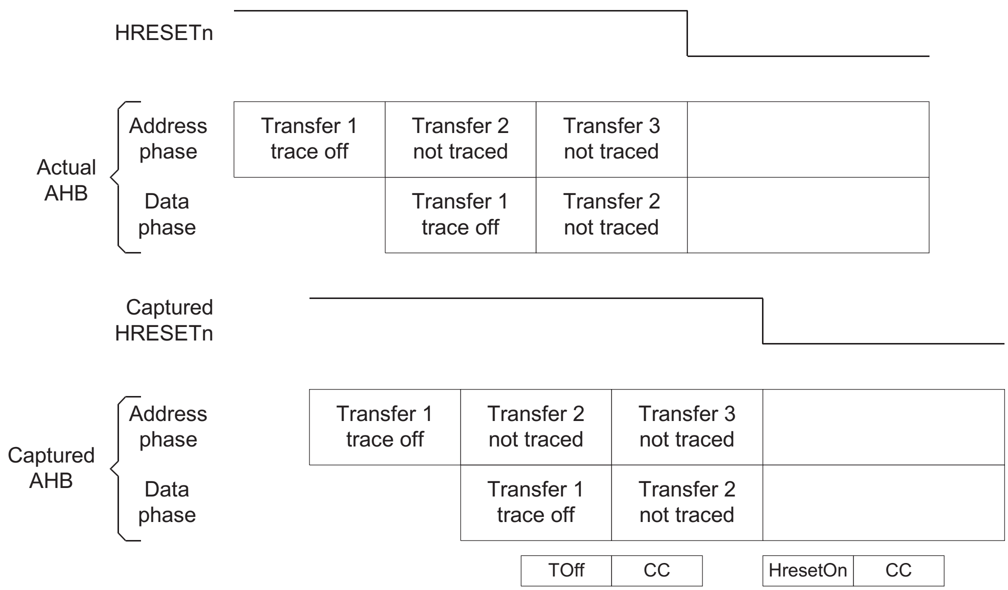


Figure 4-33 Delay of AHB Reset activities to trace generation

4.8.3 Trigger packet

The Trigger packet is generated one cycle behind the trigger event. Because of the design of the HTM, correlation between the Trigger packet and CycleCount packet behaves differently to other packets, as shown in Figure 4-34 on page 4-29:

- In case 1, when a packet sequence of a Trigger packet is directly followed by a CycleCount packet, the Trigger and CycleCount packet are generated in the same clock cycle.
- In case 2, when an Address packet is placed between the Trigger packet and CycleCount packet, the CycleCount packet is one cycle behind the Trigger packet. That is, there is a two cycle delay from trigger to cycle count.

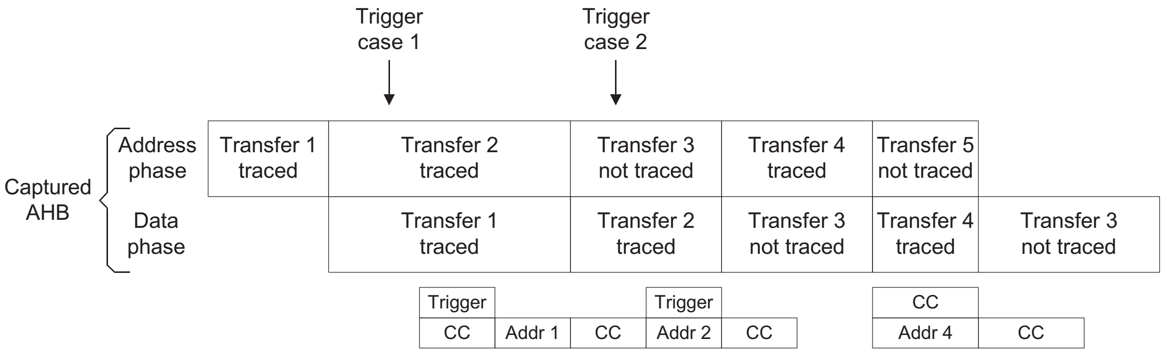


Figure 4-34 Delay of trigger activities to trace generation

4.9 Cycle timing characteristics of ATB packets and signals

Table 4-8 summarizes the cycle timing characteristics of ATB packets and signals

Table 4-8 Cycle timing characteristics of ATB packets and signals

Item	Type	Cycle Timing characteristics
Address packet	Trace packet	Timing accurate by coupling with cycle count packet in next cycle.
Auxiliary packet	Trace packet	Timing accurate by coupling with cycle count packet in next cycle.
Data packet	Trace packet	Timing accurate by coupling with cycle count packet in next cycle.
Sequential address	Trace packet	Timing accurate by coupling with cycle count packet in next cycle.
Trigger packet	Trace packet	Timing accurate by coupling with cycle count packet with two possible cases.
Overflow packet	Status packet	Indicate FIFO overflow, timing reconstruction after overflow is broken.
Data suppress packet	Status packet	Indicate data and auxiliary packets are suppressed. If address packets generation is enabled it does not affect timing reconstruction. But if address packet generation is disabled (only auxiliary packet or data packets are generated) then timing reconstruction after data suppress is broken.
AHB Reset on/off	Bus status packet	Timing accurate by coupling with cycle count packet in next cycle.
Ignore	Protocol packet	No coupling with cycle count.
A-SYNC	Protocol packet	No coupling with cycle count.
HTMEXTIN	Input signals	Delayed by 1 **HCLK** cycle by input register stage.
HTMEXTOUT	Output signals	Delayed by 1 **HCLK** cycle by output register stage.
HTMTRIGGER	Output signals	Delayed by 1 **HCLK** cycle by output register stage.

4.10 Synchronizing trace

There are several synchronization mechanisms in the HTM data protocol:

- A-SYNC packets are output regularly, and have a unique pattern. When the decompressor software finds the A-SYNC packet in a trace data stream, it can then decode the trace information after the A-SYNC.
- The address packets and auxiliary packets are compressed by sending only the lower bytes that have changed. Occasionally, a full address packet and auxiliary should be sent instead of compressed packets so that the decompressor can ensure the data can be decompressed correctly.

A counter in the HTM activates these synchronization mechanisms. The 12-bit counter decrements for each byte sent to the ATB. When the counter reaches zero, the next counter value is the value specified by the reload value register. By using the counter value, several synchronization points can be defined, as shown in Figure 4-35.

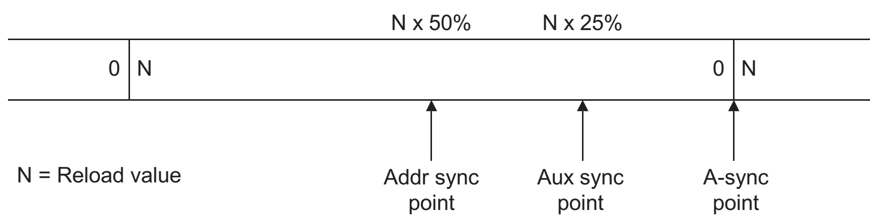


Figure 4-35 Synchronizing compression

When the counter reaches 0% of the counter reload value, the HTM starts looking for a cycle where an A-SYNC packet can be inserted, for example, when the auxiliary/trigger packet generator is not outputting data and the FIFO is not full.

When the counter reaches 25% of the counter reload value, the next time an auxiliary packet is generated, it is output in full.

When the counter reaches 50% of the counter reload value, the next time an address packet is generated, it is output in full.

The counter and the reload value registers are both 12 bits wide. The synchronization feature is disabled when the reload value register is set to 0. The A-SYNC packet is also output when trace operation starts.

The auxiliary packets can be used as a tool for profiling bus activity, by enabling only the auxiliary packet, with no address or data packet. The auxiliary packet is then output for every traced transfer, even if the control information remains the same. Data compression is still used to reduce the total number of bytes sent out in the auxiliary packet. With HTMAUXSEL set to `0xC`, `0xD`, `0xE`, or `0xF`, the auxiliary packet can provide useful information for generation of bus activity statistics.

4.11 Bandwidth limitations

In theory, the HTM can generate more than four words of trace information per cycle, but the ATB interface can only output one word per cycle in its maximum performance. Therefore it is easy to overflow the FIFO if too many transactions are traced. To avoid FIFO overflow, it is recommended that you only trace useful data accesses. Instruction executions must be traced with an ETM.

In a typical debug environment, an HTM is useful for:

- tracking accesses to a peripheral
- checking memory mapping with a small number of transfers
- monitoring accesses to small memory regions
- bus profiling (this uses only an auxiliary packet, with a maximum of 2 bytes per cycle).

The amount of information you can trace depends on the ATB bandwidth. A number of factors can affect the maximum bandwidth of the ATB bus:

- ATB bus frequency
- bandwidth sharing with other trace sources
- TPIU data port width.

