Intel® Trace Hub

Developer's Manual

March 2015

Revision 1.0

Notice: This document contains information on products in the design phase of development. The information here is subject to change without notice. Do not finalize a design with this information.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

Intel software products are copyrighted by and shall remain the property of Intel Corporation. Use, duplication, or disclosure is subject to restrictions stated in Intel's Software License Agreement, or in the case of software delivered to the government, in accordance with the software license agreement as defined in FAR 52.227-7013.

The Intel® Trace Hub may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

The code names presented in this document are only for use by Intel to identify products, technologies, or services in development that have not been made commercially available to the public, i.e., announced, launched, or shipped. They are not "commercial" names for products or services and are not intended to function as trademarks.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature may be obtained by calling 1-800-548-4725 or by visiting Intel's website at *http://www.intel.com/design/literature.htm*.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See *http://www.intel.com/products/processor_number* for details.

Intel, The Intel® Trace Hub and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries. MIPI® service marks and logo marks are owned by MIPI Alliance, Inc. and any use of such marks by Intel Corporation is under license.

The MIPI specifications were created by the Mobile Industry Processor Interface (MIPI) Alliance.

Contents

Figures

Tables

Revision History

Revision	Description	Date
1.0	Initial release.	January 2015

§

1 Introduction

The Intel® Trace Hub hardware is a set of functional blocks with the ability to perform full system debugging. That is, the Intel Trace Hub is not intended for hardware only or software only, but for full *system* debug. Specifically, it allows for testing the interaction of hardware and software as they produce complex system behaviors. Because the Intel® Trace Hub is intended for use by third-party debug tools, it is purposely designed and aligned with industry-standard debug methods and tools.

To accomplish these goals, the Intel® Trace Hub defines new features and leverages existing debug technologies to provide a complete framework for hardware/software co-debug, software development and tuning, and overall system performance optimization. Moreover, the Intel® Trace Hub ensures a consistent programming model by working as a Peripheral Component Interconnect (PCI) device that is visible to software.

The Intel® Trace Hub is composed of trace sources, a global hub, trace destinations, and a trigger unit. Trace sources include internal hardware signals from the Visualization of Internal Signals Architecture, performance data from the SoC Hardware and Performance counters, and software/firmware trace and debug data from the Software Trace Hub. Trace destinations include system memory, a MIPI* PTI port, and USB. The Global Trace Hub routes the data from the trace sources to the trace destinations according to the user's configuration (via software), and the Trigger Unit controls the starting and stopping of tracing operations. Finally, the Time Stamp Correlation Unit provides a timestamp to the trace sources, and also allows correlating of time-stamped data from the CPU's time domain to the Intel Trace Hub time domain.

1.1 Terminology

Term	Description
ART	Always-Running Timer
BPB	Byte-Packing-Buffer
Channel	STPv2 Channel: The lower-level data source identifier in a hierarchical STP data stream. Each Master has an independent Channel space.
CSR	Registers in a functional block that control an operation, or report the status of an operation. This single term is used to refer to both control registers and status registers.
CTC	Common Timer Copy
CTS	Common Trigger Sequencer
DbC or DBC	The host mode of an xHCI controller.
DCI	Direct Connect Interface
DbC.Trace	The endpoint within the xHCI controller that transports trace data directly from the Intel Trace Hub to the USB port/connector

Term	Description
FIFO	First-In First-Out
GTH	Global Trace Hub
HAS	High-Level Architecture Specification
IA	Intel Architecture
IO	Input/Output
IP	Intellectual Property
Master	STPv2 Master: The higher-level data source identifier in a hierarchical STP data stream.
MMI	MSC Memory Interface
MMIO	Memory Mapped IO
MSC	Memory Storage Controller
MSI	Message Signaled Interrupt
MSU	Memory Storage Unit
MTB	MSC Trace Buffer
ODLA	On-Die Logic Analyzer
OS	Operating System
PCI	Peripheral Component Interconnect
PHY	Physical Layer
PTI	Parallel Trace Interface
RTIT	Real Time Instruction Tracing. An older version of Intel® Processor Trace.
SoC	System-on-Chip
SoCHAP	System-on-Chip Performance Counters
SSC	Spread-Spectrum-Clocking
STH	Software Trace Hub
STP	MIPI* System Trace Protocol
STPv2	MIPI System Trace Protocol, version 2
TSC	Timestamp Counter
TSCU	Time Stamp Correlation Unit
UC	Uncacheable: A type of memory that cannot be stored in a CPUs cache. UC memory is strongly ordered.
USWC	Uncacheable Speculative Write Combining: A type of memory that allows for multiple individual writes to be combined into a single burst write. USWC memory is weakly ordered.

Term	Description
VER	VIS Event Recognition
VIS	Visualization of Internal Signals

§

2 Feature Set

This chapter includes a brief list of the requirements and features of the Intel® Trace Hub architecture host interface and shared blocks.

2.1 Features

2.1.1 Software Trace Hub Features

- **Support for industry standard MIPI STPv2.1:** Support for latest industry standard tracing protocol developed by the MIPI Alliance and known as STPv2.1 (System Trace Protocol version 2.1) with up 2^{16} Masters and up to 2^{8} Channels per Master.
- **Software/firmware tracing through primary and sideband fabric**
- **Intel® Processor Trace (PT) support through primary fabric with bandwidth up to 2GB/s:** Support for Intel Processor Trace (PT) through primary fabric with bandwidth up to 2GB/s[1].
- **Uncacheable Speculative Write Combining (USWC) MMIO region for PT trace data**: Offers higher performance than Un-Cacheable (UC) memory, maintaining IA core performance for Intel Processor Trace (PT).
- **Un-Cacheable (UC) MMIO region for general-purpose software trace data:** Guarantees the strongly-ordered memory model for software/firmware tracing and debug.
- **Time stamping Support:** Supports time-stamping of incoming debug data via the Time Stamp Correlation Unit (TSCU), another component of the Intel Trace Hub architecture.
- **System Deadlock avoidance function**: STH supports a deadlock avoidance counter to eliminate input-to-out dependencies that could result in a deadlock on the primary fabric.
- **Supports up to 4 Match/Mask Events:** Built-in support for simultaneously detecting up to 4 configurable software events.

2.1.2 Intel Trace Hub PTI Port Features

- Compliant with MIPI-PTIv1
- Configurable width output data of 4 bits, 8 bits, and 16 bits.
- No explicit constraints on the trace clock speed as long as the required setup and hold times are met (see MIPI-PTIv1 specification).
- VIS-bypass mode support.

*[1] Processor data rates are estimated at 1-2 bits per clock per Core. 2b/clock/core * 4 cores * 2 GHz = 16 Gb/s = 2 GB/s*

2.2 Trace Sources

- Software/firmware tracing through the primary and sideband fabric with software instrumentation.
- Intel® Processor trace (Intel® PT) support through the primary fabric.
- Hardware event capturing through ODLA and the VIS network.
- Hardware event performance monitoring through SoCHAP and the VIS network.
- Support for non-VIS based hardware tracing including fabric trace hooks.

2.3 Trace Destinations

- Support on-chip trace collection to system memory.
- Support off-chip trace collection to capture hardware using the MIPI-PTIv1 port.
- Support off-chip trace collection through USB3 via the Direct Connect Interface (DCI) technology with two supported modes (BSSB push mode and USB3 pull mode).

2.4 Primary Fabric Interface Features

- Provide PCI device presence
- Accessible from multiple root spaces.
- Support for single Message Signaled Interrupt (MSI).
- Implements multiple levels of error handling.

2.5 Sideband Fabric Interface Features

- Support for a single outstanding transaction at any given time.
- Provides alternate downstream access to memory mapped regions and configuration space registers.
- Accessible from multiple root spaces.
- Support for legacy INTx interrupt generation.

2.6 Intel Trace Hub TAP Interface Features

- Provides alternate access method to all Intel Trace Hub components including memory mapped CSRs and configuration space registers, as well as to embedded memory arrays for debug.
- Support for running VIS Controller from TAP clock if other clocks are still down.

2.7 Consistent Programming Model

- Seen by software running on the main IA CPU as a PCI device.
- Support for three 64-bit PCI Base Address Registers (BARs).
- Support for an additional 64-bit programmable BAR for firmware sources.

2.8 Error Handling and Survivability

- Support for primary fabric and PCI standard error handling and logging.

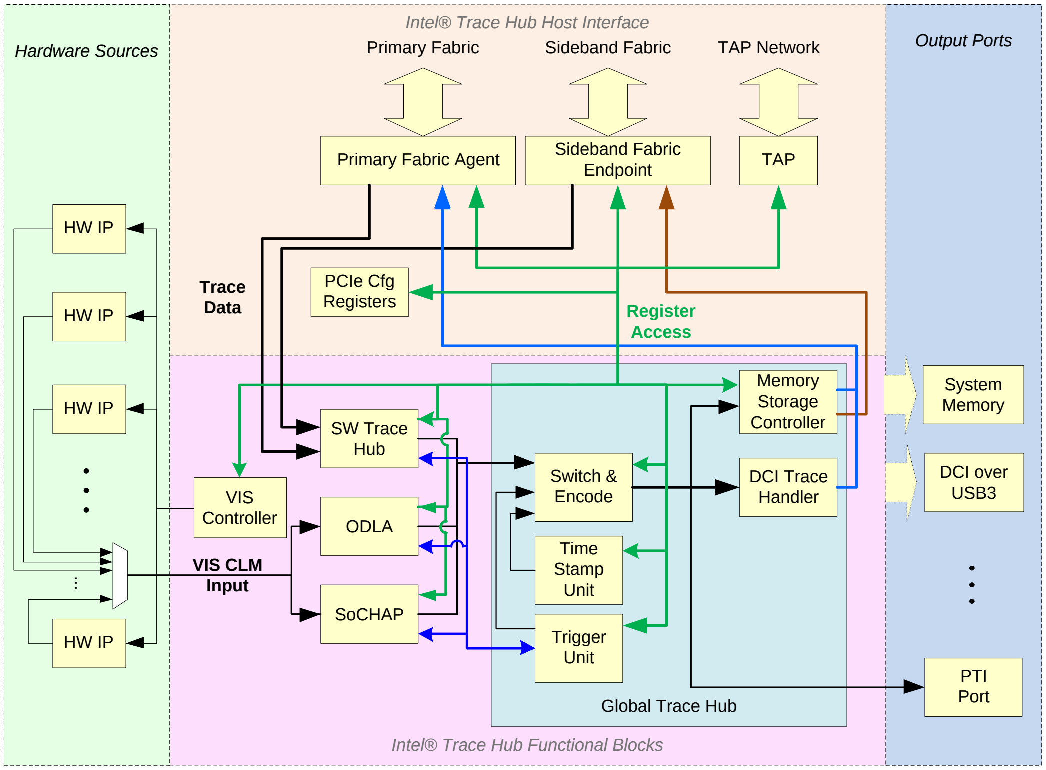
Support for software controlled functional reset.

3 Functional Description

3.1 Overview

The Intel® Trace Hub is comprised the blocks shown in Figure 1. Two other blocks not part of the Intel Trace Hub IP shown in Figure 1 include the hardware blocks/units feeding *trace data*[2] into the Intel Trace Hub through the Visualization of Internal Signals (VIS) network and other hardware trace data interfaces, hereafter simply referred to as *non-VIS trace sources*, and the group of output ports, which can include one or more destinations for the trace data. System memory is the only required destination while others are optional.

Figure 1: Intel® Trace Hub architecture high-level diagram



As shown in Figure 1, trace data is sourced from one or more sources. These sources include hardware events routed through the VIS network or other non-VIS trace sources. Trace data can also be software driven and delivered through one of the fabric interfaces from any agent that can master one or both of the interfaces, and driven into the STH. Hardware-based trace data routed through the VIS network is driven into the ODLA, VER,

[2] *In the context of the Intel Trace Hub architecture, trace data and debug data are used interchangeable to refer to the data collected from various hardware and software sources that enable an accurate reconstruction of system behavior for debug and performance optimization purposes.*

and the SoCHAP blocks for processing. The output of these blocks is then driven into the GTH which encodes the data into MIPI STPv2.1 and routes to one of the output trace destinations. One trace destination required for all Intel Trace Hub implementations is system memory, which is handled by the Memory Storage Unit (MSU) through the primary fabric. Another trace destination is the USB3 Debug Controller (DbC), which can stream data off-chip (through the USB3 port) and is managed by the Intel Trace Hub DCI Handler (ITH DCIH) and transferred through the primary and sideband fabric interfaces with maximum bandwidth of 450MB/s. Finally, other trace destinations can be instantiated to provide a direct off-chip path for trace data capturing using internal/external tools. Those destinations include the PTI port with a bandwidth up to 800MB/s (16 single ended lanes double pumped at 200MHz).

The Intel Trace Hub host interface supports a primary fabric interface, and a sideband fabric interface. It also supports a Test Access Port (TAP) for JTAG-based access to Intel Trace Hub components. In addition, the Intel Trace Hub host interface[3] contains some other units that are shared (or common) between all the functional blocks.

3.1.1 Intel® Trace Hub Functional Blocks

Referring to Figure 1, the Intel Trace Hub functional blocks include the Software Trace Hub (STH) block, the On-Die Logic Analyzer (ODLA)/VIS Event Recognition (VER) block, the System-on-Chip Performance Counters (SoCHAP) block, the Visualization of Internal Signals (VIS) Controller block, and the Global Trace Hub (GTH) block.

The GTH, being at the heart of the Intel Trace Hub functional blocks, is comprised of multiple sub-blocks including the Time Stamp Correlation Unit (TSCU), the Intel Trace Hub DCI Trace Handler block, the Memory Storage Unit (MSU) block, and the Common Trigger Sequencer (CTS)[4]. A brief overview of these functional blocks is presented in this section, with detailed descriptions being presented in each block's respective chapter.

The Visualization of Internal Signals (VIS) Controller block is responsible for programming/configuring the VIS network using a simple serial interface to route the various internal signals (that are later combined to form hardware events). Those signals can then be fed into the ODLA, VER, and the SoCHAP blocks through the Central Level Mux (CLM), for hardware debug and performance monitoring.

The On-Die Logic Analyzer (ODLA), when combined with VIS Even Recognition (VER) logic, is designed to capture hardware events using the Common Trigger Sequencer (CTS) for user-generated triggers. On the other hand, the SoCHAP architecture is designed to periodically sample certain hardware events and to capture useful information about system behavior and performance[5]. Those blocks are trace data sources to the GTH which collect

[3] Some design documentation will refer to the Intel Trace Hub Host Interface as the Common Blocks Cluster since it contains all the blocks shared between the different functional units.

[4] The Common Trigger Sequencer (CTS) is a component of the Trigger Unit (TU).

[5] The ODLA, the VIS Event Recognition, and SoCHAP architectures have been part of the debug architecture known as Lakemore. However, they have undergone multiple changes to fit into the Intel Trace Hub architecture framework.

data and encode it into the industry standard MIPI STPv2.1 (System Trace Protocol version 2)[6] before routing to one of the destination ports.

The STH is another source of trace data fed into the GTH. It is the means by which a device, often a CPU or micro-controller, can send debug data over the system fabric to the GTH, where it is collected, time stamped via the TSCU, encoded, and later sent to a trace destination (such as system memory, or a PTI port). Intel® PT data can be sent to the STH when Intel® PT is configured to specifically direct its output to Intel Trace Hub MMIO space. (See Chapter 4 for more information.) STH can also include regular (non-Intel® PT) software trace data, such as the message from a Linux kernel printk instruction or application printf instructions. All data destined for the STH can be sent over the primary fabric or the sideband fabric, according to bandwidth and the availability of the resources. The STH also provides software event triggering capabilities that are controlled by the CTS.

The Global Trace Hub (GTH) is responsible for collecting the data from the various sources mentioned above. Trigger sequences which enable and disable trace data collection and storage are generated in the GTH Trigger Unit. Once the GTH receives trace data, it is encoded into MIPI STPv2.1 protocol, and sent to a trace destination.

3.1.2 Intel Trace Hub Output Ports

The Intel Trace Hub architecture is designed to support multiple simultaneous trace destinations for the collected trace data. The only mandatory destination is system memory, which is reachable through the primary fabric interface. Other optional output ports include the USB3 port interface, and the Parallel Trace Interface (PTI) port.

The Intel Trace Hub Handler is the means by which trace data can be sent to the USB3 DbC and later off-chip through a USB3 port. The Intel® Trace Hub DCI Handler is responsible for providing all the services needed, including signaling the availability of trace data in the MSC Trace Buffer (MTB) to the USB3 DbC, and returning read completion data when requested by the USB3 DbC.

The MIPI Parallel Trace Interface (PTI) port is another standard interface defined by the MIPI Alliance which is composed of a double pumped parallel output port along with a strobe that is used primarily in mobile market segment to send trace data off-chip. For detailed information about interfacing with the PTI port, see Chapter 10 .

3.2 Intel Trace Hub MTB/CSR Memory Map

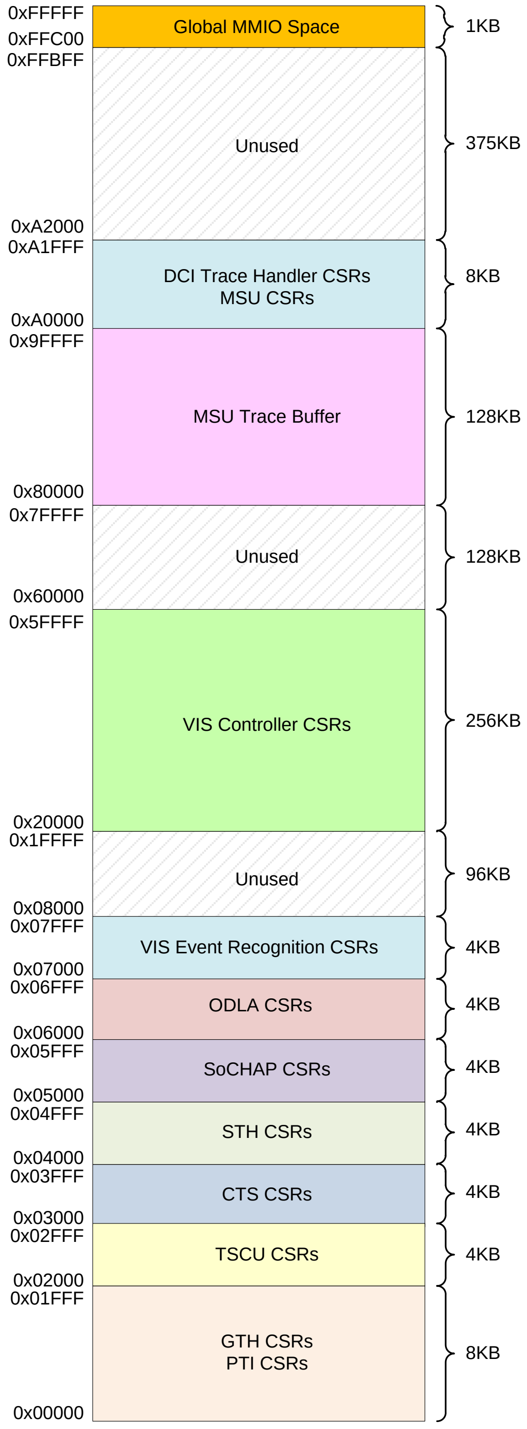
The CSR_MTB_BAR maps the configuration registers of Intel Trace Hub and the MTB to 1MB of MMIO space. The *local* base address values for each base address of Intel Trace Hub's functional blocks is given in Table 3-1 and a visual presentation is shown in Figure 2.