If the trace has been unsuccessful because of FIFO overflow, you can solve this situation by reducing the amount of trace packets generated by:

- setting up better trace filtering rules
- using trace resources so that only useful information is traced and trace does not start until it is required
- disabling unnecessary packet types, for example, CycleCount
- increasing the ATB clock frequency
- increasing the TPIU data width if possible
- reducing the amount of trace data generated from other trace sources
- if it is necessary to trace instruction accesses, only enable the address packet generation.

During the system design stage, you can improve the trace capability by:

- using a wide TPIU data port width.
- including ETB in the design
- using a higher ATB clock frequency.

Chapter 5
Implementation-specific Characteristics

This chapter contains implementation-specific information relating to the CoreSight HTM. It contains the following sections:

5.1 HTM64 and HTM32 features summary

Table 5-1 summarizes the features of the 64-bit HTM and 32-bit HTM versions.

Table 5-1 HTM features for HTM64 and HTM32

Features	HTM64	HTM32	HTM32L
Sequencers	1	0	0
Counters	1	1	1
Address comparators	4	2	2
Address range comparators	2	1	1
AMBA control comparators	1	1	1
FIFO size	64Bytes	32Bytes	64Bytes
AHB data trace	64-bit	-	-
	32-bit	32-bit	32-bit
AHB data bus	64-bit wide	32-bit wide	32-bit wide
ARM11 signal extension support	Yes	No	No
ASIC Control Register size	8-bit	8-bit	8-bit
TrustZone signals support	**SPINDEN**	SPINDEN	SPINDEN
	SPIDEN	**SPIDEN**	**SPIDEN**
	NIDEN	NIDEN	NIDEN
	DBGEN	**DBGEN**	**DBGEN**
	HPROT[6]	-	-

HTM64 can be used to trace 64-bit wide data buses and 32-bit wide data buses. HTM32 can be used to trace 32-bit wide data buses only. Table 5-2 on page 5-3 summarizes values for the read bus and write bus signals.

Table 5-2 Bus sizes and HTM read and write data bus signals

Signal	With a 64-bit wide data bus	With a 32-bit wide data bus
HWDATAL[31:0]	Use for the lower 32 bits [31:0] of the write data bus.	Use for the write data bus.
HWDATAH[31:0]	Use for the upper 32 bits[63:32] of the write data bus.	Tie LOW.
HRDATAL[31:0]	Use for the lower 32 bits [31:0] of the read data bus.	Use for the read data bus.
HRDATAH[31:0]	Use for the upper 32 bits[63:32] of the read data bus.	Tie LOW.

5.2 CoreSight registers

The CoreSight registers are described in:

- *CoreSight management registers*
- *Peripheral and component identification registers* on page 5-5
- *Integration test registers* on page 5-6
- *Configuration code register* on page 5-6.

5.2.1 CoreSight management registers

Table 5-3 shows the CoreSight management registers.

Table 5-3 CoreSight management registers

Address offset	Name	Type	Width (bits)	Reset value	Description
0xF00	HTMITCR	R/W	1	0x0	*HTM Test Control Register, HTMITCR* on page 6-7
0xFA0	HTMCLAIMTAGSET	R/W	4	0x0F	*HTM Claim Tag Set Register, HTMCLAIMTAGSET* on page 3-43
0xFA4	HTMCLAIMTAGCLR	R/W	4	0x00	*HTM Claim Tag Clear Register, HTMCLAIMTAGCLR* on page 3-44
0xFB0	HTMLOCK_ACCESS	WO	32	-	*HTM Lock Access Register, HTMLOCK_ACCESS* on page 3-44
0xFB4	HTMLOCK_STATUS	RO	3	0x3	*HTM Lock Status Register, HTMLOCK_STATUS* on page 3-45
0xFB8	HTMAUTHSTATUS	RO	8	8b1-001-00	*HTM Authentication Status Register, HTMAUTHSTATUS* on page 3-47
0xFC8	HTMDEVID	RO	32	0x0	*HTM Device CoreSight ID Register, HTMDEVID* on page 3-48
0xFCC	HTMDEV_TYPE	RO	8	0x43	*HTM ATB Device Type Register, HTMDEV_TYPE* on page 3-48

5.2.2 Peripheral and component identification registers

Table 5-4 shows the peripheral and component identification registers.

Table 5-4 Peripheral and component identification registers

Address offset	Name	Type	Width (bits)	Reset value	Description
`0xFD0`	HTMPERIPHID4	RO	8	`0x04`	*Peripheral Identification Registers, HTMPERIPHID0-7* on page 3-49
`0xFD4`	HTMPERIPHID5	RO	8	`0x00`	Reserved
`0xFD8`	HTMPERIPHID6	RO	8	`0x00`	Reserved
`0xFDC`	HTMPERIPHID7	RO	8	`0x00`	Reserved
`0xFE0`	HTMPERIPHID0	RO	8	`0x17`	*Peripheral Identification Registers, HTMPERIPHID0-7* on page 3-49
`0xFE4`	HTMPERIPHID1	RO	8	`0xB9`	*Peripheral Identification Registers, HTMPERIPHID0-7* on page 3-49
`0xFE8`	HTMPERIPHID2	RO	8	`0x3B`	*Peripheral Identification Registers, HTMPERIPHID0-7* on page 3-49
`0xFEC`	HTMPERIPHID3	RO	8	`0x00`	*Peripheral Identification Registers, HTMPERIPHID0-7* on page 3-49
`0xFF0`	HTMCOMPONID0	RO	8	`0x0D`	*Identification Registers, HTMPCOMPONID0-3* on page 3-55
`0xFF4`	HTMCOMPONID1	RO	8	`0x90`	*Identification Registers, HTMPCOMPONID0-3* on page 3-55
`0xFF8`	HTMCOMPONID2	RO	8	`0x05`	*Identification Registers, HTMPCOMPONID0-3* on page 3-55
`0xFFc`	HTMCOMPONID3	RO	8	`0xB1`	*Identification Registers, HTMPCOMPONID0-3* on page 3-55

5.2.3 Integration test registers

Table 5-5 shows the integration test registers.

Table 5-5 HTM integration test registers

Address offset	Name	Type	Width (bits)	Reset value	Description
`0xF00`	HTMITCR	R/W	1	`0x0`	*HTM Test Control Register, HTMITCR* on page 6-7
`0xEF8`	HTMITATBCTR0	WO	10	`0x0`	*ATB Control Integration Test Register 0, HTMITATBCTR0* on page 6-7
`0xEF4`	HTMITATBCTR1	R/W	7	`0x0`	*ATB Control Integration Test Register 1, HTMITATBCTR1* on page 6-8
`0xEF0`	HTMITATBCTR2	RO	2	-	*ATB Control Integration Test Register 2, HTMITATBCTR2* on page 6-9
`0xEEC`	HTMITATBDATA0	WO	5	`0x0`	*ATB Data Integration Test Register 0, HTMITATBDATA0* on page 6-9
`0xEDC`	HTMITCYCCOUNT	R/W	32	-	*Cycle Counter Test Register, HTMITCYCCOUNT* on page 6-10
`0xED8`	HTMITTRACEDIS	RO	1	-	*External Trace Disable Integration Test Register, HTMITTRACEDIS* on page 6-10
`0xED4`	HTMITTRIGGER	WO	1	`0x0`	*Trigger Output Integration Test Register, HTMITTRIGGER* on page 6-11
`0xED0`	HTMITTRIGOUTACK	RO	1	-	*Trigger Output Acknowledge Integration Test Register, HTMITTRIGOUTACK* on page 6-12
`0xECC`	HTMITEXTIN	RO	2	-	*External Input Integration Test Register, HTMITEXTIN* on page 6-12

5.2.4 Configuration code register

The HTM Configuration Control Register provides the implementation configuration for the HTM64 and HTM32. For details on the register, see *HTM Configuration Code Register, HTMCFGCODE* on page 3-10.

5.3 HTM clocks and resets

The HTM has three clock signals:

- **HCLK**
- **PCLKDBG**
- **ATCLK**.

HCLK drives the logic in the SoC power domain, and **PCLKDBG** and **ATCLK** drive the logic in the debug power domain. **HCLK** can be asynchronous to the two other clock signals, but **ATCLK** and **PCLKDBG** must be synchronous to each other.

The clock frequency in the logic driven by **ATCLK** and **PCLKDBG** can be scaled down using the clock enable signals, **ATCLKEN** and **PCLKENDBG** respectively. This enables the logic to be operated at a speed which is an integer-divided ratio of the supplied clock frequency. Alternatively, the clock enable signal can be used to disable the debug logic.

There are three asynchronous reset signals for each of the clock domains:

- **HTMHRESETn** for **HCLK** domain logic
- **ATRESETn** for **ATCLK** domain logic
- **PRESETDBGn** for **PCLKDBG** domain logic.

These reset signals must be synchronized to their corresponding clocks by reset synchronization logic.