[6] *The MIPI STPv2 is the industry standard tracing protocol developed by the MIPI Alliance.*

Table 3-1: Intel® Trace Hub MMIO Offset/Size

Functional Block	Symbol	BASE_ADDR	Size
Global Trace Hub CSR PTI Port CSRs	GTH_CSR_OFFSET	0x00000	8KB
Time Stamp Correlation CSR	TSCU_CSR_OFFSET	0x02000	4KB
Comm. Trigger Sequencer CSRs	CTS_CSR_OFFSET	0x03000	4KB
Software Trace Hub CSRs	STH_CSR_OFFSET	0x04000	4KB
SoC CHAP Counters CSRs	CHAP_CSR_OFFSET	0x05000	4KB
On-Die Logic Analyzer	ODLA_CSR_OFFSET	0x06000	4KB
VIS Event Recognition CSR	VER_CSR_OFFSET	0x07000	4KB
VIS Register Controller CSRs	VRC_CSR_OFFSET	0x20000	256KB
MSC Trace Buffer MMIO MTB1 -> MTB_LOCAL_OFFSET MTB2 -> MTB1 + 0x8000	MTB_LOCAL_OFFSET	0x80000	128KB
Memory Storage Unit CSRs DCI Trace Handler CSRs	MSU_CSR_OFFSET	0xA0000	8KB

Figure 2: CSR_MTB_BAR MMIO Address Space Map

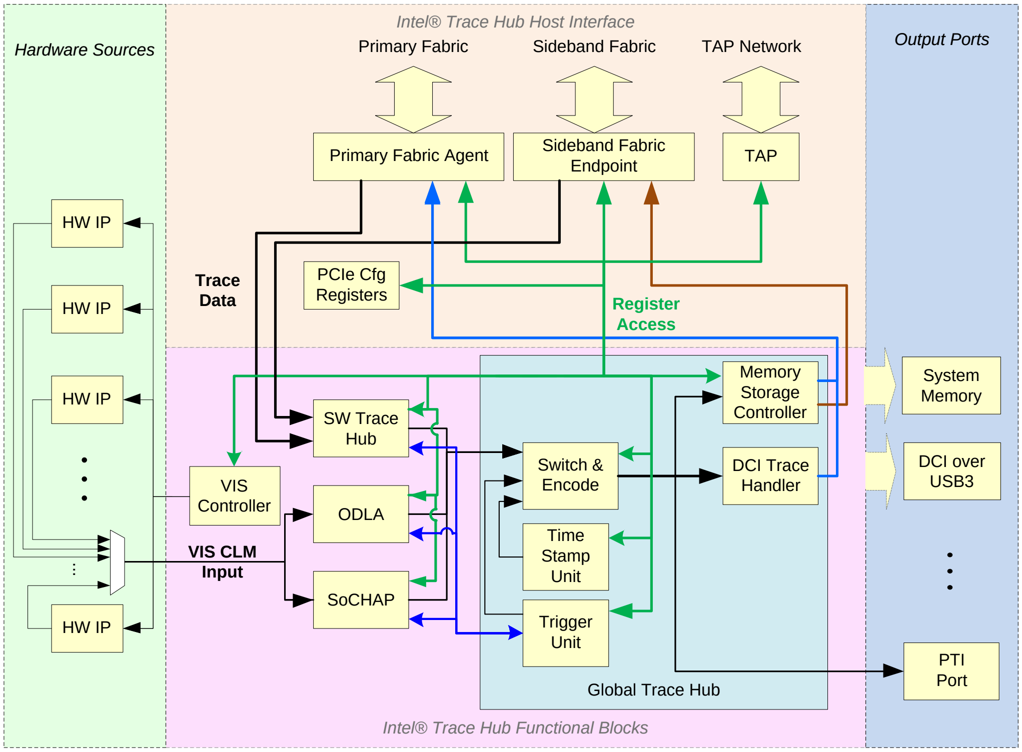


4 Software Trace Hub

The Software Trace Hub (STH) is one of the trace data sources feeding data into the Global Trace Hub (GTH) component of the Intel® Trace Hub architecture, as shown in Figure 3. It is the means by which a device, often a CPU or micro-controller, can send debug messages over the system fabric to the GTH to be collected, encoded, time stamped, and later sent to a trace destination. Trace destinations can be either off-chip (for example, a high-speed trace port) or on-chip (for example, system memory).

Debug messages can include Intel® Processor Trace (PT) data, which can be sent to the STH when Intel® PT hardware in the CPU is configured to direct its output to Intel Trace Hub MMIO space as explained elsewhere in this document. It can also include regular (non-PT) software and firmware trace data, such as the message from a Linux kernel `printk` instruction or application `printf` instructions. Hereafter, non-PT data will be referred to simply as *software trace data* or *regular software data*. Software messages are not limited to software running on the CPU core(s); they can also be sent from other microcontrollers running firmware, so long as they can master one of the interfaces to the STH (primary or sideband fabric).

Figure 3: Intel Trace Hub Architecture



The STH is accessible through both primary fabric and the sideband fabric interfaces, and is mapped to multiple spaces. However, despite being visible and accessible through different interface/spaces, the STH presents a single memory map, identical to all interfaces. This allows full access to all STH capabilities independent of the source or path the transaction took to reach the STH MMIO space. The STH is capable of high data transfer rates (up to 2GB/s[7]) to support up to four PT data streams (from four CPUs at up to 2GHz[8]), plus software trace data.

The STH is seen by software as three independent memory-mapped regions configured through the Intel Trace Hub PCI device Base Address Registers (BARs). The first region is used by the STH Control and Status Registers (CSRs) and includes some global packet generation commands as explained later in this document. The two other memory regions (or targets) are used for software trace data, and for Intel® PT data, respectively. The Software Trace Memory Region (STMR) receives software trace data, which is then mapped to an internal protocol that can later be encoded by the GTH into STPv2.1 packets.

One important note is that, while the MIPI STPv2 specification supports data packets of widths 4, 8, 16, 32, and 64 bits, the Intel Trace Hub STH only supports widths of 8, 16, 32, and 64 bits (that is, it does not support 4-bit data packets). Further, to keep the logic simple, the STH will automatically round-up any data received that is not of the above widths to the next STPv2 supported size (for example, a 48-bit write will automatically be rounded up into a 64-bit dataset).

On the other hand, Intel® PT data is claimed by the Intel® PT Trace Memory Region (RTMR) controller (also referred to as the Intel® PT Trace data target), and later forwarded to the GTH input buffer for encoding and transmission.

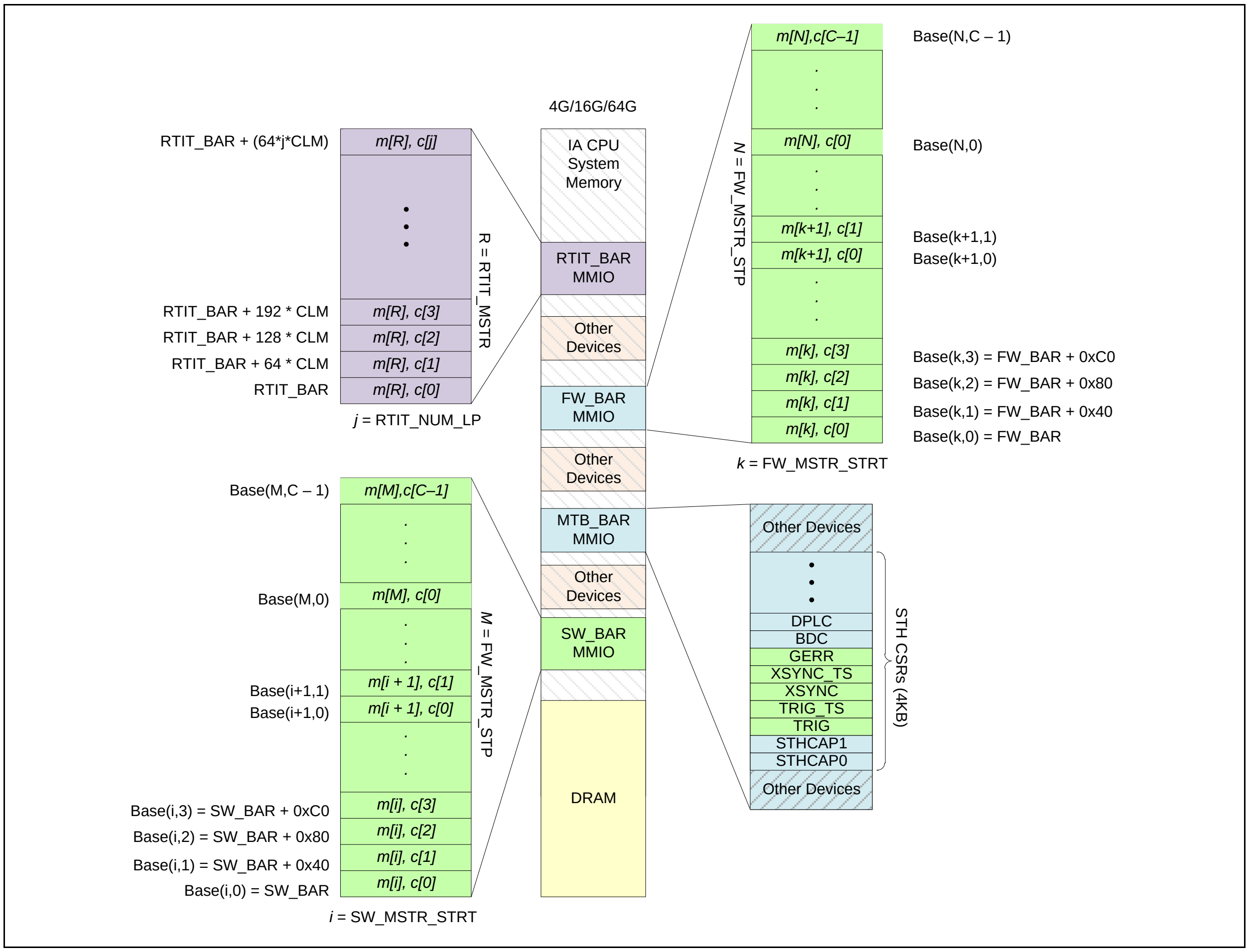
4.1 STH MMIO Regions

The STH appears to software in three MMIO regions in system memory. The STH has two dedicated MMIO regions while its configuration registers, including a few special registers, share a third MMIO region with the rest of Intel Trace Hub components. The architecture of those regions is the topic of this section with an illustrative diagram of how the STH MMIO regions appear in system memory given in Figure 4.

[7] *When operating at 250MHz or higher with a bus width of 64bits*

[8] *Intel® Processor Trace data rates are estimated at 1-2 bits per clock per Core. 2b/clock/core * 4 cores * 2GHz = 16Gb/s = 2GB/s*

Figure 4: Intel Trace Hub / STH MMIO Regions



4.1.1 Software Trace Memory Region (STMR)

4.1.1.1 General

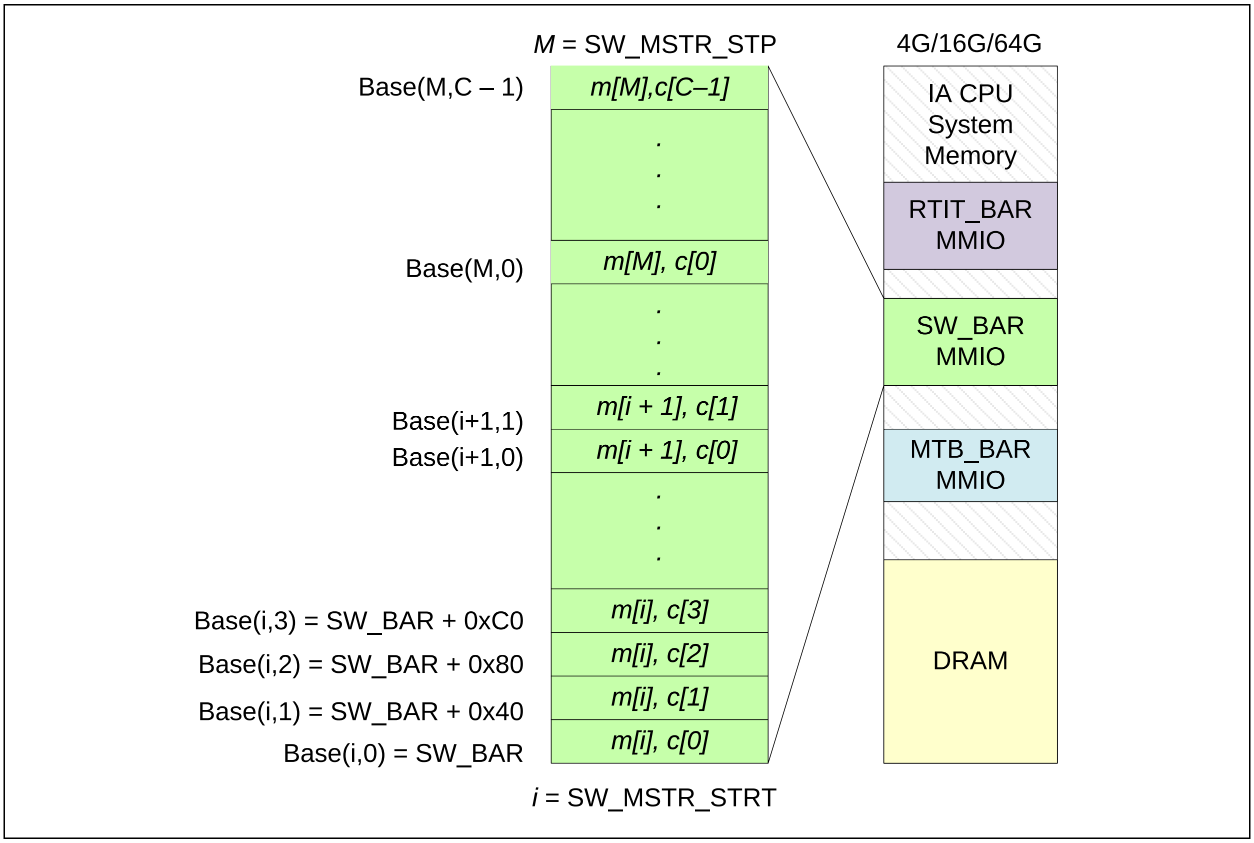
For software trace data, the Software Trace Memory Region (STMR) appears in system memory as determined by the SW_BAR PCI BAR (which is a concatenation of SW_UBAR and SW_LBAR). One important requirement for the STMR region is the necessity of mapping it to Un-Cacheable (UC) memory space because it is designed only for a strongly-ordered memory model.

Software can determine the number of masters in the STMR region using the STHCAP0 register. Assuming STHMNUM – STHMSTR number of masters will be allocated for software with SW_CHCNT channel per master, then, with 64B for each channel, the total size of STMR, in bytes, is given by the following formula:

$$Size(STMR) = 2^{\log_2 \lceil 64 \times (SW\_MSTR\_STP - SW\_MSTR\_STRT) \times SW\_CHCNT \rceil}$$

For example, an STMR implementation with 64 Masters and 128 Channels per Master would occupy 64*64*128=512KB of MMIO space.

Figure 5: Software Trace Memory Region Memory Map



As shown in Figure 5, the address to which (debug) data is written determines the Master and Channel assigned to the data. For example, if software writes to address SW_BAR, then it will be assigned to the first software Master number, denoted by the SW_MSTR_STRT parameter. The base address for master *m*, channel *c*, is given by the following formula:

$$Base(m - SW\_MSTR\_STRT, c) = 0x40 \times \left(SW\_CHCNT \times (m - SW\_MSTR\_STRT) + c\right) + SW\_BAR$$

Thus, given the following conditions:

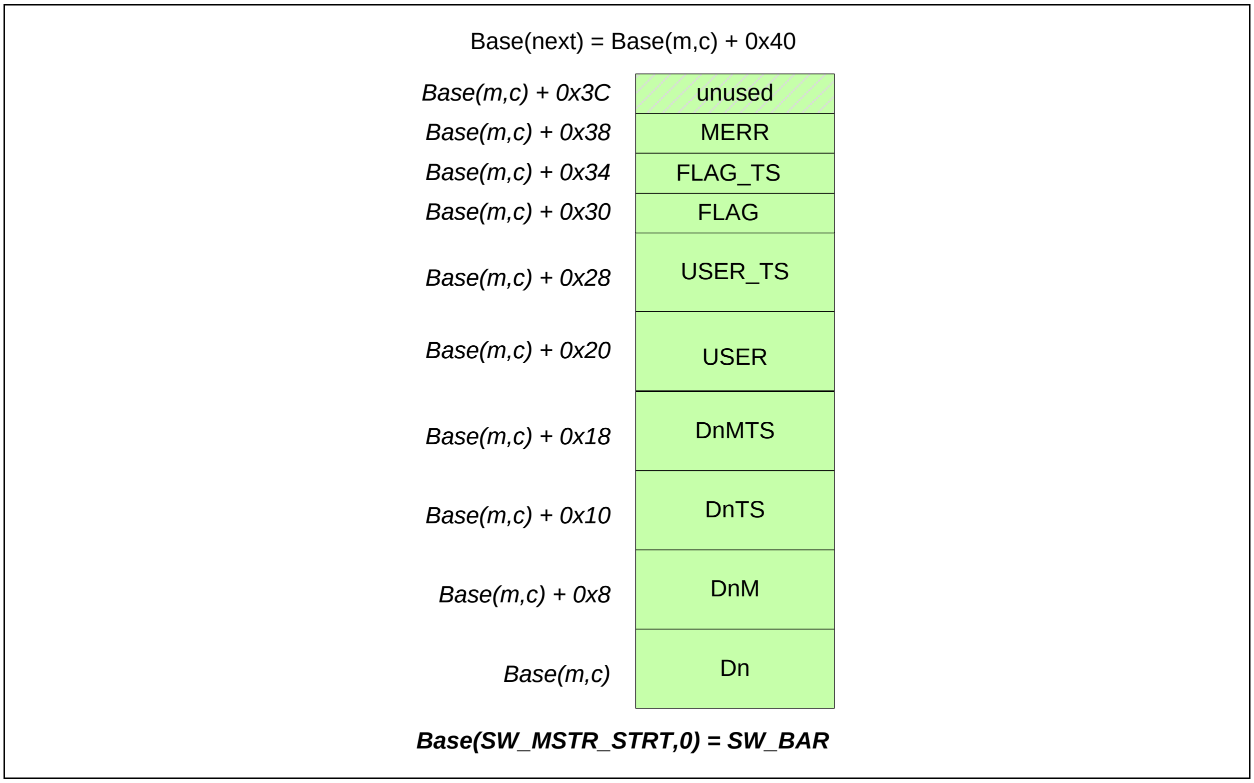
- STH supports
  - 512 Masters
  - 16 Channels per Master
  - Base address of 0x3800_0000
  - Starting master number of 16
- Master 34
- Channel 3

Then the base address for Master 34, Channel 3 is `0x380048C0`.

As another example, if each software master is allocated 128 channels, and if software wishes to write some debug message to Master SW_MSTR_STRT + 4, Channel 1, then it needs to issue a write request to the base address SW_BAR + 0x8040.

Within each Master/Channel MMIO region, the memory is presented as 64-bit wide because this is the maximum packet size that the MIPI STPv2.1 standard supports. Because there are a number of packet types that the STH can generate, and because software desires to control this specifically, a total of 64B is allotted for each Master/Channel. Each 64B region is divided into several areas, according to the type of MIPI STPv2 packet that will be generated when software writes to that address. The various packets and their corresponding addresses within each Master/Channel region are shown in Figure 6.

Figure 6: Software Master/Channel Memory Map Detail



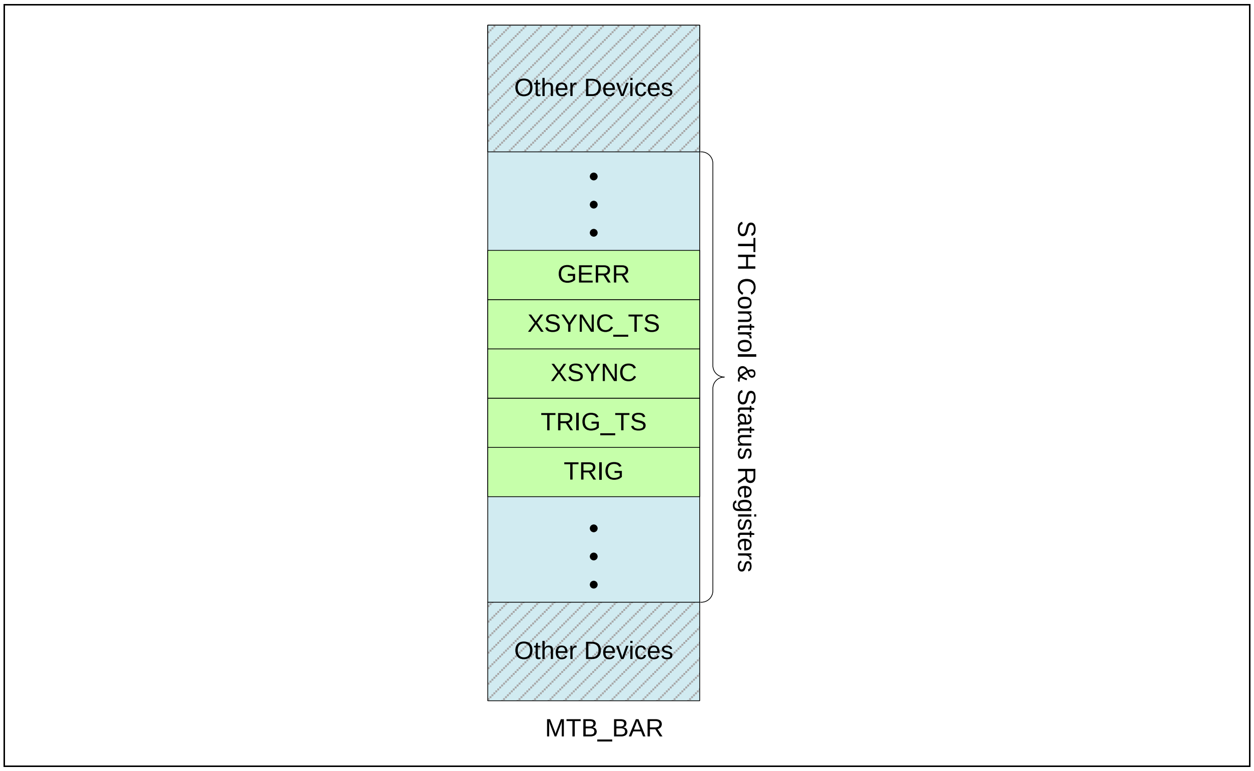
At the bottom of the region are the data packets: regular data, marked, time-stamped, and marked plus time-stamped. These are denoted in Figure 6 as Dn, DnM, DnTS, and DnMTS, respectively. The other packet types are above the data packet types: USER, USER_TS, FLAG, FLAG_TS, and MERR. Note that the MERR packet is a per-Master packet, not a per-Channel packet. That is, a MERR packet written to Master 0, Channel 0 will appear identical to a MERR packet written to any other Channel in that same Master.

4.1.1.2 Special Packet Generation

The STH has the ability to generate five *global* packet types: Trigger (TRIG), Trigger with Time Stamp (TRIG_TS), Cross-Synchronization (XSYNC), Cross-Synchronization with Time

Stamp (XSYNC_TS), and Global Error (GERR); see Figure 7. There is also one form of USER_TS packet which is considered a special, or global, packet, described later in this section. These packets are 'global' because they are not associated with any Master or Channel. As such, they are generated by a write to the identically-named STH CSR (TRIG, TRIG_TS, XSYNC, etc.). Since the MIPI STPv2 specification allows for 8-bits of payload with these packet types, the STH only sends the lowest-order 8-bits of data on a write to these 32-bit sized CSR locations.

Figure 7: Special Packet Generation CSRs Map



4.1.1.2.1 Special Packets Master/Channel Assignment

It is very important to note that all of the special (or global) packet types described in this section are *not* related or tied to any specific Master or Channel, including the TRIG and TRIG_TS packets. Because of this, all the packet types referred to as special or global packets are sent with the Master and Channel fields sets to zero. As such, they will be routed to the trace destination for Master 0 Channel 0 (specified by the GTH register SWDEST[0]).

4.1.1.2.2 Trigger Packets

As mentioned above, a trigger packet can be generated by writing to the TRIG or TRIG_TS CSR. The trigger with timestamp (TRIG_TS) dataset can also be generated by the STH controller in response to the Common Trigger Sequencer (CTS) indicating a trigger event (i.e. asserting its trigger signal). This can be due to several possible scenarios as explained in Section 4.4.

4.1.1.2.3 Cross Synchronization Packets

The cross synchronization packet can only be generated by writing to the XSYNC or XSYNC_TS CSR. In this revision of the STH specification, there are no use cases for the XSYNC or XSYNC_TS packets. While the STH will be able to generate these packets, the details of how and when these are generated are outside the scope of this specification.

4.1.1.2.4 Global Error Packets

Global Error (GERR) packets are created in response to a data drop condition as explained in Section 4.3, and also when requested by a software write to the GERR register. The definition and use of GERR payload, except for data loss condition, is outside the scope of this specification.

4.1.1.2.5 User Packets

There are two specific formats of USER_TS packets that are considered “special packets”: the start-of-data-loss packet, and the end-of-data-loss packet. These are generated when the STH enters its lossy operation mode and starts dropping data. In this case, USER_TS packets are used to indicate the start and end of the lossy operation period, so that reconstruction software can properly display the data. Like other special packets, these USER_TS packets are sent only to Master 0 Channel 0.

User-defined packets (USER) and user-defined packets with time stamp (USER_TS) are also created like other packets when software writes the appropriate address in the STMR MMIO region for the corresponding Master/Channel. These USER/USER_TS packets *are* associated with a particular master and channel. The format, interpretation, and use of USER and USER_TS packets other than for indicating the start and end of the data loss period is beyond the scope of this specification.

4.1.2 Firmware Trace Memory Region

4.1.2.1 General

The Firmware Trace Memory Region (FTMR) appears in system memory as determined by the FW_BAR PCI configuration space register (which is a concatenation of FW_UBAR and FW_LBAR). The term “firmware” (and FTMR) is used here to differentiate it from the STMR. “Firmware” trace sources are those devices, and their associated firmware (if any), that do not have PCI enumeration or discovery capabilities. As such, these devices need to be told, by some other device or process (e.g., system BIOS), where the FTMR is located in the system memory map. Once they are told the location of the FTMR, this location should not change. It is important to note that the FW_BAR is not located in a standard PCI BAR location. This, again, emphasizes the unusual nature of this MMIO region, in that it will not be scanned or enumerated like a standard PCI BAR. Thus, an operating system will not be able to relocate this BAR as it does other PCI memory regions.

Similar to the STMR, the FTMR must be located in Un-Cacheable (UC) memory space because it is designed for only a strongly-ordered memory model.

Software can determine the number of masters in the FTMR region using the STHCAP1 and STHCAP2 registers. Assuming FW_MSTR – FW_MSTR_STRT number of masters will be allocated for firmware, with SW_CHCNT channel per master, then, with 64B for each channel, the total size of FTMR, in bytes, is given by the following formula:

$$Size(FTMR) = 2^{\log_2\lceil 64\times(FW\_MSTR\_STP-FW\_MSTR\_STRT)\times SW\_CHCNT\rceil}$$

For example, an FTMR implementation with 32 Masters and 128 Channels per Master would occupy 64*32*128=256KB of MMIO space.

The FTMR is structured identically to the STMR in terms of addressing Masters/Channels, STP packet types within each channel, and so on. That is, Figure 5 and Figure 6 apply to the FTMR (substituting FW for SW as appropriate). Please refer to section 4.1.1.1 for more details.

4.1.3 Intel® PT Trace Memory Region

The STH optionally supports Intel Processor Trace (Intel® PT); Intel Processor Trace support can be determined in the STH_CAP1 register. Intel Processor Trace hardware uses a set of MSRs (specifically, the IA32_RTIT_* MSRs) to set the base address and size of the destination port for each Core/Thread. Correspondingly, the STH memory window for Intel® PT is configurable, using the RTIT_BAR PCI Base Address Registers. The CPU MSRs and the RTIT_BAR must be set to the same starting address in order for the STH to accurately capture the Intel® PT trace data.

The STH's Intel® PT Trace Memory Region (RTMR) is shown in Figure 8. The RTMR must be mapped into Uncacheable Speculative Write Combining (USWC) memory space, as the RTMR is designed specifically for this type of memory, matching the high-performance implementation of the IA Core's Intel® PT hardware, which was designed to specifically utilize the CPUs write-combining buffers and USWC memory space.

Similar to the STMR, the RTMR is designed to accommodate a variable number of masters and channels. A channel is assigned to each logical processor (where a logical processor is a thread for a multi-threaded core, or a core for a single-threaded Atom® processor), starting at channel 1 (the STP Channel 0 in the Intel PT Master is not used). Assuming *NUM_LP* Intel® PT logical processors will be implemented, and by allocating CLLP cachelines for each Intel® PT logical processor, the total size of RTMR is given by the following formula in bytes:

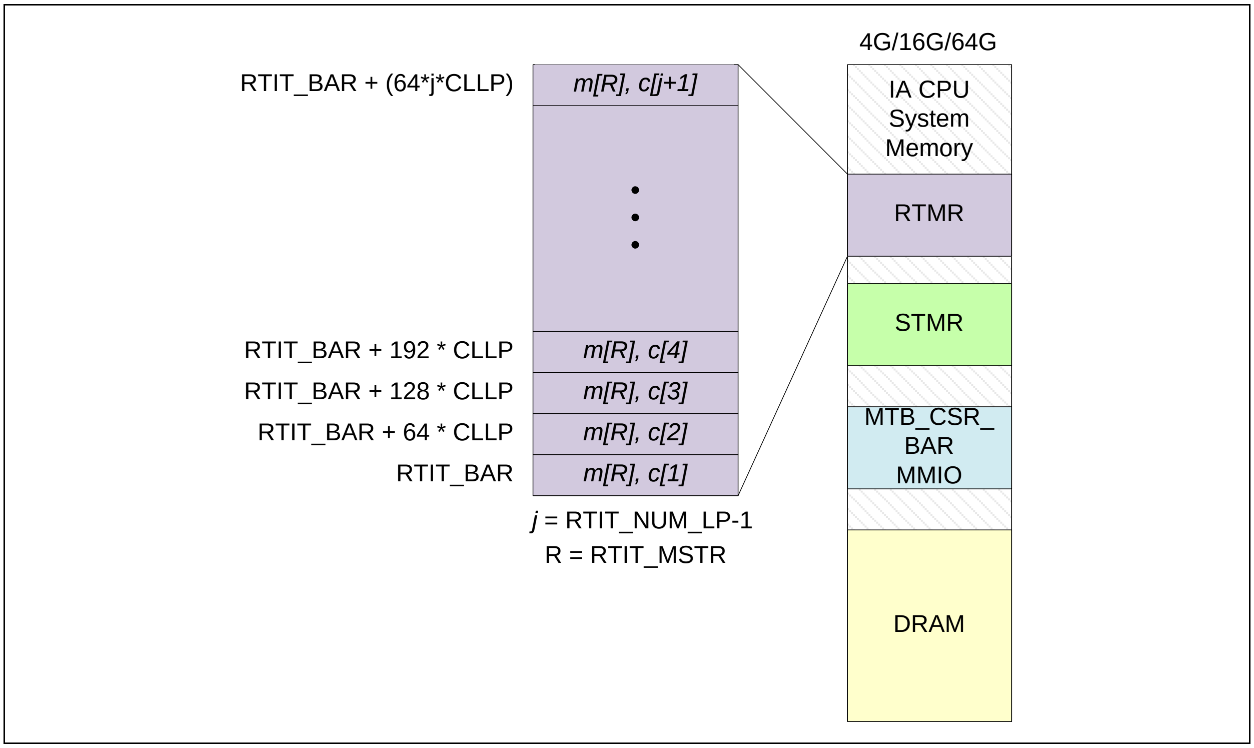
$$Size(RTMR) = 2^{\log_2\lceil 64*NUM\_LP*CLLP\rceil}$$

For example, an RTMR implementation with 4 Intel® PT sources (threads) and 4 cache lines per Intel® PT logical processor, would occupy 64*4*4=1KB of MMIO space.

From Figure 8, it can be shown that the base address for each Intel® PT logical processor is given by the following formula:

$$Base(RTIT\_MSTR\_STRT\ + m) = 64 \times m + CLLP + RTIT\_BAR$$

Figure 8: Intel® PT Trace Memory Region Memory Map



4.1.3.1 Intel Processor Trace support

Unlike the STMR and FTMR, Intel Processor Trace support in the Intel Trace Hub implements only “plain” STPv2 data packets (data packets marked and time stamped are not needed). That is, any write to a location within the RTMR will result in an STPv2 data (neither marked nor timestamped) packet

While technically possible, it is not advisable to send Intel PT data to DRAM through the Intel® Trace Hub. From a system perspective, if PT data needs to be sent to DRAM, the CPU can directly send it to DRAM without involving the Intel® Trace Hub.

4.1.3.2 Re-ordering

Because the CPU can send Intel® PT data to the Intel Trace Hub out-of-order, a protocol has been created for Intel Trace Hub which provides the ordering information to downstream decoding software, eliminating the need for hardware buffering in the Intel Trace Hub. Thus, the STH will send all PT writes straight through to the destination in the order they were received from the CPU—whether in-order or out-of-order.

To enable the decoding software’s ability to properly re-order the resultant PT data, a header is prepended to the PT data to aid software in doing the re-ordering. The header is only sent when it is needed: for the first PT write in an in-order stream or collection, and for each out-of-order write. Headers are not sent for in-order data. Data is expected to be in-order most of the time, so this makes for the most efficient protocol possible, exceeding 94% efficiency (including STPv2 encoding).

The Intel PT header is an STP D32M packet, with data payload as shown in Table 4-1.

Table 4-1: Software Re-Ordered Intel Processor Trace Header Definition

Bit(s)	Field	Description
31	Start	Start of PT Packet sequence. Indicates PT packet forwarding in the Intel® Trace Hub was just turned on.
30	Tag	Tag bit. This bit is toggled for each set of cache lines received by STH. In effect, it becomes an address bit, or sequence number, useful to reconstruction software.
29	Rsvd	Reserved
28	cls	Cache Line Size. 0 = 64B 1= reserved
27:23	CLLP	Cache Lines per logical processor. 0 = 1 cache line 1 = 2 cache lines … 1F = 32 cache lines
22:14	len	Number of valid PT bytes in this group of packets. Encoding is len – 1. Values 0 – 0x1FF correspond to 1 to 512 bytes.
13:0	address	Starting address (byte address) of valid data. 14 bits enables ability to distinguish 32 cache lines, 512B per cache line.

The re-ordering protocol can be distilled down to a set of requirements that govern how Intel Trace Hub sends PT data out-of-order. These are enumerated in the list below. All requirements are per-logical-processor unless otherwise specified. See section 4.1.3.2.1 for definitions of terms used.

A PT packet set is a set or group of STPv2 packets that, together, comprise a single PT write received by Intel Trace Hub on the primary fabric bus. Because Intel Trace Hub operates on only 64 bits of data at any given time, but PT data can be as much as 64 bytes, Intel Trace Hub will split the single PT write into multiple STPv2 packets.