There is an additional reset input, **HRESETn**, which is used as an AHB reset status input. This signal does not reset the HTM logic, but activities on the signal can generate AHB reset on/off packets on the trace stream to indicate AHB reset has taken place.

If **HCLK** is turned off, **HTMRESETn/nCSOCPWRDN** is not asserted, and a debug agent attempts to access to HTM registers in the **HCLK** domain, it can cause a lockup of the Debug APB bus. To prevent this, it is recommended that an **HTMRESETn** or **nCSOCPWRDN** input is asserted when **HCLK** is turned off.

5.4 HTM restrictions

The following signals are not supported:

AHB signals

HSPLIT[15:0] (Split complete bus for split-capable AHB systems).

ARM11 sideband signals

HSIDEBAND[3:0] (ARM1136 inner memory access attribute).

WRITEBACK (ARM1136 cache memory access attribute).

You cannot use HTM32 to trace 64-bit wide data buses.

The use of data signals in the AMBA control comparator (when HTMHCTRLSEL = `0xF`) is limited to little-endian only.

Chapter 6
Programmer's Model for Test

This chapter describes the additional logic for functional verification and provisions made for production testing. It contains the following sections:

6.1 HTM test harness overview

The additional logic for functional verification and production testing enables:

- capture of HTM input signals to the block
- stimulation of the HTM output signals.

The integration vectors provide a way of verifying that the trace interface of the HTM is correctly wired into a system. This is done by separately testing four groups of signals:

APB signals These are tested by register access tests, which can verify the connections of all the address and data bits.

AHB signals These are tested by register access tests, which can verify the connections of all the address and data bits.

ATB signals These are tested by an example AHB capture, which can verify the connections of all the data bits. Additional test logic is added for testing of flush interface connection.

Intra-chip signals The tests for these signals are system-specific, and enable you to write the necessary tests. Additional logic is implemented enabling you to read and write to each intra-chip input/output signal.

Table 6-1 shows the test methods for the HTM signal groups.

Table 6-1 Test method for HTM signal connections

Signal group	Signals	Test method
APB bus	**PCLK**, **PCLKENDBG**, **PRESETDBGn**, **PSELDBG**, **PADDRDBG**, **PENABLEDBG**, **PWRITEDBG**, **PWDATADBG**, **PRDATADBG**, **PREADYDBG**, **PSLVERRDBUG**	APB register access test.
AHB bus	**HCLK**, **HRESETn**, **HTMHRESETn HADDR**, **HWRITE**, **HTRANS**, **HSIZE**, **HBURST**, **HPROT**, **HMASTER**, **HDOMAIN**, **HMASTLOCK**, **HUNALIGN**, **HBSTRB**, **HRDATAL**, **HRDATAH**, **HWDATAH**, **HWDATAH**, **HRESP**, **HREADY**, **HSEL**	Example AHB capture.

Table 6-1 Test method for HTM signal connections (continued)

Signal group	Signals	Test method
ATB bus	**ATCLK**, **ATCLKEN**, **ATRESETn**	Example AHB capture.
	ATVALIDM, **ATREADYM**, **ATDATAM**, **ATBYTESM**, **AFREADYM**, **AFVALIDM**, **ATIDM**	Example AHB capture and integration test registers.
Intra-Chip	**HTMSYNCBYPASS**	Read from HTMSTATUS Register.
	HTMMAXBUS	Read from HTMCFGCODE2 Register.
	HTMBUSSELECT	Output only. Test depends on SoC implementation.
	HTMASICCTRL	Output only. Test depends on SoC implementation.
	HTMTRIGGER, **HTMTRIGOUTACK**	Integration test registers.
	HTMEXTIN	Integration test registers.
	HTMEXTOUT	Output only. Test depends on SoC implementation.
	HTMSPNIDEN, **HTMNIDEN**, **HTMDBGEN**, **HTMSPIDEN**	Read only. For security reasons these signals must not be changed by integration logic.
	HTMTRACEDISABLE	Integration test registers.

Test registers control these test features. This enables you to test the trace interface of the HTM in isolation from the rest of the system using only transfers from the AHB

A global register called HTMITCR must be activated before any integration tests can be performed. The HTMITCR register contains the ITEN bit, which is set to 1 during integration testing.

Figure 6-1 on page 6-4 shows the integration logic.

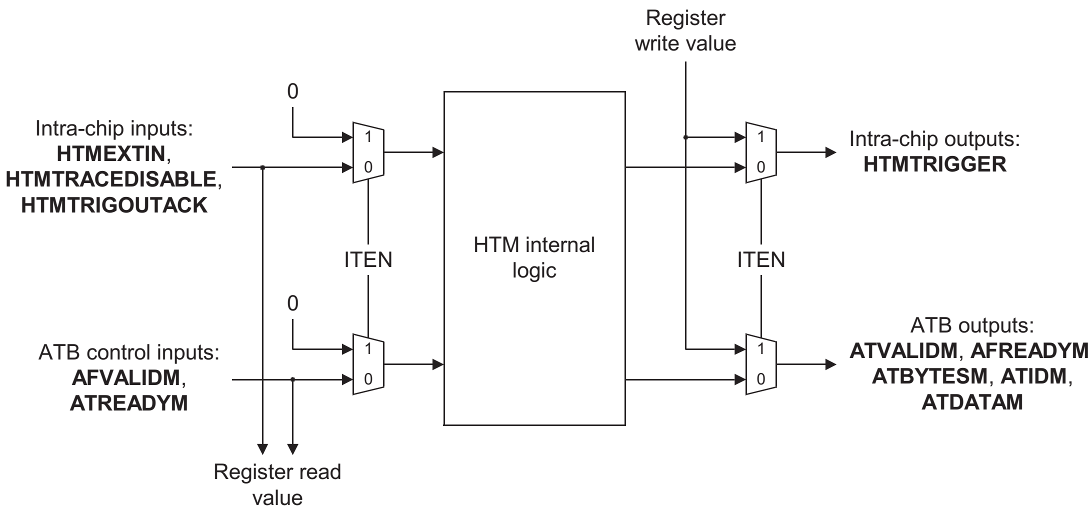


Figure 6-1 Integration logic

6.1.1 Integration test for AHB and ATB connections

To verify connections for the AHB and ATB interface, a set of test vectors is included in the CoreSight Integration Kit. These test vectors generate simple traces that can verify the AHB and ATB connections. See the *CoreSight Design Kit Implementation and Integration Manual* for more information.

6.2 Scan testing

The HTM design simplifies:

- insertion of scan test cells
- use of *Automatic Test Pattern Generation* (ATPG).

This is the recommended method of manufacturing test. Dummy scan control ports are provided in RTL for scan insertion. Scan data input and output ports are generated by synthesis scripts.

6.3 Test registers

The HTM test registers are memory-mapped as shown in Table 6-2.

Table 6-2 HTM integration test registers

Address offset	Name	Type	Width (bits)	Reset value	Description
`0xF00`	HTMITCR	R/W	1	`0x0`	*HTM Test Control Register, HTMITCR* on page 6-7
`0xEF8`	HTMITATBCTR0	WO	10	`0x0`	*ATB Control Integration Test Register 0, HTMITATBCTR0* on page 6-7
`0xEF4`	HTMITATBCTR1	R/W	7	`0x0`	*ATB Control Integration Test Register 1, HTMITATBCTR1* on page 6-8
`0xEF0`	HTMITATBCTR2	RO	2	-	*ATB Control Integration Test Register 2, HTMITATBCTR2* on page 6-9
`0xEEC`	HTMITATBDATA0	WO	5	`0x0`	*ATB Data Integration Test Register 0, HTMITATBDATA0* on page 6-9
`0xEDC`	HTMITCYCCOUNT	R/W	31:0	-	*Cycle Counter Test Register, HTMITCYCCOUNT* on page 6-10
`0xED8`	HTMITTRACEDIS	RO	1	-	*External Trace Disable Integration Test Register, HTMITTRACEDIS* on page 6-10
`0xED4`	HTMITTRIGGER	WO	1	`0x0`	*Trigger Output Integration Test Register, HTMITTRIGGER* on page 6-11
`0xED0`	HTMITTRIGOUTACK	RO	1	-	*Trigger Output Acknowledge Integration Test Register, HTMITTRIGOUTACK* on page 6-12
`0xECC`	HTMITEXTIN	RO	2	-	*External Input Integration Test Register, HTMITEXTIN* on page 6-12

All registers are only accessible when the HTMITCR has been configured as described in *HTM Test Control Register, HTMITCR* on page 6-7.

All registers can be accessed with a minimum of three clock cycles, if the **HCLK** and Debug APB **PCLKDBG** are the same, otherwise wait states depend on the ratio between the two clocks.

For all registers, unimplemented bits return 0 when read and ignore writes. The test registers are included to aid test, debug and integration of the HTM and the trace interface block as a whole.

6.3.1 HTM Test Control Register, HTMITCR

The HTMITCR register is a read/write test control register. The ITEN bit in this register controls the input and output of test control registers.