1) The first PT packet set transmitted by the Intel Trace Hub will have a header.
2) The first PT packet set transmitted by the Intel Trace Hub will have a start bit set.
3) All header packets will contain encoded values indicating cache line size (CLS) and number of cache lines per processor/thread (CLLP).
4) All header packets will include the number of bytes, and the starting address within the cache line for the first byte.
    a. For full cache lines, the number of bytes is equal to the cache line size (CLS)

5) All out-of-order and partial-cache-line packet sets transmitted by the Intel Trace Hub will have a header.
6) The Intel Trace Hub will include a “tag” bit in the header, identifying which open bucket a cache line belongs to. The tag bit will start at zero, toggle to one, then back to zero, and so on.
7) The determination of whether a packet is in-order or out-of-order is based solely on the starting address of the previous PT write set. That is, if, for CLLP = 4, cache lines are received in the following order: 1, 3, 2, 4, then headers will be generated for all four cache lines, as they are all out-of-order with respect to the previous write set. To further illustrate this point, if the cache lines are written in the following order: 1, 2, 4, 3, then headers are generated for cache lines 1 (the starting cache line), 4, and 3; cache line 2 is the only cache line that is in-order with respect to the previous write set.
8) The header packet will not be delivered for any in-order full cache line. That is, any 64 byte IPT write to Intel Trace Hub that follows, in-order, either a full or partial cache line, will not have a header.
    a. The partial cache line write preceding such an in-order full cache line write must complete that partial cache line. That is, if bytes 0…n of cache line M were received by the Intel Trace Hub already, the only way cache line M+1 can be transmitted without a header packet is if the write immediately preceding cache line M+1 contains bytes n+1 … 63 of cache line M.
9) Intel Trace Hub STH will optimize the size of STPv2 data packet whenever possible.

4.1.3.2.1 Intel Processor Trace Re-Ordering Definitions

Terms:

1) A cache line is the fundamental size of data in Intel® PT.
2) “Bucket” refers to the number of cache lines for a logical-processor. It is the same as CLLP parameter.
3) Only *one* out-of-order window for a logical processor can be open at any given time. An out-of-order window is two buckets. That is, a maximum of two buckets can be “open” at any given time when re-ordering data.
4) A PT packet set is a set or group of STPv2 packets that, together, comprise a single PT write received by Intel Trace Hub on the primary fabric. Because Intel Trace Hub operates on only 64 bits of data at any given time, but PT data can be as much as 64 bytes, Intel Trace Hub will split the single PT write into multiple STPv2 packets.
5) A trace window is a consecutive series of data, made up of more than zero buckets. A trace window could be as small as one byte, or as large as several gigabytes. A trace window is bounded by the STH storenEn=0→1 and 1→0.
6) A trace buffer is a collection of data that can be as “small” as a partial trace window (e.g., attempting to store a large trace in a small ring buffer) to as “large” as several trace windows.
7) In-order refers to data bytes written to consecutive byte addresses (modulo CLLP). In-order scope only applies to transaction n and n+1. That is, transaction n+1 in-order-ness is determined solely by the address of transaction n. The following cache lines numbers are all considered in-order (CLLP = 4), even though they span multiple buckets: 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3, 0
8) Out-of-order refers to data bytes written to Intel Trace Hub that occupy non-consecutive byte addresses (modulo CLLP). The following cache line write examples (all of which are valid examples) have at least one out-of-order cache line. Each number represents a cache line (numbered 1 to 4), and CLLP = 4.

1, 3, 2, 4

1, 2, 4, 3

1, 2, 3, 4, 4

1, 1, 2, 3, 4

1, 2, 3, 2, 4

4.1.3.3 Processor Trace and Lossy Operation

The Intel Processor Trace protocol is designed to accommodate unreliable transmission of data to its destination. In order for the trace data to be reconstructed, the loss must be sensed, and accounted for, at the CPU itself. If data is dropped after being sent from the CPU, but before arriving at the final destination, then the PT trace will not be reconstructible; in other words, it will be useless.

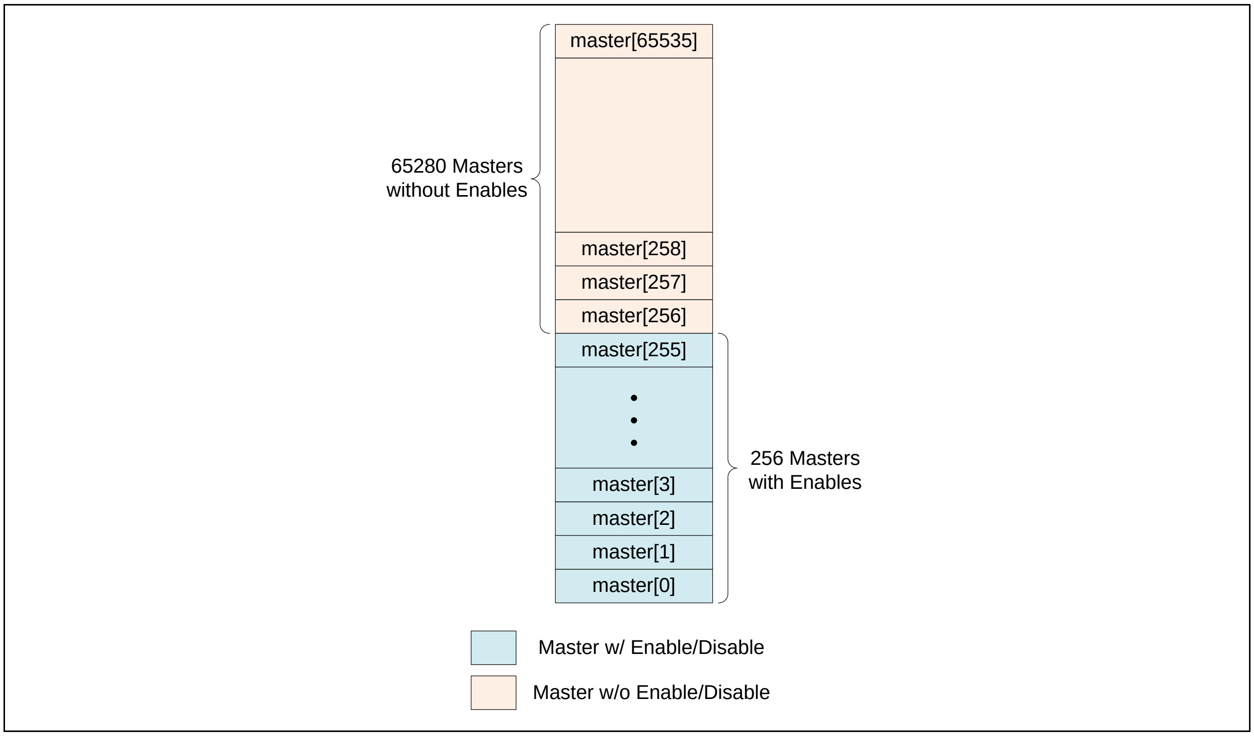
In support of this functionality, the STH BDC can be completely disabled (see the GTHMISC register), so that the STH becomes lossless. In this configuration, the STH (and, therefore, Intel Trace Hub) has the potential to negatively impact system performance because it will assert backpressure to the primary fabric if Intel PT data cannot be accepted.

4.2 Master Number Assignment and Allocation

Intel Trace Hub has two kinds of Masters (Master, in this context, refers to a MIPI STPv2 Master): those with a dedicated enable bit (a per-master enable/disable), and those without. The purpose of a Master enable bit is to allow a master to send data to the GTH, while giving some other software (e.g., debug tool software – the user) the ability to control whether that data is sent to the destination or dropped. The primary use case for this capability is for “firmware” masters: these devices typically have very limited code space, and strongly desire that the execution path not change, whether Intel Trace Hub debug messages are being sent or not. These firmware Masters will always send their debug data to Intel Trace Hub. If the debugger desires to not see that debug information, simply disable the relevant Masters.

The number of Masters with a per-master enable is limited to the lower 256 (as shown in Figure 9) in order to minimize the STH physical size, while still providing sufficient per-master-enable capability.

Figure 9 STH Master Allocation

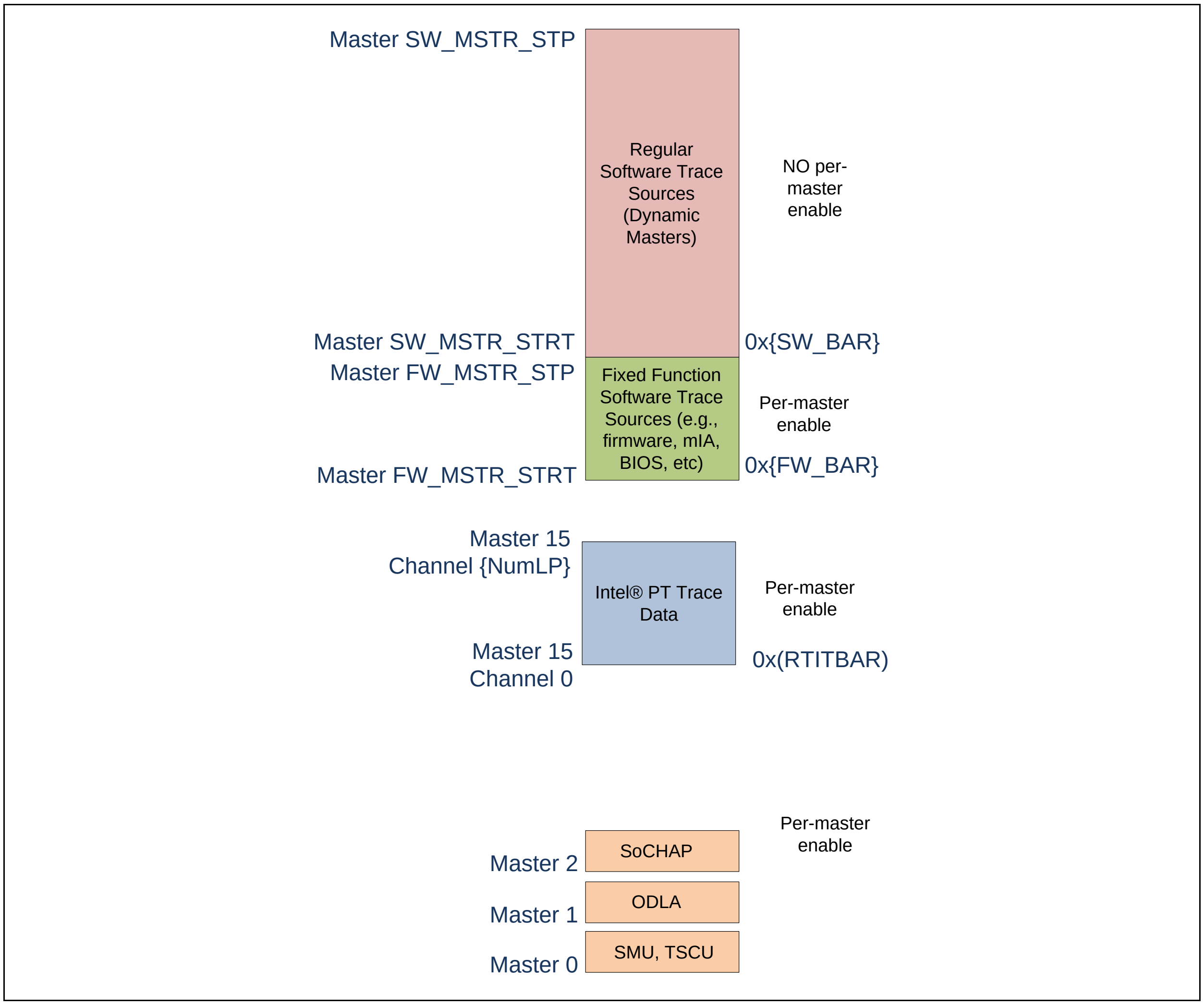


However, to enable different projects' ability to specify which sources use the Masters that implement an enable/disable control, and those that do not, the STH contains three registers that can be used to map the software and firmware Masters to a subset of the available Masters. These registers are contained within STHCAP0/1/2.

Table 4-2: STH Master Allocation Registers

Register	Description
FW_MSTR_STRT	**Firmware Master Start:** This parameter specifies the Master number of the first Firmware source. Firmware masters are those that are accessible via the FW_BAR memory window.
FW_MSTR_STP	**Firmware Master Stop:** This parameter specifies the highest Master number allocated to "firmware" devices. This number should be less than or equal to 255, as firmware masters need the per-master enable capability of Intel Trace Hub. The difference between SW_FW_MSTR_STRT and FW_MSTR_STP gives the number of Masters allocated to firmware.
SW_MSTR_STRT	**Software Master Start.** This parameter specifies the starting Master number for Software Masters. Software masters are those which are accessible via the PCI BAR #1, called the STMR.
SW_MSTR_STP	**Software Master Stop:** This parameter specifies the highest Master number allocated to "software". This number should be less than or equal to 65535 (2^{16}-1).

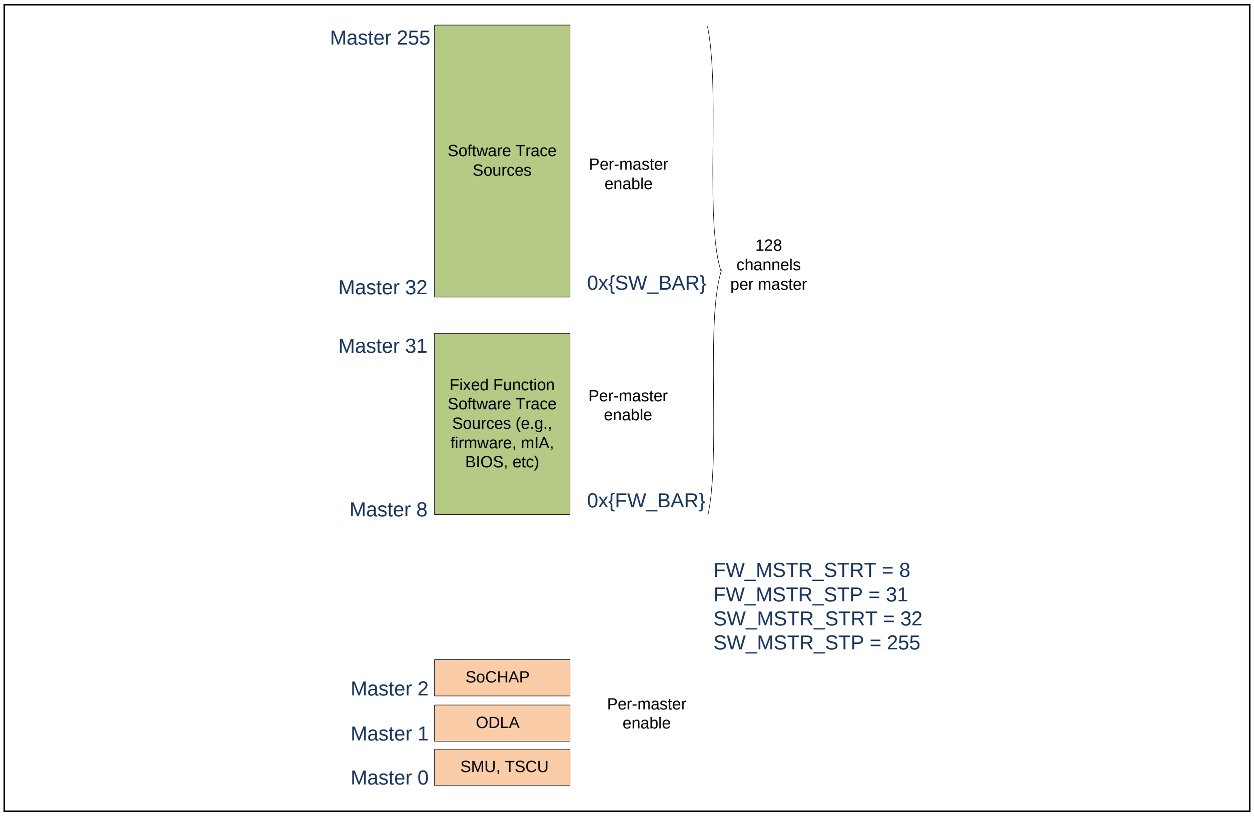
Figure 10 Intel Trace Hub Master Allocation



Each product can allocate its firmware and fixed-function masters, as well as software masters, to match their requirements. Two examples are provided as follows.

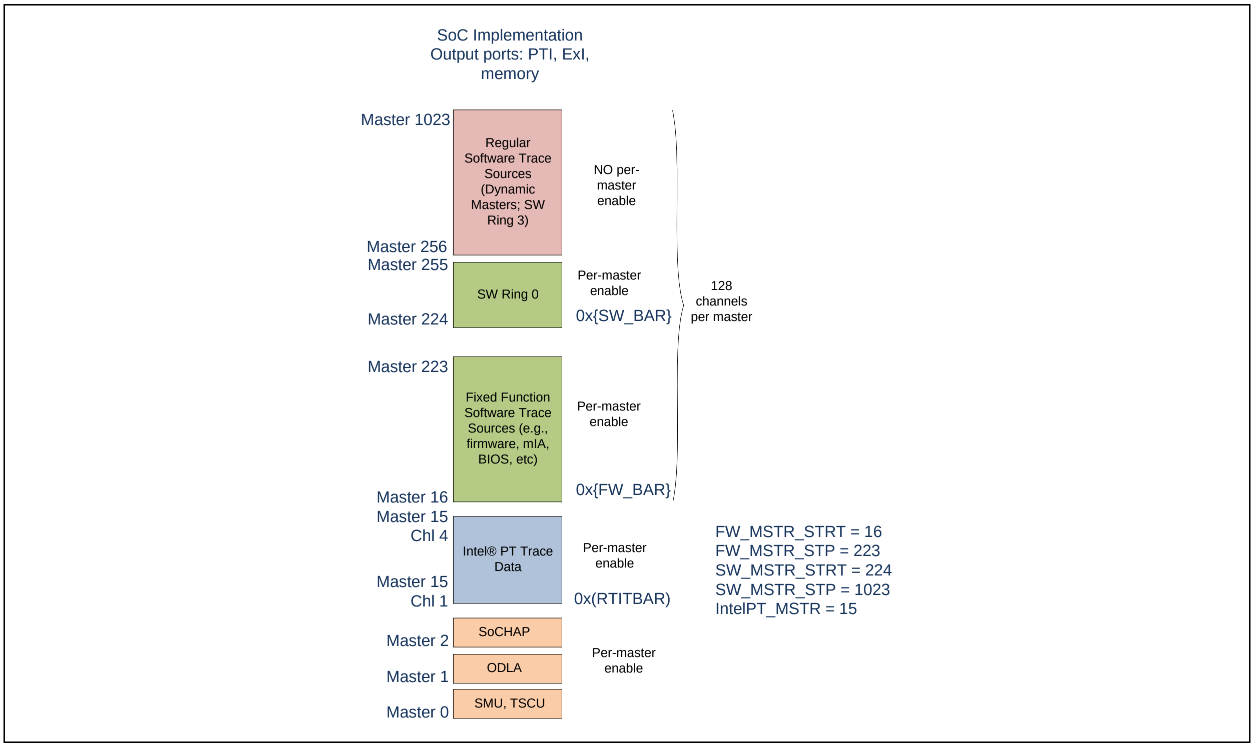
In the first example, shown in Figure 11, Intel® PT support is not implemented, and only hardware and software tracing are supported. In Intel Trace Hub architecture, hardware sources are allocated to the lowest-numbered Masters (0, 1, and 2 in this generation). Firmware trace sources are assigned 24 Masters, starting at Master 8; and general software trace sources are assigned 224 Masters, starting at Master 32. In this example, all firmware and software trace sources have a per-master enable.

Figure 11 Master Allocation Example 1



In the second example, shown in Figure 12, Intel® PT support is present, and many more software Masters are supported (greater than 256; the actual number is unimportant). In this example, firmware and fixed-function software trace sources were allocated to the “firmware” region, and have per-master enables. Software Ring 0 is assigned 28 Masters starting at Master 224. Because of the Master range assignment, the Ring 0 sources have a per-master enable. Starting at Master 256, regular software trace sources can be assigned to one or more Masters, according to the algorithm of the software or the desire of the user.

Figure 12 Master Allocation Example 2



Note: It is possible that a range of software Masters will appear in both the FW_BAR region (FTMR) and the SW_BAR region (STMR). This is because the FW_BAR must ask for, and be allocated, a memory size that is an integer power of two (i.e., it cannot be allocated 1.875MB; the nearest allowed size is 2MB). Since firmware Masters start above zero, and the SW_MSTR_STRT value is merely added to the Master number for offset zero of the SW_BAR), then generally, masters FW_MSTR_STP to FW_MSTR_STP + FW_MSTR_STRT will be present in both the FW_BAR and the SW_BAR. It is the responsibility of firmware and software engineers to ensure their device(s) write to their assigned Master number in their assigned memory region (BAR).

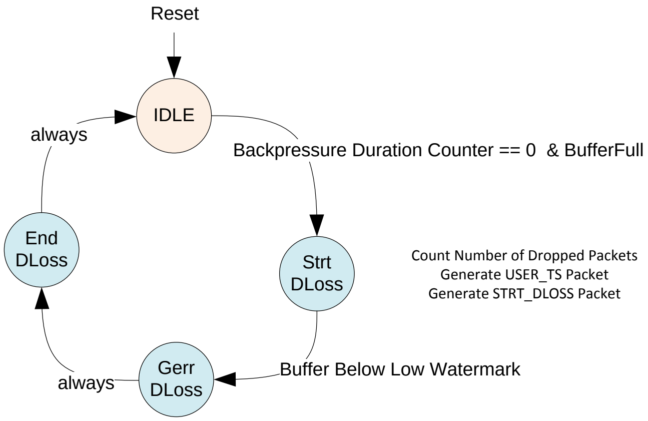
4.3 Lossy Operation

The STH is designed for both lossy and lossless operation. Its lossless operation mode supports the need for complete trace data acquisition, while the lossy operation mode supports minimal system perturbation while storing to primary fabric-based destinations.

When new incoming data is presented to the STH, but it is unable to accept incoming data (most likely because the GTH Input Buffer to which the STH is connected is full) it will start decrementing the Backpressure Duration Counter (BDC). If the GTH starts accepting data again before the counter expires, the counter is reset to its programmed value.

On the other hand, if the counter expires before the GTH starts accepting data again, the STH will enter its Drop Mode, and start its Recovery FSM (finite state machine). Once Drop mode is entered, all data written to the STMR or RTMR will be accepted but immediately dropped. Similarly, any write requests targeting the special CSRs will be accepted immediately and the resulting dataset dropped. Finally, no hardware triggers will be accepted and all resulting trigger datasets will also be dropped.

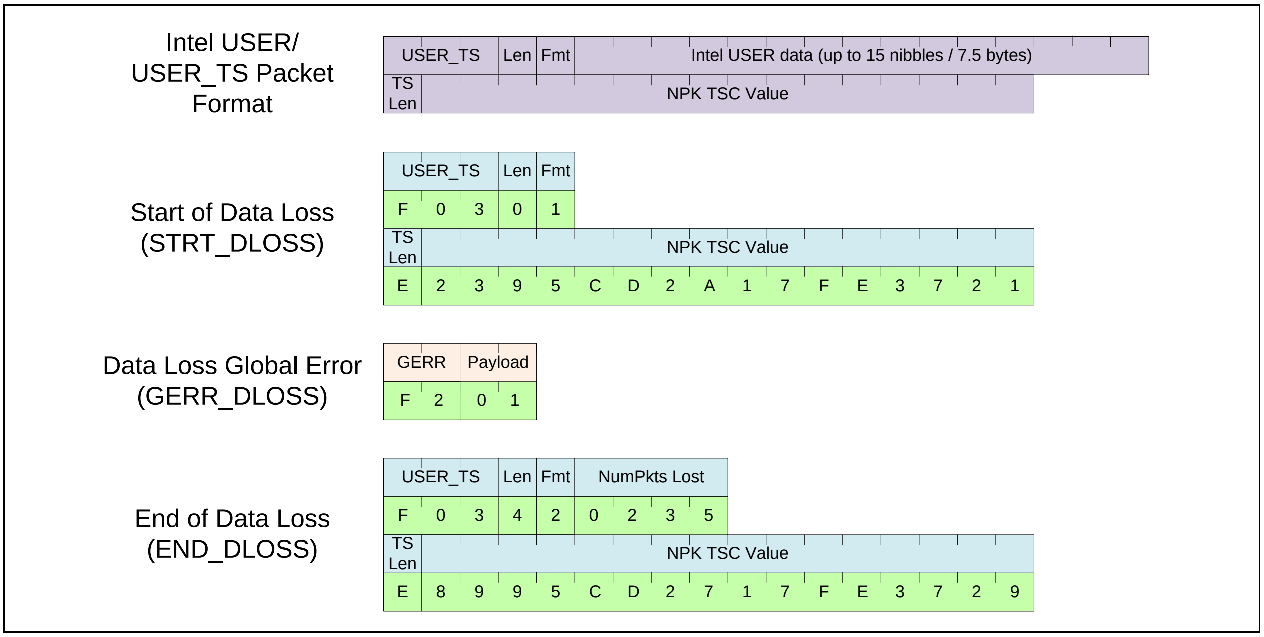
Figure 13 STH Lossy Mode Recovery FSM


Reset
IDLE
always
Backpressure Duration Counter == 0 & BufferFull
End DLoss
Strt DLoss
Count Number of Dropped Packets
Generate USER_TS Packet
Generate STRT_DLOSS Packet
Gerr DLoss
Buffer Below Low Watermark
always

In addition, the first of three special data packets defined to convey the data loss condition to the capture device is queued up. This special packet is referred to as the *Start of Data Loss Packet*, shown in detail in Figure 14. While the FSM is in the *StrtDLoss* state, it will also use a counter known as the Data Packet Lost Counter (DPLC) to count the number of data packets that have been dropped while in this mode of operation. The number of lost packets includes packets from the inbound bus (i.e., coming from primary or sideband fabric), but does not include trigger packet(s) that would have been generated as a result of a TrigIn assertion (from the Trigger Unit). This counter is also visible as a CSR and includes a sticky overflow bit that indicates when the number of data packets lost is greater than the 15-bit counter can accommodate.

The FSM will remain in the *StrtDLoss* state until the corresponding GTH's input buffer asserts its *lowWaterMark* signal, indicating that the input buffer is full only to its low water mark. This approach will ensure that, once data is accepted into STH again, a reasonable amount of data will be passed through before there is a chance of another backpressure and more data loss. Once the *lowWaterMark* signal is asserted, the FSM will move to the *GerrDLoss* state. The FSM is in this state for one clock cycle during which it will generate the second “data loss” special packet, the GERR_DLOSS packet. Moreover, the DPLC will stop being incremented at this state. Next, the FSM will move to the *EndDLoss* state for one clock cycle, during which the third special packet is generated, referred to as End of Data Loss (END_DLOSS) packet, which conveys the number of packets dropped during the lossy period, plus the time stamp right before the STH goes back into normal operation. The FSM will then move back to the IDLE state, allowing the STH to resume its normal operation. The three special packets generated by the Recovery FSM are depicted in Figure 14.

Figure 14: Data Loss Special Packets Format


Intel USER/ USER_TS Packet Format
USER_TS
Len
Fmt
Intel USER data (up to 15 nibbles / 7.5 bytes)
TS Len
NPK TSC Value
Start of Data Loss (STRT_DLOSS)
USER_TS
Len
Fmt
F 0 3 0 1
TS Len
NPK TSC Value
E 2 3 9 5 C D 2 A 1 7 F E 3 7 2 1
Data Loss Global Error (GERR_DLOSS)
GERR
Payload
F 2 0 1
End of Data Loss (END_DLOSS)
USER_TS
Len
Fmt
NumPkts Lost
F 0 3 4 2 0 2 3 5
TS Len
NPK TSC Value
E 8 9 9 5 C D 2 7 1 7 F E 3 7 2 9

4.4 Triggering and Events

4.4.1 Event Match and Mask

The STH monitors incoming data (from the Intel Trace Hub Host Interface) for up to four different data packets, simultaneously. If the incoming data matches the specification in one of the Event Mask/Match registers, the STH will assert an *event* signal to the Intel Trace Hub Trigger Unit. While the match/mask control register contains a BAR (Base Address Register) match and mask value, only transactions targeting the STH will actually be monitored. More specifically, only writes to the STMR, RTMR, and STH CSR range will be monitored; writes to GTH CSRs, or other units' CSRs, will not be monitored, and cannot result in an event assertion to the Trigger Unit. Further, event monitoring is independent of the STH's *storeEn* signal. That is, the event match/mask logic is monitoring and reporting events, whether *storeEn* is asserted or deasserted.

Based on its configuration, the CTS can assert back the *TrigIn* signal one or more times. The behavior of the STH in this situation is described in the following section.

Note: The CTS has the capability to assert *storeEn* to the STH based on an event signal from STH to CTS. Unfortunately, due to the latency from STH event detection to TrigIn assertion, the event that precipitates a *storeEn* assertion will disappear (will not be stored; not sent from STH to GTH) before the *storeEn* is actually asserted. Thus, the "triggering event" will not be the first event to appear in the resultant trace data.

4.4.2 Trigger with Timestamp Packet Generation

If TRIGTSPP.TRIGEN is set, then each time the *TrigIn* is asserted the STH will insert a TRIG_TS packet into its data stream.

Take note that the STH can keep track of only one input trigger at a time while the GTH is full. If two or more TrigIn assertions are detected while the GTH is full, only the most recent TrigIn assertion will result in a TRIG_TS packet (specifically a single TRIG_TS packet, not multiple TRIG_TS packets), with the timestamp of the most recent TrigIn assertion.

On the other hand, if BDC ever expires during the above process, everything is simply dropped/ignored until the STH recovers from the Drop Mode operation then re-start collecting and sending the received TRIG_TS packets.

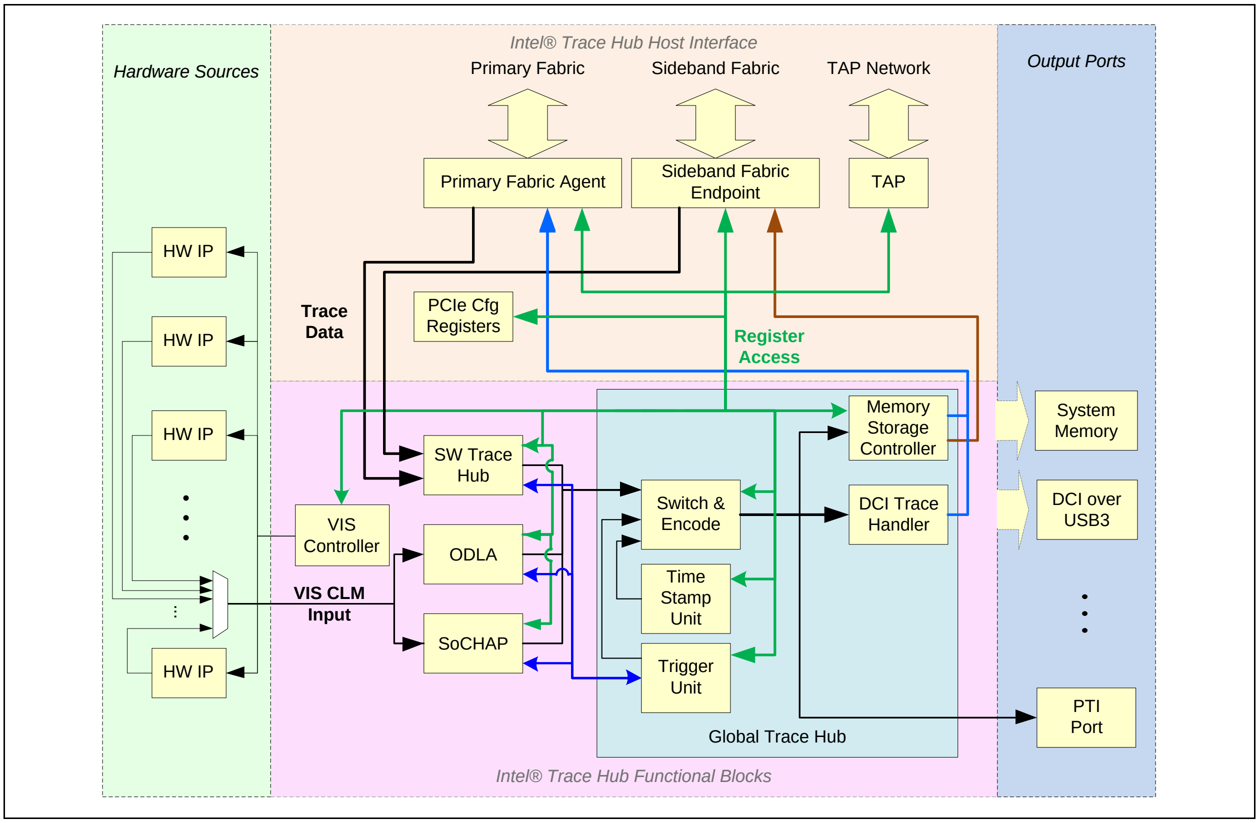
4.5 Time Stamping

The STH has the ability to time-stamp incoming data. Since it is a member of the Intel® Trace Hub Architecture, it uses the Time Stamp Counter from the TSCU for time-stamping data upon demand. Because the STH will be physically very near the TSCU, the full 64-bit Intel® Trace Hub TSC is connected as a direct input to the STH for use in time stamping.

5 Global Trace Hub

The Global Trace Hub (GTH) is an ingredient of the Intel® Trace Hub Architecture: it is the central block that gathers all debug trace data[9], encodes it in MIPI STPv2.1 format, and sends it to various destinations to suit the needs of the debugger. The debug trace data can be stored in system memory for a truly probe-less debug solution. It can also be sent to any supported trace destination or debug port, including the DCI and MIPI-PTI. Figure 30 illustrates the GTH architecture in the context of the Intel® Trace Hub architecture, and shows some of the internal GTH components.

Figure 15: GTH in Intel® Trace Hub



The GTH can be bypassed completely, supporting a from-power-on debug capability. In this mode, the GTH will be bypassed, allowing the user to route VIS signals directly to an appropriate debug port (for example, PTI/GPIO).

Finally, the GTH can be used as a means to dump system memory to an output port, such as PTI. This is useful for such things as dumping previously-acquired Intel® Processor Trace

[9] *Debug trace sources are those IP blocks (usually DFx blocks) that gather or generate data used for debug, and include ODLA, STH, and SoCHAP*

data (stored in DRAM) directly to a data capture device connected to the PTI port. It can also be very useful in SoCs where a fairly high-speed output port exists, such as DCI/USB3, but the trace data bandwidth needed for a debug scenario is higher than the port's capability. In this case, the trace data can be stored to system memory, and later sent out the high-speed trace port, without loss.