Figure 6-2 shows the bit assignments.

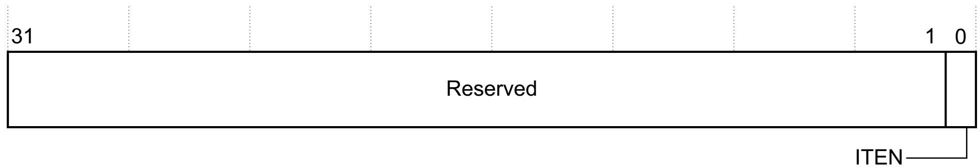


Figure 6-2 HTMITCR register bit assignments

Table 6-3 shows the bit assignments.

Table 6-3 HTMITCR register bit assignments

Bits	Name	Function
[31:1]	Reserved	Reserved. Read as zero, do not modify.
[0]	ITEN	Integration test enable: 1 = test mode enabled. 0 = normal mode (reset).

6.3.2 ATB Control Integration Test Register 0, HTMITATBCTR0

The HTMITATBCTR0 register is used to control the values of the **AFREADYM**, **ATVALIDM**, and **ATBYTESM** outputs in integration test mode. This register must only be used in test mode.

Figure 6-3 shows the bit assignments.

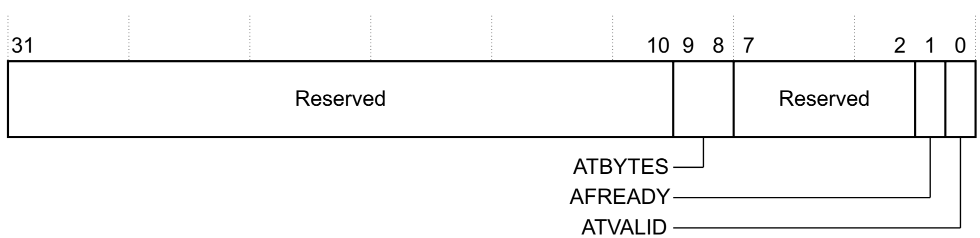


Figure 6-3 HTMITATBCTR0 Register bit assignments

Table 6-4 shows the bit assignments.

Table 6-4 HTMITATBCTR0 Register bit assignments

Bits	Name	Function
[31:10]	Reserved	Reserved. Read as zero, do not modify.
[9:8]	ATBYTES	When ITEN is 1, the value in this field determines the **ATBYTESM[1:0]** output.
[7:2]	Reserved	Reserved. Read as zero, do not modify.
[1]	AFREADY	When ITEN is 1, the value in this field determines the **AFREADYM** output.
[0]	ATVALID	When ITEN is 1, the value in this field determines the **ATVALIDM** output.

6.3.3 ATB Control Integration Test Register 1, HTMITATBCTR1

The HTMITATBCTR1 register is used to control the value of the **ATIDM** output in integration test mode. This register is shared with the HTMATIDOUT register, therefore changing this register changes HTMATIDOUT and vice versa. This register must only be used in test mode.

Figure 6-4 shows the bit assignments.

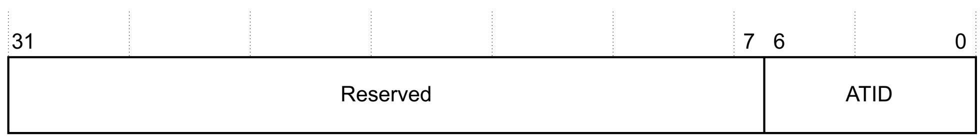


Figure 6-4 HTMITATBCTR1 Register bit assignments

Table 6-5 shows the bit assignments.

Table 6-5 HTMITATBCTR1 Register bit assignments

Bits	Name	Function
[31:7]	Reserved	Reserved. Read as zero, do not modify.
[6:0]	ATID	The value in this field determines the **ATIDM[6:0]** output.

6.3.4 ATB Control Integration Test Register 2, HTMITATBCTR2

The HTMITATBCTR2 register is used to control and read the values of the **AFVALIDM** and **ATREADYM** inputs. This register must only be used in test mode.

Figure 6-5 shows the bit assignments.

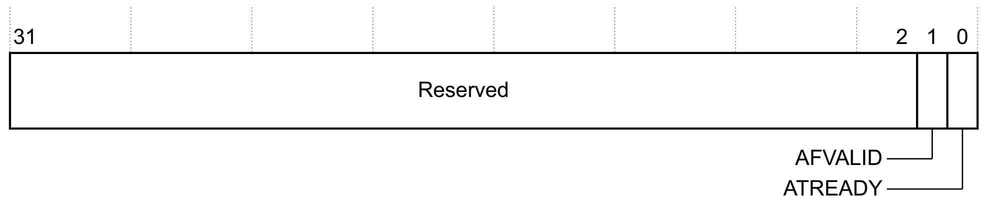


Figure 6-5 HTMITATBCTR2 Register bit assignments

Table 6-6 shows the bit assignments.

Table 6-6 HTMITATBCTR2 Register bit assignments

Bits	Name	Function
[31:2]	Reserved	Reserved. Read as zero, do not modify.
[1]	AFVALID	Read the current value on **AFVALIDM** input.
[0]	ATREADY	Read the current value on **ATREADYM** input.

6.3.5 ATB Data Integration Test Register 0, HTMITATBDATA0

The HTMITATBDATA0 register is used to control the values of the **ATDATAM** outputs in integration test mode. This register must only be used in test mode.

Figure 6-6 shows the bit assignments.

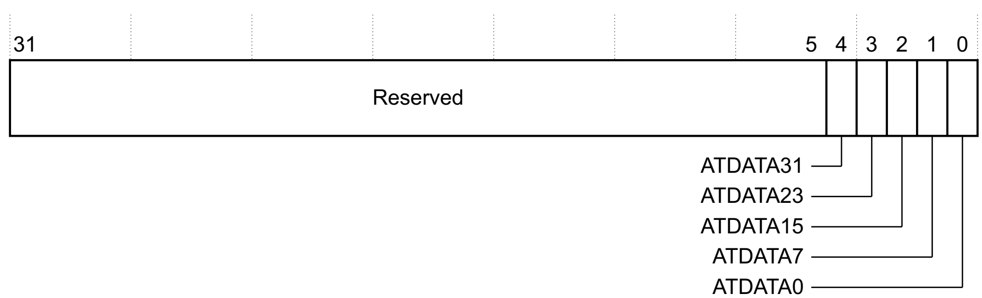


Figure 6-6 HTMITATBDATA0 Register bit assignments

Table 6-7 shows the bit assignments.

Table 6-7 HTMITATBDATA0 Register bit assignments

Bits	Name	Function
[31:5]	Reserved	Reserved. Read as zero, do not modify.
[4]	ATDATA31	When ITEN is 1, this field determines the **ATDATAM[31]** output.
[3]	ATDATA23	When ITEN is 1, this field determines the **ATDATAM[23]** output.
[2]	ATDATA15	When ITEN is 1, this field determines the **ATDATAM[15]** output.
[1]	ATDATA7	When ITEN is 1, this field determines the **ATDATAM[7]** output.
[0]	ATDATA0	When ITEN is 1, this field determines the **ATDATAM[0]** output.

6.3.6 Cycle Counter Test Register, HTMITCYCCOUNT

The HTMITCYCCOUNT register is used for testing of cycle counter logic in the cycle count packet generation. Writing to this register can change the value of the cycle counter. The cycle counter is cleared to 1 if bit [4] of the HTMCONTROL register is clear. The counter value increments when the PROG bit in the HTMCONTROL is 0, and cleared to 1 when a cycle count packet is generated. If the generation of cycle count packet is enabled and the PROG bit is zero, and no trace is generated, the counter increments continuously. If the counter value reaches `0xFFFFFFFF`, it stays unchanged. This register must only be used in test mode.

Table 6-8 shows the bit assignments.

Table 6-8 HTMITCYCCOUNT Register bit assignments

Bits	Name	Function
[31:0]	HTMITCYCCOUNT	Read/write to the value of the cycle counter. The cycle counter is cleared to 1 if bit [4] of HTMCONTROL register is cleared. Otherwise, the counter can be stopped or be programmed to any value by setting the PROG bit in the HTMCONTROL register to 1.

6.3.7 External Trace Disable Integration Test Register, HTMITTRACEDIS

The HTMITTRACEDIS register is used to control and read the value of the **HTMITTRACEDIS** input. This register must only be used in test mode.

Figure 6-7 on page 6-11 shows the bit assignments.

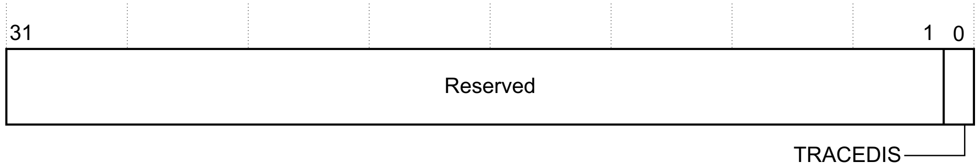


Figure 6-7 HTMITTRACEDIS Register bit assignments

Table 6-9 shows the bit assignments.