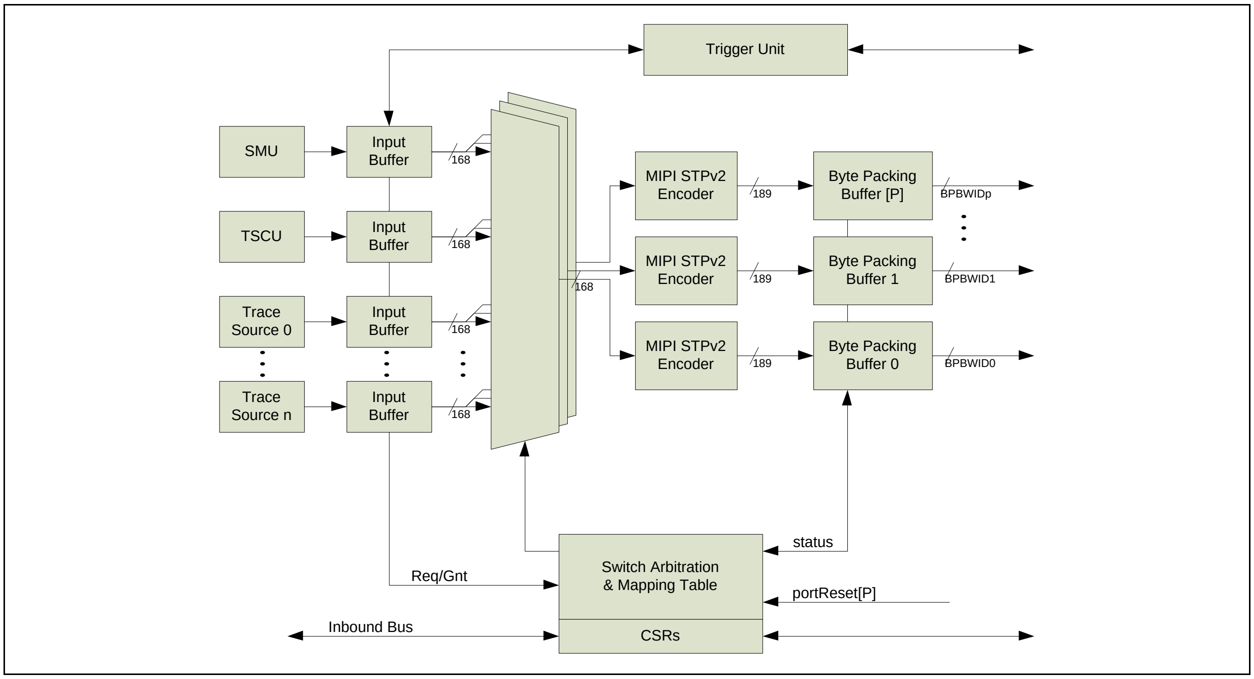
5.1 Overview

- **Supports the following trace data sources:**
  - ODLA
  - SoCHAP
  - Software Trace Hub
    - Intel® Processor Trace Sources
    - Regular Software sources
    - Firmware sources
  - Time Stamp Correlation Unit (TSCU)
  - STP Maintenance Unit (an integral component of GTH)
- **Software-assisted memory dump:**
  - MIPI STPv2.1 encoder can be disabled, allowing software to send trace data in memory to an output port.
- **Compliant to MIPI STPv2.1 protocol standard**
  - Supports up to 65536 Masters, and 256 Channels per Master
- **Supports multiple output ports**
  - System DRAM (via primary fabric)
  - DCI
  - MIPI-PTI
- **One destination for each STPv2.1 Master**
  - No multicast or broadcast
- **Backpressure:** Supports debug data sources with and without backpressure support
  - Trace Sources with backpressure support will cleanly drop packets
  - Trace sources without backpressure support will have forcibly dropped packets.
- **Intel® Trace Hub Internal Bus interfaces** for sending and receiving debug trace data, and accessing CSRs (control/status registers)

5.2 Functional description

The GTH is a central element of the Intel® Trace Hub architecture. The GTH is essentially a data switch with a protocol encapsulation function. It takes, as inputs, debug data from various trace sources, encapsulates that data into the MIPI STPv2.1 protocol, and sends it to a destination port. Because the GTH is a switch, supporting multiple simultaneous destination ports, different types of data can be sent to different ports. For example, Intel® Processor Trace data can be sent to the PTI port, while software (application, OS, and driver) debug messages can be sent to system memory (via primary fabric). The number of output streams is implementation dependent, though two ports is the minimum. Output ports include, but are not limited to: system memory (which can also be thought-of as the primary fabric interface) and the MIPI PTI.

Figure 16: Global Trace Hub Architecture



The Global Trace Hub is comprised of the following functional blocks:

- Input Buffers: a simple set of buffers that are provided to allow for absorbing input bursts.
- Data Switch: a full-crossbar switch, allowing any single input to be routed to any single output.
- MIPI STPv2.1 Encoders
- Byte Packing Buffers
- STP Maintenance Unit (SMU): sends periodic MIPI STPv2.1 synchronization packets to the trace destination.
- Switch Arbitration and Mapping Table
- Configuration and Status Registers: provides access to CSRs

These blocks are presented in more detail in the following subsections.

Throughout this document, the designation '[N]' and '[P]' are used to refer to trace sources and output ports, respectively. That is, whenever a signal or aspect of the architecture applies to the number of trace sources, or a particular trace source, the letter 'N' is used to refer to it. Whenever a signal or aspect of the architecture applies to, or refers to, the number of trace destinations, or a particular trace destination, the letter 'P' is used to refer to it. For example, in Figure 16, the signal *full[P]* uses the letter 'P', indicating there is a *full* signal for each output port. Similarly, the signal *grant[N]* uses the letter 'N', indicating there is a *grant* signal for each trace source (input buffer).

5.2.1 Input Buffers

The Input Buffer serves as a peak bandwidth buffer, as well as a latency buffer between the trace source and the Data Switch.

Data is accepted into the Input Buffer from a trace source when there is at least one free entry.

The Input Buffer performs the first two tasks in the overall data switching operation. First, it filters out trace data from disabled Masters. That is, when the MAST[N]EN bit in DEST[N] register is cleared (zero) for a particular Master, trace data is received at the input buffer from that Master and it will be immediately dropped.

Second, for Masters that are not disabled, the Input Buffer looks up the output port for that Master, and stores the destination port number (dest[2:0]) in the Input Buffer, along with the trace data.

The input buffer has one responsibility in supporting the GTH switch performance optimizations: it requests to send data to the Data Switch when there are sufficient data sets in the Input Buffer to ensure the programmable grant duration (*grantDur[3:0]*) will be fully utilized. The Data Switch arbiter is designed to accommodate the case where fewer than *grantDur* data sets are ready to be forwarded to a byte packing buffer. However, this results in lower efficiency for use of the STPv2.1 protocol, and should be avoided. The only exception to this rule is when the trace source is turned off (by deasserting its Store Enable). The input buffer then asserts its *req* signal as long as there is data in the input buffer, regardless of how many entries are in the buffer.

5.2.1.1 Standard Trace Source Input Buffer Assignment

Since there are a number of standard trace sources that come with the Intel® Trace Hub IP, these sources are assigned to fixed input buffers, so that software has a baseline from which to build a standard software release. Toward this end, the following assignments are part of the fundamental architecture of Intel® Trace Hub:

Table 5-1: Standard Trace Source Assignments

Trace Source	Trace Source Number	CTS Store Qual signal	CTS Gasket StoreEn Signal
SMU	0	n/a	
TSCU	1	n/a	

SoCHAP	2	1	2
ODLA	3	2	3
STH	4	3	4
RTS	5	4	5

Additional details on *StoreQual* and *StoreEn* signal assignments are discussed later in this section.

5.2.1.2 Input Buffer Depth

The depth of the input buffer is a synthesis parameter, and will depend on the needs of the particular data source, as well as the characteristics of the available output ports. A depth of zero is permitted, but only for single-destination trace sources. This will be used for the STP Maintenance Unit, as its data is always fixed-format, and does not need buffering. The TSCU requires an input buffer with depth of 1, as it generates one packet every several-hundred to several-thousand cycles, but has no ability to save that packet. Zero Depth Input Buffers must have their grant duration set to 1.

5.2.1.3 Backpressure / Flow Control

The Input Buffer can be connected to trace sources that have backpressure capabilities, as well as those that do not. For sources that have backpressure capabilities, the Input Buffer will indicate to the source when the input buffer is full, and cannot accept more data. When the Input Buffer is full, and the trace source itself is unable to further-buffer data, it should engage its backpressure or flow-control mechanism. The method for dropping debug data and handling that condition in software or post-processing is determined solely by the trace source.

For debug trace sources that do not have backpressure capabilities, the Input Buffer's full condition signal will be left unconnected. In this case, the debug trace source will send its debug data, assuming the Input Buffer has captured it. When the Input Buffer is full, it will not capture the data sent from the trace source; the data will be lost.

The Input Buffer also has a threshold function, whereby it can indicate when the Input Buffer is filled below, or beyond, a certain value. The intended primary functionality is that of a low water mark: the signal (LowWaterMark) will be asserted when there are LWMn (where *n* is the trace source number; see LWM registers) entries or less in the Input Buffer. The primary use of this signal is by the STH, which will use *lowWaterMark* to exit its 'drop mode'. In order to ensure the STH sees the *lowWaterMark* signal asserted and deasserted at the right time, the values programmed into the LWM0 and LWM1 registers need to be such that the low water mark is 1 less than the grant duration for the given trace source.

5.2.1.4 Input Buffer Empty

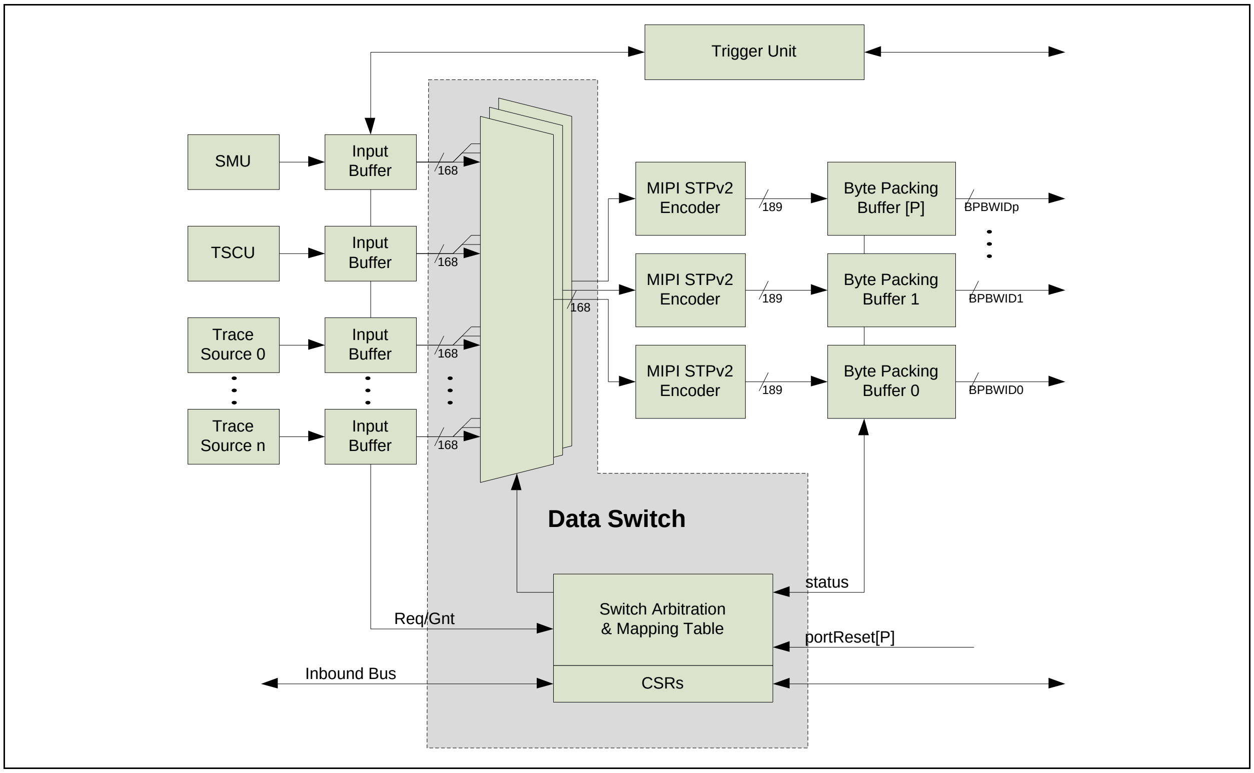
The Input Buffer generates a signal called iBufEmpty. This signal is a vector, iBufEmpty*t*, where '*t*' is the number of Memory Storage Controllers (MSCs). The iBufEmpty signal indicates that, for purposes of the trace destination, the input buffer is empty. It does not indicate the input buffer is actually empty, but rather that, as far as the trace destination *t* is concerned, the input buffer is empty. This signal is driven by each input buffer to the MSC to determine if the pipeline feeding it is empty.

5.2.2 Data Switch

The Data Switch performs the switching function of the GTH, forwarding data from an input buffer to a destination according to the settings in GTH DEST[N] registers. All data switching decisions are based solely on the STPv2.1 Master, and not on the Channel. That is, all data from all Channels of a given Master will be sent to the same trace destination.

The architecture of the Data allows any input to be sent to any output.

Figure 17: Data Switch



Arbitration among the input buffers in the GTH is a simple round-robin algorithm, per Data Switch mux, starting with trace source #0, which must always be the STP Maintenance Unit (SMU). This ensures that, when tracing is started, the ASYNC, VERSION, and FREQ packets are sent first. If the TSCU is configured before tracing is started, then the SMU packets will be followed immediately by a single time stamp correlation packet, since TSCU is trace source #1. Arbitration among the trace sources (2 and higher) is also round-robin, with each arbiter receiving single requests from trace sources that have data available for that particular trace destination.

The arbiter employs a mechanism to increase data and packet throughput: programmable grant duration. The grant duration is programmable, allowing designs to maximize protocol efficiency on a per-debug-scenario basis. A greater number of packets sent back-to-back from one trace source results in increased efficiency in the STPv2.1 protocol, in terms of number of data bits transferred vs. total bits to perform the transfer. This is important because the efficiency of the first packet from a given master is rather low: it ranges from

22% to 70%, depending on the data size. The efficiency of the STPv2.1 protocol can be increased to as much as 94% if buffering is increased, and several data sets from a single master are sent in a burst. Grant duration can be set from 1 to MAXGNT[n] consecutive clocks. Since there is very little efficiency increase beyond 16 consecutive packets (2%, when increasing from 16 to 32), even for the smallest packet size (8 data bits), the value of the MAXGNT[n] parameter is limited to 16 or less.

When a trace destination is reset, or otherwise needs re-initialization in the middle of sending trace data, it will assert its *portReset* signal. Depending on the setting of GTHOPT.DRP bit, the byte packing buffer may fill up, asserting its full output, and stalling the entire pipeline. The Input buffers will eventually fill up, and stall the trace sources. Because the *portReset* signal is asserted, the SMU and TSCU will prepare packets for the port in reset. Also, the arbiter for the output port must be “reset”, in the sense that it will wait for portReset to deassert, and then immediately start granting at Input Buffer 0 (that is, it does not pick up where it left off). When *portReset* is deasserted, the pipeline will be unstalled. In-flight packets will be drained, and the arbiter will begin again at Input Buffer 0 for its arbitration.

5.2.3 MIPI* STPv2.1 Encoder

The MIPI STPv2.1 Encoder’s main function is to translate a data set received from the Data Switch to one or more MIPI STPv2.1 encoded packets.

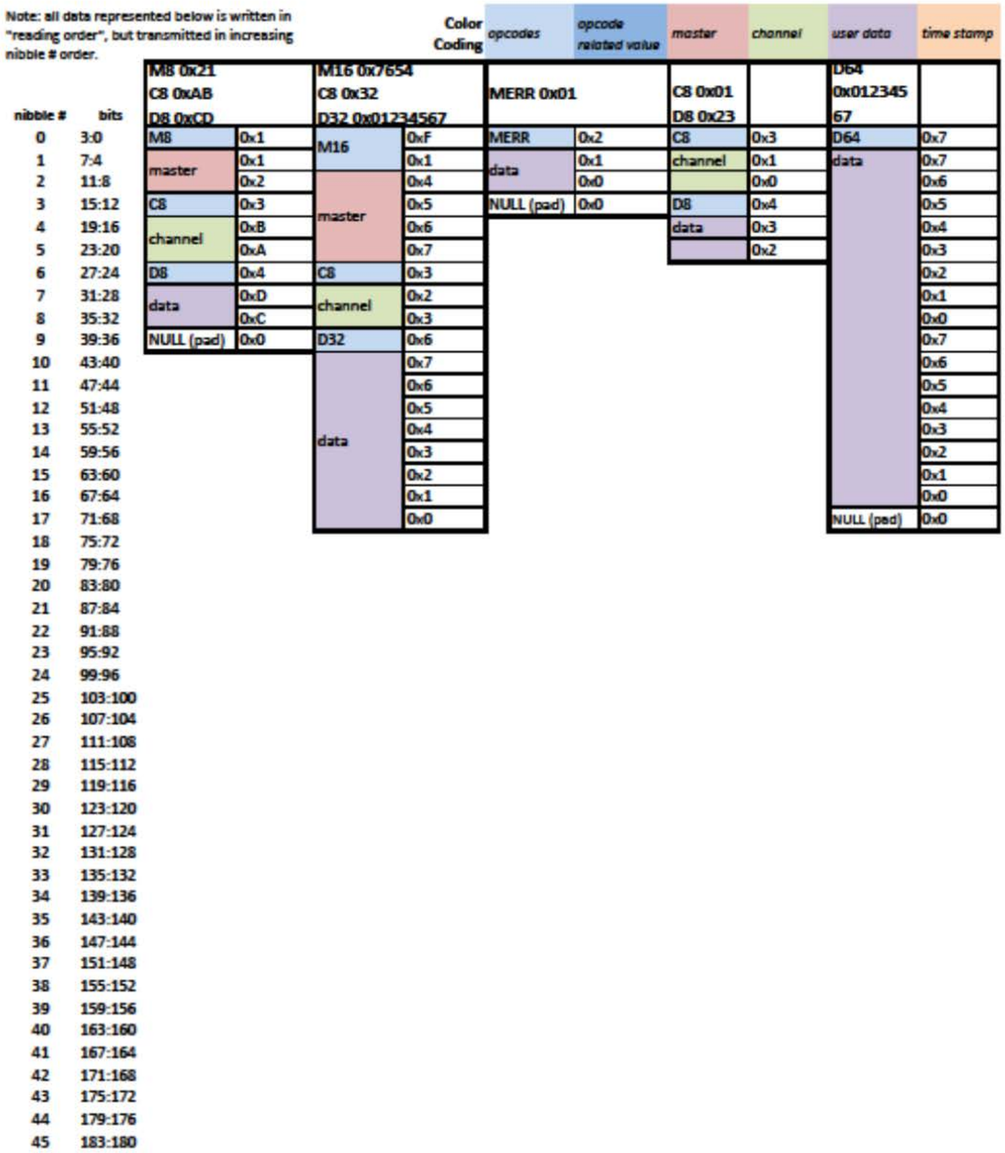
The encoder keeps track of the currently active Master and Channel. For each new data set received from the Data Switch, if the Master and Channel are unchanged, the encoder will not create Master (M16, M8) and Channel (C8) packets, resulting in a more efficient use of the available bandwidth. Conversely, when the new data set Master differs from the current Master, a Master (M16 or M8) packet will be created. Similarly, when the new data set Channel is different than the current Channel, a Channel (C8) packet will be created. Thus, the encoder will create from one, two or three MIPI STPv2.1 packets for each data set received.

The currently active Master and Channel are reset to zero when the STPv2.1 Encoder detects the VERSION packet in its input data set. This will result in the next data set having a Master packet, and perhaps a Channel packet, unless the next data set is from Master 0 and Channel 0. The stored timestamp for all clock domains (tsClkDom[3:0]) is also reset to zero, which will result in the next time stamp encoded to be a maximum-length time-stamp. This is done to meet the MIPI STPv2.1 requirement that the next timestamp after a VERSION packet is the longest time stamp that the system is able to produce.

The final stage of the MIPI STP v2.1 encoder is the Aligner block, which is responsible for ensuring that the MIPI STPv2.1 encoded packet(s) are placed onto the low order byte lanes of the output bus. The data is mapped onto the output bus in network order, by nibble, from low (least significant) nibble to high (most significant) nibble. That is, the least-significant nibble on the output bus is the first nibble, in terms of network order.

Since the Intel Trace Hub operates inside Intel SoCs, the STP-encoded *payload* data it produces is little-endian (STP opcodes are merely a stream of nibbles, and have no endianness associated with them. The payload portion of the packets can have an endianness, hence the translation from nibble order to network order is important). That is, the least-significant nibble of a byte, word, DWord, or Qword is sent first, followed by the next least significant nibble, and so on. When sent to an output port, the first nibble is, obviously, the first nibble to be sent. When stored to memory, the data will be stored with first nibble in the least significant nibble of byte 0, second nibble in the most significant nibble of byte 0, the third nibble in the least significant nibble of byte 1, and so on. It is also the first nibble received, and the first to be interpreted. See the next five figures below for detailed information on nibble order for each STP packet encoded by the Intel Trace Hub. See Figure 23 for a diagram showing MIPI STP v2.1 encoded data as it is stored in memory.

Figure 18: Nibble order (1 of 5)

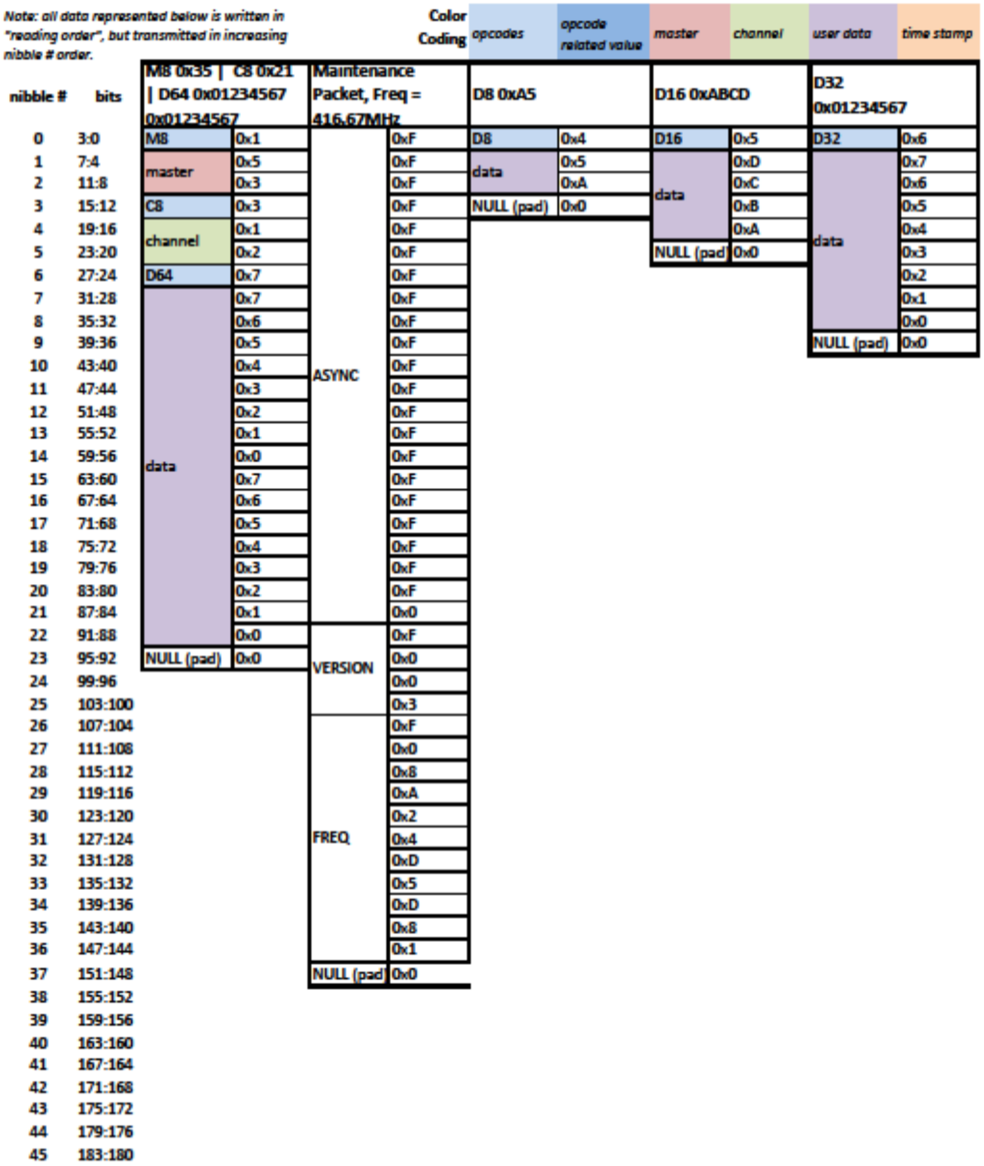


Note: all data represented below is written in "reading order", but transmitted in increasing nibble # order.

Color Coding					
opcodes	*opcode related value*	*master*	*channel*	*user data*	*time stamp*

nibble #	bits	M8 0x21 C8 0xAB D8 0xCD		M16 0x7654 C8 0x32 D32 0x01234567		MERR 0x01		C8 0x01 D8 0x23		D64 0x012345 67	
0	3:0	M8	0x1	M16	0xF	MERR	0x2	C8	0x3	D64	0x7
1	7:4	master	0x1		0x1	data	0x1	channel	0x1	data	0x7
2	11:8		0x2	master	0x4		0x0		0x0		0x6
3	15:12	C8	0x3		0x5	NULL (pad)	0x0	D8	0x4		0x5
4	19:16	channel	0xB		0x6			data	0x3		0x4
5	23:20		0xA		0x7				0x2		0x3
6	27:24	D8	0x4	C8	0x3						0x2
7	31:28	data	0xD	channel	0x2						0x1
8	35:32		0xC		0x3						0x0
9	39:36	NULL (pad)	0x0	D32	0x6						0x7
10	43:40			data	0x7						0x6
11	47:44				0x6						0x5
12	51:48				0x5						0x4
13	55:52				0x4						0x3
14	59:56				0x3						0x2
15	63:60				0x2						0x1
16	67:64				0x1						0x0
17	71:68				0x0					NULL (pad)	0x0
18	75:72										
19	79:76										
20	83:80										
21	87:84										
22	91:88										
23	95:92										
24	99:96										
25	103:100										
26	107:104										
27	111:108										
28	115:112										
29	119:116										
30	123:120										
31	127:124										
32	131:128										
33	135:132										
34	139:136										
35	143:140										
36	147:144										
37	151:148										
38	155:152										
39	159:156										
40	163:160										
41	167:164										
42	171:168										
43	175:172										
44	179:176										
45	183:180										

Figure 19: Nibble order (2 of 5)

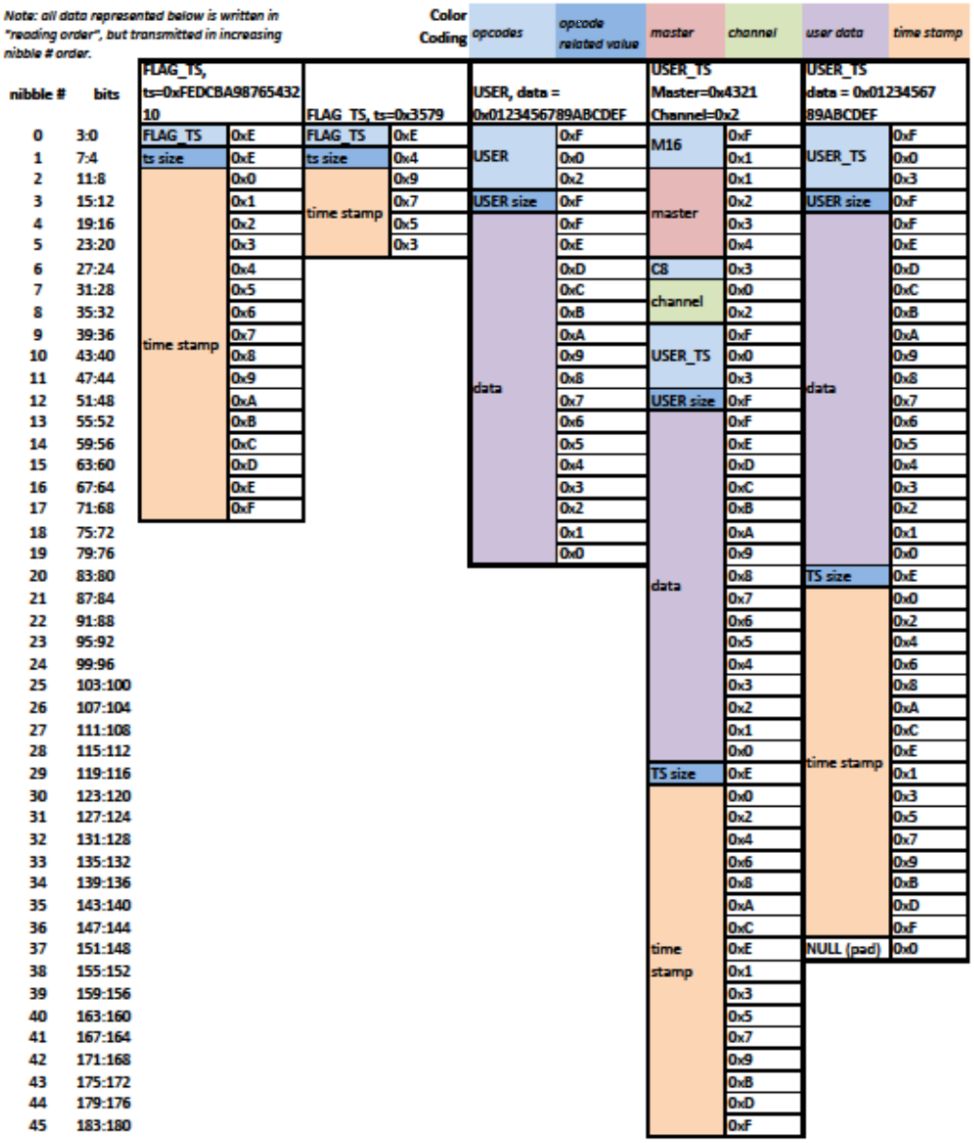


Note: all data represented below is written in "reading order", but transmitted in increasing nibble # order.

Color Coding	*opcodes*	*opcode related value*	*master*	*channel*	*user data*	*time stamp*

nibble #	bits	M8 0x35 | C8 0x21 | D64 0x01234567 0x01234567		Maintenance Packet, Freq = 416.67MHz		D8 0xA5		D16 0xABCD		D32 0x01234567	
0	3:0	M8	0x1	ASYNC	0xF	D8	0x4	D16	0x5	D32	0x6
1	7:4	master	0x5		0xF	data	0x5	data	0xD	data	0x7
2	11:8		0x3		0xF		0xA		0xC		0x6
3	15:12	C8	0x3		0xF	NULL (pad)	0x0		0xB		0x5
4	19:16	channel	0x1		0xF				0xA		0x4
5	23:20		0x2		0xF			NULL (pad)	0x0		0x3
6	27:24	D64	0x7		0xF						0x2
7	31:28	data	0x7		0xF						0x1
8	35:32		0x6		0xF						0x0
9	39:36		0x5		0xF					NULL (pad)	0x0
10	43:40		0x4		0xF						
11	47:44		0x3		0xF						
12	51:48		0x2		0xF						
13	55:52		0x1		0xF						
14	59:56		0x0		0xF						
15	63:60		0x7		0xF						
16	67:64		0x6		0xF						
17	71:68		0x5		0xF						
18	75:72		0x4		0xF						
19	79:76		0x3		0xF						
20	83:80		0x2		0xF						
21	87:84		0x1		0x0						
22	91:88		0x0	VERSION	0xF						
23	95:92	NULL (pad)	0x0		0x0						
24	99:96				0x0						
25	103:100				0x3						
26	107:104			FREQ	0xF						
27	111:108				0x0						
28	115:112				0x8						
29	119:116				0xA						
30	123:120				0x2						
31	127:124				0x4						
32	131:128				0xD						
33	135:132				0x5						
34	139:136				0xD						
35	143:140				0x8						
36	147:144				0x1						
37	151:148			NULL (pad)	0x0						
38	155:152										
39	159:156										
40	163:160										
41	167:164										
42	171:168										
43	175:172										
44	179:176										
45	183:180										

Figure 20: Nibble order (3 of 5)

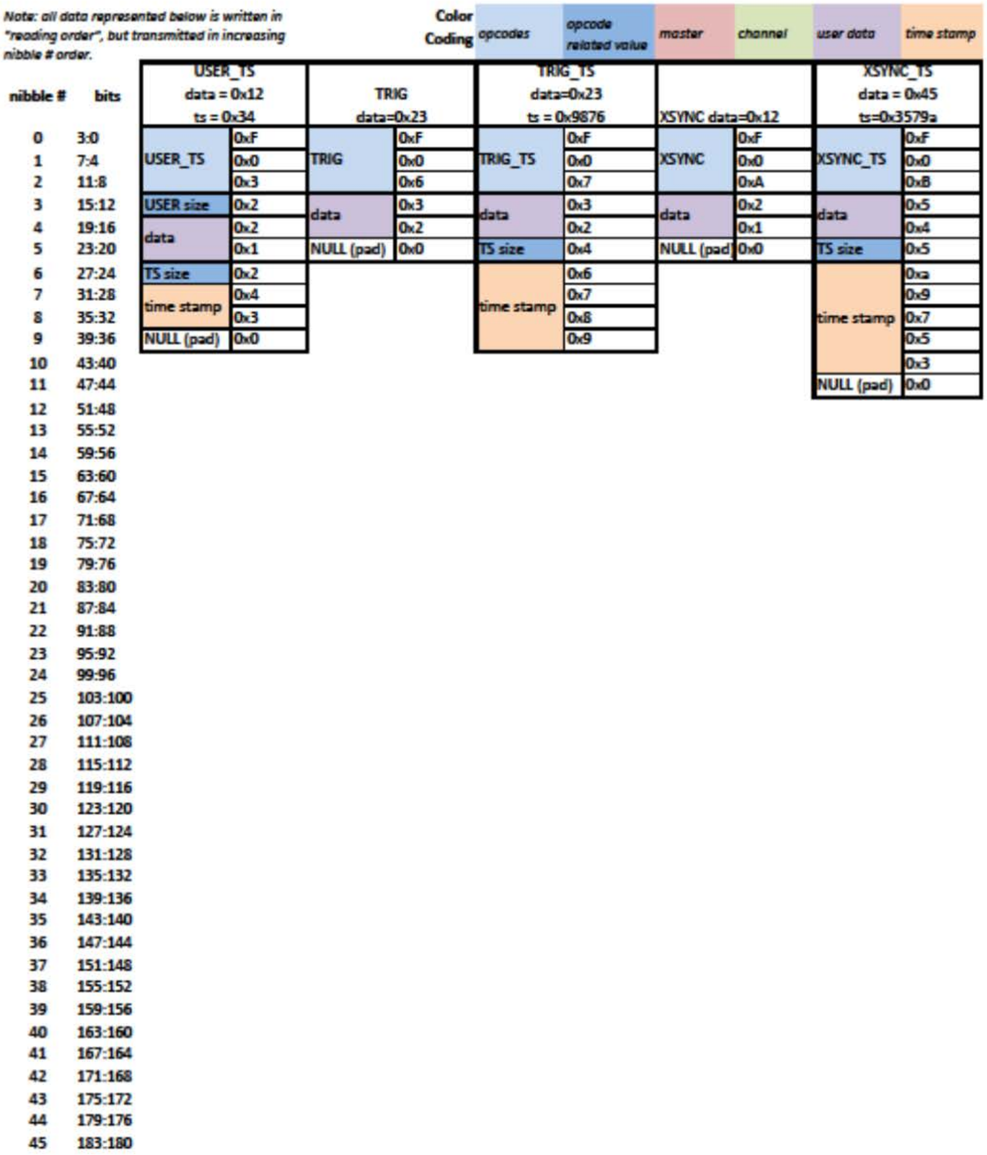


Note: all data represented below is written in "reading order", but transmitted in increasing nibble # order.

Color Coding: *opcodes* | *opcode related value* | *master* | *channel* | *user data* | *time stamp*

nibble #	bits	FLAG_TS, ts=0xFEDCBA9876543210		FLAG_TS, ts=0x3579		USER, data = 0x0123456789ABCDEF		USER_TS Master=0x4321 Channel=0x2		USER_TS data = 0x0123456789ABCDEF	
0	3:0	FLAG_TS	0xE	FLAG_TS	0xE	USER	0xF	M16	0xF	USER_TS	0xF
1	7:4	ts size	0xE	ts size	0x4		0x0		0x1		0x0
2	11:8	time stamp	0x0	time stamp	0x9		0x2	master	0x1		0x3
3	15:12		0x1		0x7	USER size	0xF		0x2	USER size	0xF
4	19:16		0x2		0x5	data	0xF		0x3	data	0xF
5	23:20		0x3		0x3		0xE		0x4		0xE
6	27:24		0x4				0xD	C8	0x3		0xD
7	31:28		0x5				0xC	channel	0x0		0xC
8	35:32		0x6				0xB		0x2		0xB
9	39:36		0x7				0xA	USER_TS	0xF		0xA
10	43:40		0x8				0x9		0x0		0x9
11	47:44		0x9				0x8		0x3		0x8
12	51:48		0xA				0x7	USER size	0xF		0x7
13	55:52		0xB				0x6	data	0xF		0x6
14	59:56		0xC				0x5		0xE		0x5
15	63:60		0xD				0x4		0xD		0x4
16	67:64		0xE				0x3		0xC		0x3
17	71:68		0xF				0x2		0xB		0x2
18	75:72						0x1		0xA		0x1
19	79:76						0x0		0x9		0x0
20	83:80								0x8	TS size	0xE
21	87:84								0x7	time stamp	0x0
22	91:88								0x6		0x2
23	95:92								0x5		0x4
24	99:96								0x4		0x6
25	103:100								0x3		0x8
26	107:104								0x2		0xA
27	111:108								0x1		0xC
28	115:112								0x0		0xE
29	119:116							TS size	0xE		0x1
30	123:120							time stamp	0x0		0x3
31	127:124								0x2		0x5
32	131:128								0x4		0x7
33	135:132								0x6		0x9
34	139:136								0x8		0xB
35	143:140								0xA		0xD
36	147:144								0xC		0xF
37	151:148								0xE	NULL (pad)	0x0
38	155:152								0x1		
39	159:156								0x3		
40	163:160								0x5		
41	167:164								0x7		
42	171:168								0x9		
43	175:172								0xB		
44	179:176								0xD		
45	183:180								0xF		

Figure 21: Nibble order (4 of 5)



Note: all data represented below is written in "reading order", but transmitted in increasing nibble # order.