Table 6-9 HTMITTRACEDIS Register bit assignments

Bits	Name	Function
[31:1]	Reserved	Reserved. Read as zero.
[0]	TRACEDIS	Reads the value of the **HTMTRACEDISABLE** input.

6.3.8 Trigger Output Integration Test Register, HTMITTRIGGER

The HTMITTRIGGER register is used to control the value of the **HTMITTRIGGER** output. This register must only be used in test mode.

Figure 6-8 shows the bit assignments.

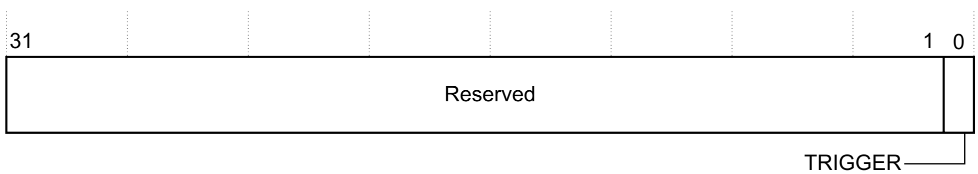


Figure 6-8 HTMITTRIGGER Register bit assignments

Table 6-10 shows the bit assignments.

Table 6-10 HTMITTRIGGER Register bit assignments

Bits	Name	Function
[31:1]	Reserved	Reserved. Read as zero, do not modify.
[0]	TRIGGER	When ITEN is 1, this field determines the **HTMTRIGGER** output.

6.3.9 Trigger Output Acknowledge Integration Test Register, HTMITTRIGOUTACK

The HTMITTRIGOUTACK register is used to read and control the value of **HTMTRIGOUTACK** input. This register must only be used in test mode.

Figure 6-9 shows the bit assignments.

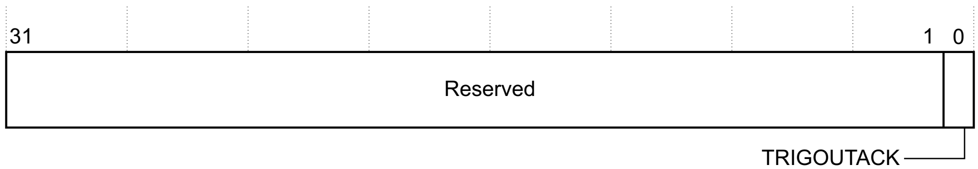


Figure 6-9 HTMITTRIGOUTACK Register bit assignments

Table 6-11 shows the bit assignments.

Table 6-11 HTMITTRIGOUTACK Register bit assignments

Bits	Name	Function
[31:1]	Reserved	Reserved. Read as zero.
[0]	TRIGOUTACK	Reads the value of the **HTMTRIGOUTACK** input.

6.3.10 External Input Integration Test Register, HTMITEXTIN

The HTMITEXTIN register is used to read and control the value of **HTMEXTIN** input. The width of this register depends on how many external inputs are implemented. This register must only be used in test mode.

Figure 6-10 shows the bit assignments.

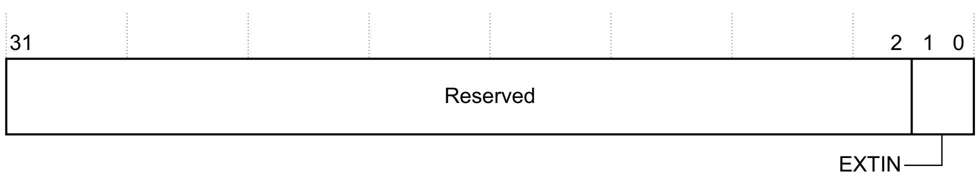


Figure 6-10 HTMITEXTIN Register bit assignments

Table 6-12 shows the bit assignments.

Table 6-12 HTMITEXTIN Register bit assignments

Bits	Name	Function
[31:2]	Reserved	Reserved. Read as zero.
[1:0]	EXTIN	Reads the value of the **HTMEXTIN** input.

Appendix A
Signal Descriptions

This appendix describes the signals used in the HTM. It contains the following sections:

A.1 HTM signals

The HTM module is connected to the AHB as a bus slave. Table A-1 shows the HTM AHB signals.

Table A-1 HTM AHB signals

Name	Type	Source/ destination	Description
HCLK	Input	Clock source	AHB bus clock, used to time all bus transfers. All signal timings are related to the rising edge of **HCLK**.
HRESETn	Input	Reset controller	AHB bus reset, active LOW.
HTMHRESETn	Input	Reset controller	HTM AHB reset, active LOW.
HADDR[31:0]	Input	Master	AHB system address bus.
HSEL[13:0]	Input	AHB decoder	AHB select information. Same phase as **HADDR**
HTRANS[1:0]	Input	Master	Transfer type, which can be NONSEQUENTIAL, SEQUENTIAL, IDLE, or BUSY.
HWRITE	Input	Master	Transfer direction: HIGH indicates a write transfer LOW indicates read transfer.
HSIZE[2:0]	Input	Master	Size of the transfer.
HPROT[6:0]/ HPROT[3:0][a]	Input	Master	Memory access protection type. **HPROT[6:4]** only present in HTM64
HBURST[2:0]	Input	Master	Transfer burst type
HMASTLOCK	Input	Master	Lock transfer indication
HMASTER[3:0]	Input	Arbiter	Master of current transfer
HDOMAIN[3:0][a]	Input	Master	Domain of current transfer
HUNALIGN[a]	Input	Master	Indicates the transfer is unaligned
HBSTRB[7:0][a]	Input	Master	Byte lane strobe for unaligned transfers
HWDATAL[31:0]	Input	Master	Lower 32-bit of write data bus, used to transfer data from bus master to bus slaves during write operations.

Table A-1 HTM AHB signals (continued)

Name	Type	Source/ destination	Description
HWDATAH[31:0]	Input	Master	Upper 32-bit of write data bus, used to transfer data from bus master to bus slaves during write operations. Connect this to bits[63:32] if 64-bit data bus is used, or tied LOW if 32-bit data bus is used.
HREADY	Input	External slave	Transfer done signal, generated by an alternate slave. When HIGH, indicates that a transfer is complete. Can be driven LOW to extend a transfer.
HRDATAL[31:0]	Input	Slave	Lower 32-bit of read data bus, used to transfer data from bus slaves to bus master during read operations.
HRDATAH[31:0]	Input	Slave	Upper 32-bit of read data bus, used to transfer data from bus slaves to bus master during read operations. Connect this to bit[63:32] if 64-bit data bus is used, or tied LOW if 32-bit data bus is used.
HRESP[2:0]/ HRESP[1:0][a]	Input	Slave	Transfer response, which provides additional transfer status information. The response can be OKAY, ERROR, RETRY, SPLIT, or XFAIL.

a. See the ARM11 specification or Systems IP ARM11 AMBA AHB Extensions for more information about this signal.

A.2 HTM ATB signals

Table A-2 shows the HTM ATB signals

Table A-2 HTM ATB signals

Name	Type	Source/ destination	Description
ATCLK	Input	Clock source	ATB clock
ATRESETn	Input	Clock source	Reset of ATB bus
ATCLKEN	Input	Clock source	ATB clock enable
ATIDM[6:0]	Output	Downstream ATB device	Current trace source ID
ATDATAM[31:0]	Output	Downstream ATB device	Trace data, 1-4 bytes, valid with data always aligned to the LSB.
ATVALIDM	Output	Downstream ATB device	There are valid signals this cycle from the trace source.
ATBYTESM[1:0]	Output	Downstream ATB device	Size of data, 1-4 bytes.
ATREADYM	Input	Downstream ATB device	If there is valid data, **ATVALIDM** HIGH, then the data was accepted this cycle.
AFVALIDM	Input	Downstream ATB device	Any data present in buffers must be flushed. New data is still added.
AFREADYM	Output	Downstream ATB device	Data flush complete. All old data in buffers before assertion of **AFVALIDM** has been removed.

A.3 HTM external signals

Table A-3 shows the HTM external signals.

Table A-3 HTM external signals

Name	Type	Source/ destination	Description
BUSWIDTH	Input	AHB	Current width of AHB data bus: 0 = 32-bit 1 = 64-bit
BUSENDIAN[a]	Input	AHB	Endianness of current transfer on AHB: 0 = Little-endian or byte invariant big endian mode BE-8 1 = Word invariant big-endian, BE32
HTMMAXBUS[2:0]	Input	System	Static signal. Maximum permitted value for **HTMBUSSELECT**.
HTMEXTIN[1:0]	Input	ECT[b]/System	Input trigger
HTMEXTOUT[1:0]	Output	ECT/System	Output event.
HTMBUSSELECT[2:0]	Output	Optional bus multiplexer	If bus multiplexer is implemented to monitor multiple AHB, this signal is used to control the multiplexer
HTMTRIGGER	Output	ECT/System	Trigger output: Assert and stay HIGH when a trigger event is generated and not suppressed (single trigger mode). This output de-asserts when **HTMTRIGOUTACK** is HIGH.
HTMTRIGOUTACK	Input	ECT/System	Acknowledge of **HTMTRIGGER**. This is a handshaking signal to enable **HTMTRIGGER** to be connected to the ECT without wrapper.
HTMASICCTRL[7:0]	Output	System	ASIC Control outputs
HTMGLBEN	Output	System	System management
HTMSPNIDEN[c]	Input	System	Secure Privileged Non-Invasive Debug Enable
HTMSPIDEN [c]	Input	System	Secure Privileged Invasive Debug Enable
HTMNIDEN[c]	Input	System	Non-invasive Debug Enable
HTMDBGEN[c]	Input	System	Invasive Debug Enable
HTMTRACEDISABLE	Input	System	Trace disable control. Can be connected to **DBGACK** from CPU (for single-core systems only)

Table A-3 HTM external signals (continued)

Name	Type	Source/ destination	Description
SYNCBYPASS	Input	System	Synchronization bypass between **HCLK** domain and **PCLK** domain.
nCDBGPWRDN	Input	System	Debug system power-down indication: When this pin is LOW, all signals from power domain CSHtmSOC[d] to CSHtmDBG are forced to zero.
nCSOCPWRDN	Input	System	System on Chip power-down indication: When this pin is LOW, all signals from power domain CSHtmDBG[b] to CSHtmSOC are forced to zero. All accesses to registers in CSHtmSOC result in slave error response.

a. BE-32 is supported on ARM10 and ARM11 cores. BE-8 is supported on ARM-11 cores. See *Endianness support* on page 2-24for more information.

b. *Embedded Cross Trigger* (ECT). See *HTM used in a CoreSight system* on page 1-3 for example uses of ECT and HTM.

c. TrustZone configuration signals. Because the HTM is non-invasive, the security access control signals SPNIDEN and NIDEN are required. To enable easy integration with most systems, SPIDEN and DBGEN are also provided.

When SPNIDEN and SPIDEN are both LOW, non-invasive debugging functions such as trace and profiling must not be permitted on operations in secure state. When NIDEN and DBGEN are both LOW, all non-invasive debugging must be disabled.

The HTM trace must be disabled when NIDEN is LOW. SPNIDEN must prevent secure transfers being traced. Previously traced data in the FIFO is not affected by these signals.

d. See *Power-down indication signals and clamping logic* on page A-9 and *HTM power-down behavior* on page A-12 for more information on power-down and clamping.

A.4 HTM APB signals

Table A-4 shows the HTM APB signals.

Table A-4 HTM APB signals

Name	Type	Source/ destination	Description
PCLKDBG	Input	Clock source	APB Clock
PCLKENDBG	Input	Clock source	APB Clock enable. Tie HIGH when not in use.
PRESETDBGn	Input	System reset generator	APB Bus Reset, active LOW
PADDRDBG[11:0]	Input	Debug APB	APB address bus
PADDRDBG31	Input	Debug APB	APB address bus bit 31 (lock bypass access mode)
PWRITEDBG	Output	Debug APB	When HIGH indicates an APB write access and when LOW a read access
PENABLEDBG	Output	Debug APB	The enable signal is used to indicate the second and subsequent cycles of a APB transfer
PWDATADBG[31:0]	Output	Debug APB	The write bus is driven by the APB Master during write cycles (when **PWRITEDBG** is HIGH)
PRDATADBG[31:0]	Input	Debug APB	The read bus is driven by the selected slave (such as a CoreSight component) during read cycles (when **PWRITEDBG** is LOW)
PREADYDBG	Output	Debug APB	The ready signal used by the slave to extend an APB transfer
PSLVERRDBG	Output	Debug APB	Error response of the Debug APB interface
PSELDBG	Input	Debug APB	Select HTM registers

A.5 HTM scan test control signals

Table A-5 shows the HTM scan test control signal.

Table A-5 HTM scan test control signal

Name	Type	Source/ destination	Description
SE	Input	Scan controller	Scan enable, for all clock domains.

A.6 Power-down indication signals and clamping logic

This section describes the built-in clamping logic for HTM and signal behavior in the following sections:

- *Power domains*
- *HTM signal behavior on power-down* on page A-11
- *HTM power-down behavior* on page A-12.

A.6.1 Power domains

The HTM design is partitioned into two sections:

- CSHtmDBG
- CSHtmSOC.

Signal clamping logic is provided between the two blocks to enable power-down of part of the system and to ensure that signals do not drive voltage into the parts of HTM that are already powered down.

Note

By default, the clamping is not instantiated. This reduces unnecessary logic in most cases. Restoring the clamping logic can be done by editing the Verilog file `CSHTMDefs.v`. Uncomment out the line

```
`define CSHTM_CLAMP_LOGIC
```

After changing the code, several of the provided test vectors might fail. This is normal.

See the *CoreSight Design Kit Implementation and Integration Manual* for more information on the compiler directives in the HTM design files.

Figure A-1 on page A-10 shows the power down signals and clamping logic.

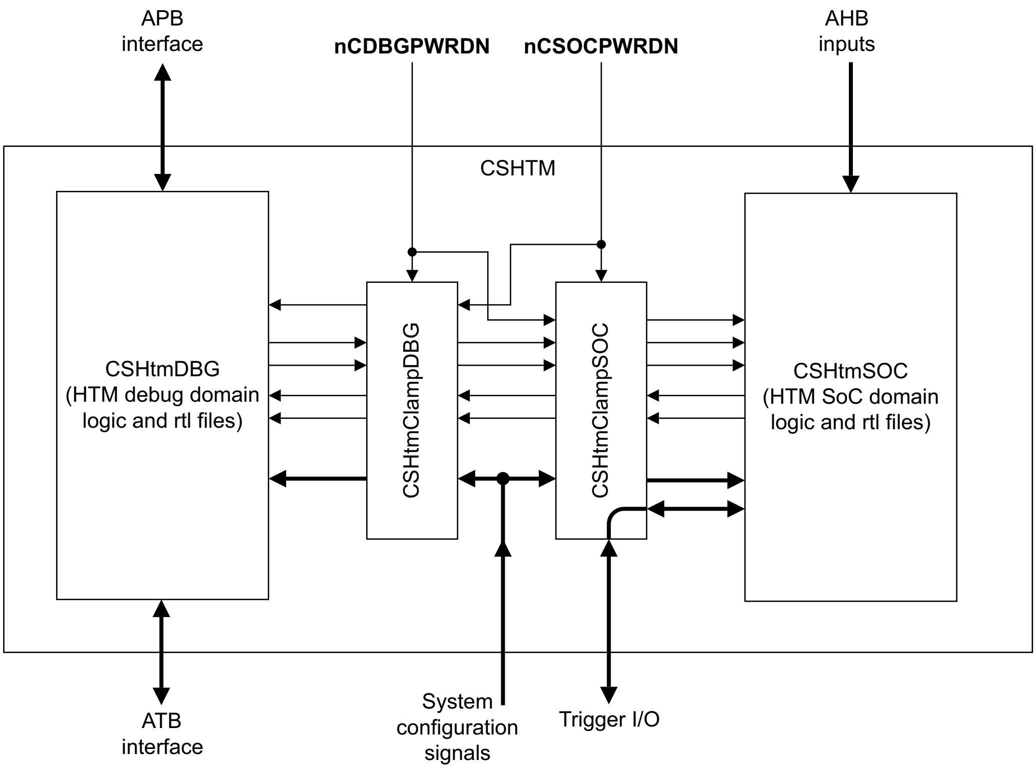


Figure A-1 Clamping logic and power-down indication signals

Most of the external inputs are also connected through the clamping logic to ensure they do not drive voltage into the parts of HTM that are powered down. There is no clamping on the bus interfaces because the bus signals are already reset to LOW during power-down.

A.6.2 HTM signal behavior on power-down

Table A-6 shows the behavior of HTM signals when the power-down indication signals are asserted.

Table A-6 HTM signal behavior on assertion of power down

Signal name	Type	Power domain	Behavior when nCSOCPWRDN = 0	Behavior when nCSDBGPWRDN = 0
HTMNIDEN	Input	Both	Clamp to 0	Clamp to 0 (status register)
HTMSPNIDEN	Input	Both	Clamp to 0	Clamp to 0 (status register)
HTMTRACEDISABLE	Input	Both	Trace Disable control to SoC domain clamped to 0	Trace Disable to debug power domain (status register) clamped to 0
HTMEXTIN	Input	SOC	Clamp to 0	Not affected
HTMEXTOUT	Output	SOC	Clamp to 0	Not affected
HTMTRIGGER	Output	SOC	Clamp to 0	Not affected
HTMTRIGOUTACK	Input	SOC	Clamp to 0	Not affected
HTMMAXBUS	Input	DBG	Not affected	Clamp to 0
HTMBUSSELECT	Output	SOC	Clamp to 0	Not affected
HTMASICCTRL	Output	SOC	Not clamped	Not affected
SYNCBYPASS	Input	Both	Clamp to 0	Clamp to 0
HTMGLBEN	Output	DBG	Not affected	Clamp to 0
AHB signals	Input	SOC	Not clamped	Not affected
BUSWIDTH	Input	SOC	Not clamped	Not affected
BUSENDIAN	Input	SOC	Not clamped	Not affected
APB signals	I/O	DBG	Signals are not clamped but accesses to registers in AHB power domain return slave error.	Not clamped. Debug APB signals must be powered down at the same time as **nCSDBGPWRDN** is LOW.
ATB signals	I/O	DBG	Not affected	Not clamped. ATB must be powered down the same time as **nCSDBGPWRDN** is LOW.

Table A-6 HTM signal behavior on assertion of power down (continued)

Signal name	Type	Power domain	Behavior when nCSOCPWRDN = 0	Behavior when nCSDBGPWRDN = 0
SE	Input	NA	No connect. Dummy input for scan insertion	No connect. Dummy input for scan insertion
Internal signals from SoC domain to debug domain	Wire	Both	Clamp to 0	Clamp to 0
Internal signals from SoC domain to debug domain	Wire	Both	Clamp to 0	Clamp to 0

Note

The **HTMASICCTRL** output is not clamped. This enables the output to retain the output value even if a power down signal is asserted. If you want to connect this signal to a different power domain, you must add additional clamping logic.

A.6.3 HTM power-down behavior

When **nCDBGPWRDN** is asserted (logic 0), trace operation stops because the Global Enable from the HTMGLBCTRL register, which is in debug domain, is forced to LOW by clamping logic when passing from the debug domain to the SoC domain.

When **nCSOCPWRDN** is asserted (logic 0), or when **HTMHRESETn** is asserted (logic 0), the accesses to registers in the SoC domain are blocked. Attempts to access these registers result in a slave error response.

Table A-7 shows the registers that can be accessed (debug power domain) while **nCSOCPWRDN** is LOW or when **HTMHRESETn** is LOW.

Table A-7 Accessible HTM registers in power down

Name	Address offset	Description
HTMGLBCTRL	`0x000`	Global Enable Register
HTMSTATUS	`0x004`	Status Register
HTMCFGCODE	`0x008`	Configuration description of HTM
HTMCFGCODE2	`0x00C`	Configuration description of HTM
HTMATIDOUT	`0x400`	ATB ID Output Value Register

Table A-7 Accessible HTM registers in power down (continued)

Name	Address offset	Description
HTMITATBCTR0-2	`0xEF0-0xEF8`	ATB Control Integration Test Registers
HTMITATBDATA0	`0xEEC`	ATB Data Integration Test Register
HTMITCR	`0xF00`	Test Control Register
HTMCLAIMTAGSET	`0xFA0`	Claim Tag Set Register
HTMCLAIMTAGCLR	`0xFA4`	Claim Tag Clear Register
HTMLOCK_ACCESS	`0xFB0`	Lock Access Register (for unlocking or locking register accesses)
HTMLOCK_STATUS	`0xFB4`	Lock Status Register
HTMDEVID	`0xFC8`	Device ID Register
HTMDEV_TYPE	`0xFCC`	Device Type Register
HTMPERIPHID4-7	`0xFD0-0xFDC`	Peripheral ID Registers
HTMPERIPHID0-3	`0xFE0-0xFEC`	Peripheral ID Registers
HTMCOMPONID0-3	`0xFF0-0xFFC`	Component ID Registers

Note

All other registers not listed in Table A-7 on page A-12 are blocked when the SoC domain is powered down, or when **HTMHRESETn** is active.

Appendix B
Troubleshooting

This appendix describes how to troubleshoot the HTM. It contains the section:

B.1 Troubleshooting the HTM

Table B-1 lists typical problems and the suggested remedies.

Table B-1 HTM typical problems and suggested remedies

Problem	Suggested remedy
Expected trace data not available.	Check if TrustZone settings are preventing the transactions from being traced.
Problem accessing the registers.	Check that AHB system is not in reset mode and **PCLKENDBG** is connected correctly.
	Make sure the lock register is unlocked.
No trace output data.	• Problem in configuration. To enable trace, as a minimum, you must: — Set the PROG bit in the HTMCONTROL for configuration. — Set the HTMGLBCTRL register. — Program the HTMTRACEEVT register. — Program the HTMTRACECTRL register. — Program the HTMADDR*x* and HTMADDRTYPE*x* registers (unless you set EXC_ONLY in the HTMTRACECTRL register to trace all transfers). — Program the HTMCONTROL register to enable packets and clear the PROG bit. Additional programming step would be required if counter/sequencer/trace start-stop resources are used. • External trace disabling or power down control is activated, for example **HTMTRACEDISABLE**. • The transfer is not traced because of a TrustZone violation. Check the HTMSTATUS register for the status of the TrustZone signals. • Integration test control registers in ATB components such as HTM, ATB funnel, ATB replicator, or TPIU might have been set. • If an external AHB multiplexer is used, ensure the multiplexer control is programmed correctly. • If an address range comparator is used, make sure both of the associated HTMADDRTYPE*x* registers are programmed for the same configuration.

Table B-1 HTM typical problems and suggested remedies (continued)

Problem	Suggested remedy
I am getting lots of overflow packet from the HTM trace stream.	• Use address filtering to reduce the amount of trace information generated. • Reprogram the HTMCONTROL register to limit the packet type. • Increase the trace clock frequency and bus width on TPIU. • If the trace port is shared between multiple trace sources, try disabling some other trace sources to see if the situation improves. • If possible, increase the ATB clock frequency.
Rubbish data appears in ATB when I power down the HTM.	Clear the Global Enable bit, GLBEN, in the HTMGLBCTRL register before power down.
Expected trigger does not take place.	Check that the trigger block is configured as single trigger mode and HTMTRIGSTATE is 1. This stops further triggers from happening.
Sequencer does not enter the state I requested.	• If two state transition trigger events take place at same time, the sequencer ignores both of them. For example, if current state is 1, and both state-1-to-state-2 event and state-1-to-state-3 event happen at the same time, the sequencer stays at state 1. • Another possible problem is that the sequencer enters a different state shortly after the expected transition. This depends on the setup of the sequencer. For example, if both state-1-to-state-2 event and state-2-to-state-3 event are configured to same event, and if the event lasts more than 1 clock cycle, the sequencer might switch from state 1 to 2, and then switch to state 3 in the following cycle.
I have two successive transfers that must generate triggers, but I only receive one trigger packet.	The trigger generation logic works on the rising edge of the trigger event.If the two transfers are joined together, the trigger event might be joined together and so there is only one rising edge detected. This is expected behavior.
When HTMEXTOUT/HTMTRIGGER outputs are used to trigger another HTM, the second HTM does not output the trigger until a few cycles later.	The HTM external I/O signal and HTMTRIGGER output are registered. Therefore there is a one cycle delay in generation of HTMEXTOUT/HTMTRIGGER and one cycle delay on capturing of HTMEXTIN. If the trigger event is propagated through an *Embedded Cross Trigger* (ECT), the delay might be more because the CTI and CTM blocks of ECT might have to resynchronize the events when it propagates through different clock domain.

Table B-1 HTM typical problems and suggested remedies (continued)

Problem	Suggested remedy
In the deliverable I received I only have one HTM design. It seems to be the 64-bit version. Where is the 32-bit version?	The HTM source code is configurable. See the *CoreSight Design Kit Implementation and Integration Manual* for details about how to configure the design into a 32-bit version.
The read values for the HTMCFGCODE2 and HTMSTATUS registers are different from what I expected.	Check if the power down indication signals are asserted (LOW). If they are asserted, this clamps the external inputs to the HTMCFGCODE2 and HTMSTATUS registers and so causes the status bits of external inputs to be zero.

Appendix C
Revisions

This appendix describes the technical changes between released issues of this book.

Table C-1 Differences between issue D and issue E

Change	Location
HTMPERIPHID2 reset value updated.	Table 3-3 on page 3-6
Number of wait states clarified	Paragraph following Table 3-13 on page 3-17
Column added for Bit 0.	Table 3-14 on page 3-19
Revision field updated.	Table 3-52 on page 3-51

Glossary

This glossary describes some of the terms used in ARM manuals. Where terms can have several meanings, the meaning presented here is intended.

Advanced High-performance Bus (AHB)

The AMBA Advanced High-performance Bus system connects embedded processors such as an ARM core to high-performance peripherals, DMA controllers, on-chip memory, and interfaces. It is a high-speed, high-bandwidth bus that supports multi-master bus management to maximize system performance.

See also Advanced Microcontroller Bus Architecture and AHB-Lite.

Advanced Microcontroller Bus Architecture (AMBA)

AMBA is the ARM open standard for multi-master on-chip buses, capable of running with multiple masters and slaves. It is an on-chip bus specification that details a strategy for the interconnection and management of functional blocks that make up a System-on-Chip (SoC). It aids in the development of embedded processors with one or more CPUs or signal processors and multiple peripherals. AMBA complements a reusable design methodology by defining a common backbone for SoC modules. AHB conforms to this standard.

Advanced Peripheral Bus (APB)

The AMBA Advanced Peripheral Bus is a simpler bus protocol than AHB. It is designed for use with ancillary or general-purpose peripherals such as timers, interrupt controllers, UARTs, and I/O ports. Connection to the main system bus is through a system-to-peripheral bus bridge that helps to reduce system power consumption.

See also Advanced High-performance Bus.

AHB

See Advanced High-performance Bus.

AHB-Lite

AHB-Lite is a subset of the full AHB specification. It is intended for use in designs where only a single AHB master is used. This can be a simple single AHB master system or a multi-layer AHB system where there is only one AHB master on a layer.

Aligned

Aligned data items are stored so that their address is divisible by the highest power of two that divides their size. Aligned words and halfwords have addresses that are divisible by four and two respectively. The terms word-aligned and halfword-aligned therefore stipulate addresses that are divisible by four and two respectively.

AMBA

See Advanced Microcontroller Bus Architecture.

APB

See Advanced Peripheral Bus.

ATB bridge

A synchronous ATB bridge provides a register slice to facilitate timing closure through the addition of a pipeline stage. It also provides a unidirectional link between two synchronous ATB domains. An asynchronous ATB bridge provides a unidirectional link between two ATB domains with asynchronous clocks. It is intended to support connection of components with ATB ports residing in different clock domains.

Big-endian

Byte ordering scheme in which bytes of decreasing significance in a data word are stored at increasing addresses in memory.

See also Little-endian and Endianness.

Burst

A group of transfers to consecutive addresses. Because the addresses are consecutive, there is no requirement to supply an address for any of the transfers after the first one. This increases the speed at which the group of transfers can occur. Bursts over AHB buses are controlled using the **HBURST** signals to specify if transfers are single, four-beat, eight-beat, or 16-beat bursts, and to specify how the addresses are incremented.

Byte

An 8-bit data item.

Byte-invariant

In a byte-invariant system, the address of each byte of memory remains unchanged when switching between little-endian and big-endian operation. When a data item larger than a byte is loaded from or stored to memory, the bytes making up that data item are arranged into the correct order depending on the endianness of the memory access.

The ARM architecture supports byte-invariant systems in ARMv6 and later versions. When byte-invariant support is selected, unaligned halfword and word memory accesses are also supported. Multi-word accesses are expected to be word-aligned.

See also Word-invariant.

Byte lane strobe
An AHB signal, **HBSTRB**, that is used for unaligned or mixed-endian data accesses to determine which byte lanes are active in a transfer. One bit of **HBSTRB** corresponds to eight bits of the data bus.

Cross Trigger Interface (CTI)
Part of an Embedded Cross Trigger device. The CTI provides the interface between a core/ETM and the CTM within an ECT.

Cross Trigger Matrix (CTM)
The CTM combines the trigger requests generated from CTIs and broadcasts them to all CTIs as channel triggers within an Embedded Cross Trigger device.

CTI
See Cross Trigger Interface.

CTM
See Cross Trigger Matrix.

CoreSight
The infrastructure for monitoring, tracing, and debugging a complete system on chip.

Debugger
A debugging system that includes a program, used to detect, locate, and correct software faults, together with custom hardware that supports software debugging.

Doubleword
A 64-bit data item. The contents are taken as being an unsigned integer unless otherwise stated.

Doubleword-aligned
A data item having a memory address that is divisible by eight.

ECT
See Embedded Cross Trigger.

Embedded Cross Trigger (ECT)
The ECT is a modular component to support the interaction and synchronization of multiple triggering events with an SoC.

Embedded Trace Buffer
The ETB provides on-chip storage of trace data using a configurable sized RAM.

Embedded Trace Macrocell (ETM)
A hardware macrocell that, when connected to a processor core, outputs instruction and data trace information on a trace port. The ETM provides processor driven trace through a trace port compliant to the ATB protocol.

Endianness Byte ordering. The scheme that determines the order in which successive bytes of a data word are stored in memory. An aspect of the system's memory mapping.

See also Little-endian and Big-endian

ETB *See* Embedded Trace Buffer.

ETM *See* Embedded Trace Macrocell.

Halfword A 16-bit data item.

IEM *See* Intelligent Energy Management.

Implementation-specific

Means that the behavior is not architecturally defined, and does not have to be documented by individual implementations. Used when there are a number of implementation options available and the option chosen does not affect software compatibility.

Imprecise tracing A filtering configuration where instruction or data tracing can start or finish earlier or later than expected. Most cases cause tracing to start or finish later than expected.

For example, if **TraceEnable** is configured to use a counter so that tracing begins after the fourth write to a location in memory, the instruction that caused the fourth write is not traced, although subsequent instructions are. This is because the use of a counter in the **TraceEnable** configuration always results in imprecise tracing.

Intelligent Energy Management (IEM)

A technology that enables dynamic voltage scaling and clock frequency variation to be used to reduce power consumption in a device.

LE Little endian view of memory in both byte-invariant and word-invariant systems. See also Byte-invariant, Word-invariant.

Little-endian Byte ordering scheme in which bytes of increasing significance in a data word are stored at increasing addresses in memory.

See also Big-endian and Endianness.

Macrocell A complex logic block with a defined interface and behavior. A typical VLSI system comprises several macrocells (such as a processor, an ETM, and a memory block) plus application-specific logic.

Multi master An AMBA bus sharing scheme (not in AMBA Lite) where different masters can gain a bus lock (Grant) to access the bus in an interleaved fashion.

Processor A processor is the circuitry in a computer system required to process data using the computer instructions. It is an abbreviation of microprocessor. A clock source, power supplies, and main memory are also required to create a minimum complete working computer system.

Replicator A replicator enables two trace sinks to be wired together and to operate independently on the same incoming trace stream. The input trace stream is output onto two (independent) ATB ports.

Reserved A field in a control register or instruction format is reserved if the field is to be defined by the implementation, or produces Unpredictable results if the contents of the field are not zero. These fields are reserved for use in future extensions of the architecture or are implementation-specific. All reserved bits not used by the implementation must be written as 0 and read as 0.

SBO *See* Should Be One.

SBZ *See* Should Be Zero.

SBZP *See* Should Be Zero or Preserved.

Should Be One (SBO)
Should be written as 1 (or all 1s for bit fields) by software. Writing a 0 produces Unpredictable results.

Should Be Zero (SBZ)
Should be written as 0 (or all 0s for bit fields) by software. Writing a 1 produces Unpredictable results.

Should Be Zero or Preserved (SBZP)
Should be written as 0 (or all 0s for bit fields) by software, or preserved by writing the same value back that has been previously read from the same field on the same processor.

TPIU *See* Trace Port Interface Unit.

Trace funnel A device that combines multiple trace sources onto a single bus.

Trace hardware A term for a device that contains an Embedded Trace Macrocell.

Trace port A port on a device, such as a processor or ASIC, used to output trace information.

Trace Port Interface Unit (TPIU)
The TPIU is used to drain trace data and acts as a bridge between the on-chip trace data and the data stream captured by a TPA.

Unaligned A data item stored at an address that is not divisible by the number of bytes that defines the data size is said to be unaligned. For example, a word stored at an address that is not divisible by four.

Unpredictable
For reads, the data returned from the location can have any value. For writes, writing to the location causes unpredictable behavior, or an unpredictable change in device configuration. Unpredictable instructions must not halt or hang the processor, or any part of the system.

Word
A 32-bit data item.

Word-invariant
In a word-invariant system, the address of each byte of memory changes when switching between little-endian and big-endian operation, in such a way that the byte with address A in one endianness has address A EOR 3 in the other endianness. As a result, each aligned word of memory always consists of the same four bytes of memory in the same order, regardless of endianness. The change of endianness occurs because of the change to the byte addresses, not because the bytes are rearranged.The ARM architecture supports word-invariant systems in ARMv3 and later versions. When word-invariant support is selected, the behavior of load or store instructions that are given unaligned addresses is instruction-specific, and is in general not the expected behavior for an unaligned access. It is recommended that word-invariant systems should use the endianness that produces the desired byte addresses at all times, apart possibly from very early in their reset handlers before they have set up the endianness, and that this early part of the reset handler should use only aligned word memory accesses.

See also Byte-invariant.