Color Coding					
opcodes	*opcode related value*	*master*	*channel*	*user data*	*time stamp*

nibble #	bits	USER_TS data = 0x12 ts = 0x34		TRIG data=0x23		TRIG_TS data=0x23 ts = 0x9876		XSYNC data=0x12		XSYNC_TS data = 0x45 ts=0x3579a	
0	3:0	USER_TS	0xF	TRIG	0xF	TRIG_TS	0xF	XSYNC	0xF	XSYNC_TS	0xF
1	7:4		0x0		0x0		0x0		0x0		0x0
2	11:8		0x3		0x6		0x7		0xA		0xB
3	15:12	USER size	0x2	data	0x3	data	0x3	data	0x2	data	0x5
4	19:16	data	0x2		0x2		0x2		0x1		0x4
5	23:20		0x1	NULL (pad)	0x0	TS size	0x4	NULL (pad)	0x0	TS size	0x5
6	27:24	TS size	0x2			time stamp	0x6			time stamp	0xa
7	31:28	time stamp	0x4				0x7				0x9
8	35:32		0x3				0x8				0x7
9	39:36	NULL (pad)	0x0				0x9				0x5
10	43:40										0x3
11	47:44									NULL (pad)	0x0
12	51:48										
13	55:52										
14	59:56										
15	63:60										
16	67:64										
17	71:68										
18	75:72										
19	79:76										
20	83:80										
21	87:84										
22	91:88										
23	95:92										
24	99:96										
25	103:100										
26	107:104										
27	111:108										
28	115:112										
29	119:116										
30	123:120										
31	127:124										
32	131:128										
33	135:132										
34	139:136										
35	143:140										
36	147:144										
37	151:148										
38	155:152										
39	159:156										
40	163:160										
41	167:164										
42	171:168										
43	175:172										
44	179:176										
45	183:180										

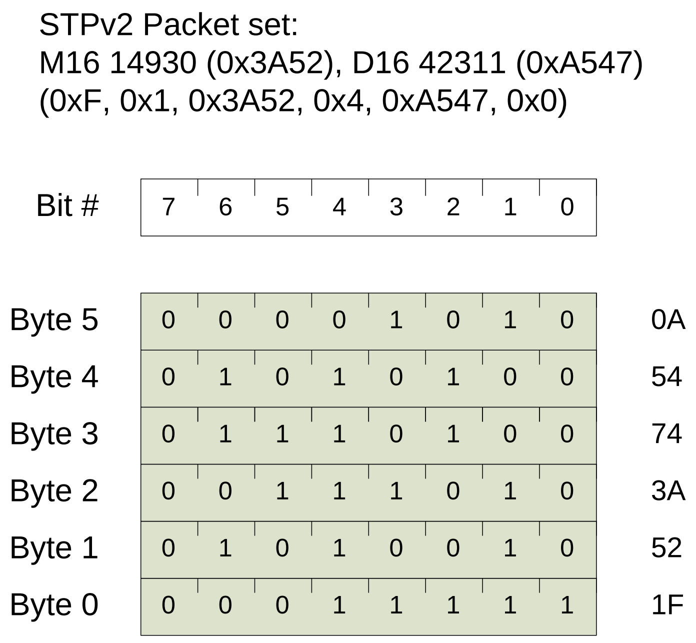
Figure 22: Nibble order (5 of 5)

Note: all data represented below is written in "reading order", but transmitted in increasing nibble # order.

Color Coding	opcodes	opcode related value	master	channel	user data	time stamp

nibble #	bits	GERR data = 0x01		D64TS data = 0x01234567 89ABCDEF		FLAG		D64TS Master 0x1234 Channel 0x56		TIME_TS ts_other[63:0] = 0x8900112233445566		TIME_TS ts_other[23:0] = 0xAB5678	
0	3:0	GERR	0xF	D64TS	0xF	FLAG	0xF	M16	0xF	TIME_TS	0xF	TIME_TS	0xF
1	7:4		0x2		0x7		0xE		0x1		0x0		0x0
2	11:8	data	0x1	data	0xF			master	0x4		0x5		0x5
3	15:12		0x0		0xE				0x3	clock_domai	0x1	clock_doma	0x1
4	19:16				0xD				0x2	other_ts_for	0x4	other_ts_fo	0x4
5	23:20				0xC				0x1	TS_Other_siz	0xE	TS_Other_siz	0x6
6	27:24				0xB			C8	0x3	TS_Other	0x6	TS_Other	0x8
7	31:28				0xA			channel	0x6		0x6		0x7
8	35:32				0x9				0x5		0x5		0x6
9	39:36				0x8			D64TS	0xF		0x5		0x5
10	43:40				0x7				0x7		0x4		0xB
11	47:44				0x6			data	0xF		0x4		0xA
12	51:48				0x5				0xE		0x3	TS size	0x5
13	55:52				0x4				0xD		0x3	time stamp	0xE
14	59:56				0x3				0xC		0x2		0xC
15	63:60				0x2				0xB		0x2		0xA
16	67:64				0x1				0xA		0x1		0x9
17	71:68				0x0				0x9		0x1		0x7
18	75:72			TS size	0x6				0x8		0x0		
19	79:76			time stamp	0xA				0x7		0x0		
20	83:80				0x9				0x6		0x9		
21	87:84				0x7				0x5		0x8		
22	91:88				0x5				0x4	TS size	0xE		
23	95:92				0x3				0x3	time stamp	0xF		
24	99:96				0x1				0x2		0xE		
25	103:100			NULL (pad)	0x0				0x1		0xD		
26	107:104								0x0		0xC		
27	111:108							TS size	0xE		0xB		
28	115:112							time stamp	0X3		0xA		
29	119:116								X06		0x9		
30	123:120								X04		0x8		
31	127:124								X09		0x7		
32	131:128								0X8		0x6		
33	135:132								0X1		0x5		
34	139:136								0X6		0x4		
35	143:140								0X2		0x3		
36	147:144								0X3		0x2		
37	151:148								0X4		0x1		
38	155:152								0X7		0x0		
39	159:156								0XF				
40	163:160								0X2				
41	167:164								0XC				
42	171:168								0X9				
43	175:172								0XA				
44	179:176												
45	183:180												

Figure 23: Encoder Nibble Ordering in Memory



STPv2 Packet set:
M16 14930 (0x3A52), D16 42311 (0xA547)
(0xF, 0x1, 0x3A52, 0x4, 0xA547, 0x0)

Bit #	7	6	5	4	3	2	1	0	
Byte 5	0	0	0	0	1	0	1	0	0A
Byte 4	0	1	0	1	0	1	0	0	54
Byte 3	0	1	1	1	0	1	0	0	74
Byte 2	0	0	1	1	1	0	1	0	3A
Byte 1	0	1	0	1	0	0	1	0	52
Byte 0	0	0	0	1	1	1	1	1	1F

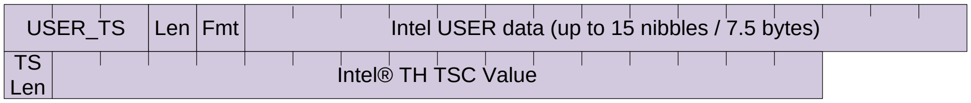
Each encoder in the GTH has the option to create or suppress NULL packets. For output ports that require constant data, NULL packet generation should be enabled; for output ports that do not require constant data, or where “filler” packets are not useful (such as system memory), NULL packet generation should be disabled.

5.2.3.1 Intel Trace Hub hardware-specific USER packet format

The MIPI STPv2.1 protocol allows for a user-defined packet format, called USER or USER_TS, shown in Figure 24. However, the STPv2.1 standard does not specify the contents of the data packet. Since Intel® Trace Hub uses this packet format for a variety of reasons and purposes, and since it is important that both hardware and software easily recognize the various USER packets that Intel® Trace Hub can create, we herein define the standard for the Intel Trace Hub hardware-specific use of the USER packet.

The USER packet format does not apply to software trace sources, which can emit USER packets by writing to the appropriate MMIO location within the STH. These hardware-generated USER packets are easily distinguished from USER packets created by software because the hardware-generated packets are sent to Master 0 Channel 0. No software-generated packets can be sent to this master/channel combination.

Figure 24: STPv2.1 USER packet format

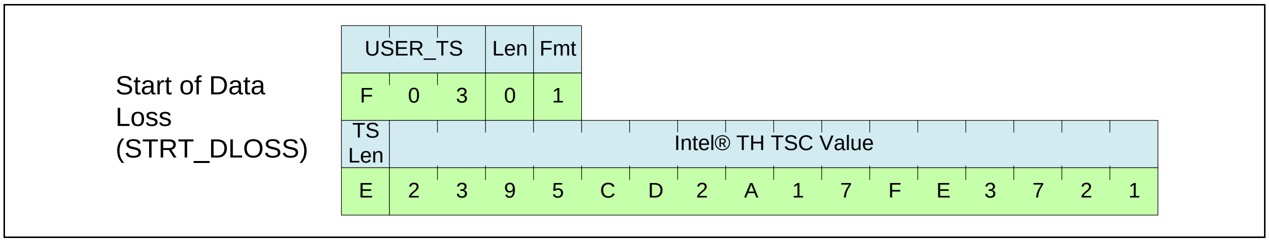


The STPv2.1 USER packet is used to convey two different types of information, or Intel-specific packet types: a Start of Data Loss packet, and an End of Data Loss packet. Exactly when these packets are inserted is the responsibility of each individual trace source. Consult the respective trace source HAS (e.g., STH HAS, ODLA HAS, SoCHAP HAS) for more information.

5.2.3.1.1 Start of Data Loss Packet

The Start of Data Loss packet is a USER_TS packet sent by the STH when it starts dropping incoming data (i.e., when drop mode is on/active). This packet is a USER_TS packet with Intel Trace Hub defined USER format type 1, as shown in Figure 25.

Figure 25: Start of Data Loss Packet

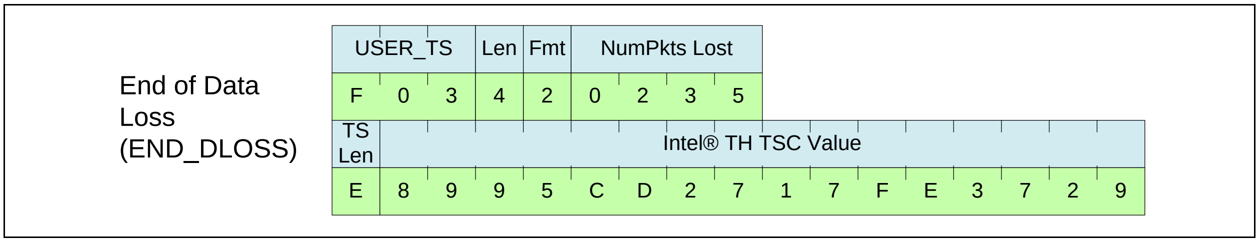


Start of Data Loss (STRT_DLOSS)	USER_TS			Len	Fmt												
	F	0	3	0	1												
	TS Len	Intel® TH TSC Value															
	E	2	3	9	5	C	D	2	A	1	7	F	E	3	7	2	1

5.2.3.1.2 End of Data Loss Packet

The End of Data Loss packet is a USER_TS packet sent by the STH when it resumes normal operation after dropping incoming data (i.e., after drop mode is ended and normal (lossless) operation resumes). This packet is a USER_TS packet with Intel Trace Hub defined USER format type 2, as shown in Figure 26. The End of Data Loss packet indicates the number of packets that were dropped by the STH (not including any TRIG packets that would have been created as a result of a trig_in received from the CTS). It also includes a time stamp so that the downstream software knows when the data loss period ended.

Figure 26: End of Data Loss Packet



End of Data Loss (END_DLOSS)	USER_TS			Len	Fmt	NumPkts Lost											
	F	0	3	4	2	0	2	3	5								
	TS Len	Intel® TH TSC Value															
	E	8	9	9	5	C	D	2	7	1	7	F	E	3	7	2	9

5.2.4 Byte Packing Buffer

The Byte Packing Buffer's function is to map, or pack, the variable number of bytes from the STPv2.1 encoder to a fixed-width output with as few NULL packets as possible.

The BPB also has the ability to drop STP-encoded packets, if specified by the GTHOPT.P*n*DRP bit. If the GTHOPT.P*n*DRP bit is asserted, the trace destination is treated as if it is always ready to receive data. If the downstream block is not ready to receive data, the data will be permanently lost. If GTHOPT.P*n*DRP bit is deasserted, the trace destination will not be presumed to always be ready for data, and the data transfer protocol to the trace destination will be followed. If the BPB fills up, it will assert its *full* signal, thereby back-pressuring the STP encoder and GTH Data Switch.

If NULL packet generation is enabled (by GTHOPT.PnNULL bit), then, for every clock cycle in which the BPB does not have a row of data to send to the trace destination, one byte of NULL packets (two STP NULL packets) will be sent to the trace destination.

Finally, the BPB can be manually flushed using the GTHOPT.PnFLUSH register bit. Manual flushing in this manner is not needed during normal operation, as the hardware initiates BPB Flush operations when they are needed. If the collection of trace data is under “manual control” (such as when using the StoreEnOvrd bits), then the BPB must be flushed at the end of tracing using the PnFLUSH bit to ensure all trace data is flushed to the destination.

5.2.5 STP Maintenance Unit

The STP Maintenance Unit (SMU) is responsible for all aspects of maintaining the STPv2.1 link at the protocol level, per the MIPI STPv2.1 specification. Requirements include:

- periodically sending out ASYNC, VERSION, and FREQ packets
- Sending ASYNC, VERSION and FREQ packets at the beginning of a trace, including whenever a destination port is reset or retrained.

The second requirement is met by sending ASYNC, VERSION, and FREQ packets at the beginning of a trace, defined as the assertion of a *StoreEn* signal. To ease implementation, the SMU is assigned to trace source #0 on the GTH data switch, and the TSCU is trace source #1. This way, the arbiter will start (out of reset) with trace source #0, automatically transition to source #1 after the maintenance packet is sent, and then move on to the data sources. With this approach, the requirements of the MIPI STPv2.1 specification are guaranteed by the architecture.

The Intel® Trace Hub architecture implements one SMU per output queue so that it may send maintenance packets to each output queue at an optimal rate for that output queue. For example, system memory may be capable of 5GB/s of user data (STPv2.1 data), and will likely have maintenance packets inserted every 5us or so. This would be far too often for a USB3 interface, which would need maintenance packets about once every 40us. Each SMU is assigned to the same master number, Master 0, as all “administrative” packets are assigned to this master number in Intel® Trace Hub architecture. The Master number is used only by the downstream MIPI STPv2.1 encoder to set the current Master (and channel) to zero, as required by the MIPI specification. That is, the master number is not used by the arbiter, as there is one SMU per arbiter and output queue.

The SMU also has an *Enable* input signal to control whether it sends trace data (maintenance packets) or not. This is done so that the SMU does not send maintenance packets when all trace sources for the respective output port are turned off (see also section 5.2.9 Automatic Trace Source Enable/Disable). Each SMU’s *Enable* signal is chosen according to its output port. In this way, each SMU is automatically enabled and disabled based on whether there are active trace sources for its respective destination.

The SMU timer is started upon first *get* received from the arbiter, after *portReset* is deasserted. Thereafter, the SMU will decrement as long as its *enable* is asserted and *portReset* is deasserted. If either *enable* or *portReset* is asserted, then the SMU timer is reset to its starting value, and the SMU sends no packets until *enable* is again asserted and *portReset* is deasserted.

When the Maintenance Timer expires, it asserts its “maintenance packet request” output, indicating a request to send a maintenance packet. When the Data Switch arbiter asserts the *get* signal to the SMU Input Buffer, the maintenance packet data set will be driven to the Input Buffer (and Data Switch, since the Input Buffer for the SMU is a zero-depth buffer), and the latched maintenance packet request is cleared.

The other conditions that can cause the SMU to request to send maintenance packets are when a trace source for the SMU’s destination is enabled (*StoreEn* assertion), and when a trace destination port becomes operational. When the *portReset* signal is deasserted, the SMU’s maintenance packet insertion process is triggered.

The content of the SMU maintenance packet data is fixed for a given implementation, with the exception of the GTH_FREQ field. The GTH_FREQ field comes from the GTH_FREQ register. The maintenance packet data and consists of three STPv2.1 packets, as shown in Table 5-2 below.

Table 5-2: SMU Packet Data

Packet Numbe r	Packet type	Packet data	Comments
1	ASYNC	0xFF_FFFF FFFF_FFFF FFFF_FFF0	21 nibbles of 0xF, one nibble of 0x0
2	VERSION	0xF003	0x3 signifies STPv2NAT timestamp format
3	FREQ	0xF08{GTH_FREQ}	

NOTES:

1. Data is shown in network order—the same order as it is sent to the destination. That is, the first nibble, reading left to right, is the first nibble to be sent.

The GTH_FREQ portion of the FREQ packet will depend on the GTH_FREQ register value (which has a default value equal to the GTH_FREQ synthesis parameter). In any case, the GTH_FREQ data is the operating frequency, in Hertz, of the GTH, specified as a 32-bit binary value. For example, 400MHz is 0x17D78400.

5.2.6 Intel® Trace Hub DCI Trace Handler

The Intel® Trace Hub DCI Trace Handler (Intel TH DCI TH) is a trace destination, or output port. While it is a separate output port in concept, it shares output port #0 with the MSU, as it shares a number of functional requirements with the MSU. For example, both the MSU and Intel TH DCI TH send data over the primary bus, and must buffer a certain amount of data before sending it to its final destination. For efficiency and area savings, the MSU and DCI Trace Handler have a single set of memory buffers and output logic between them.

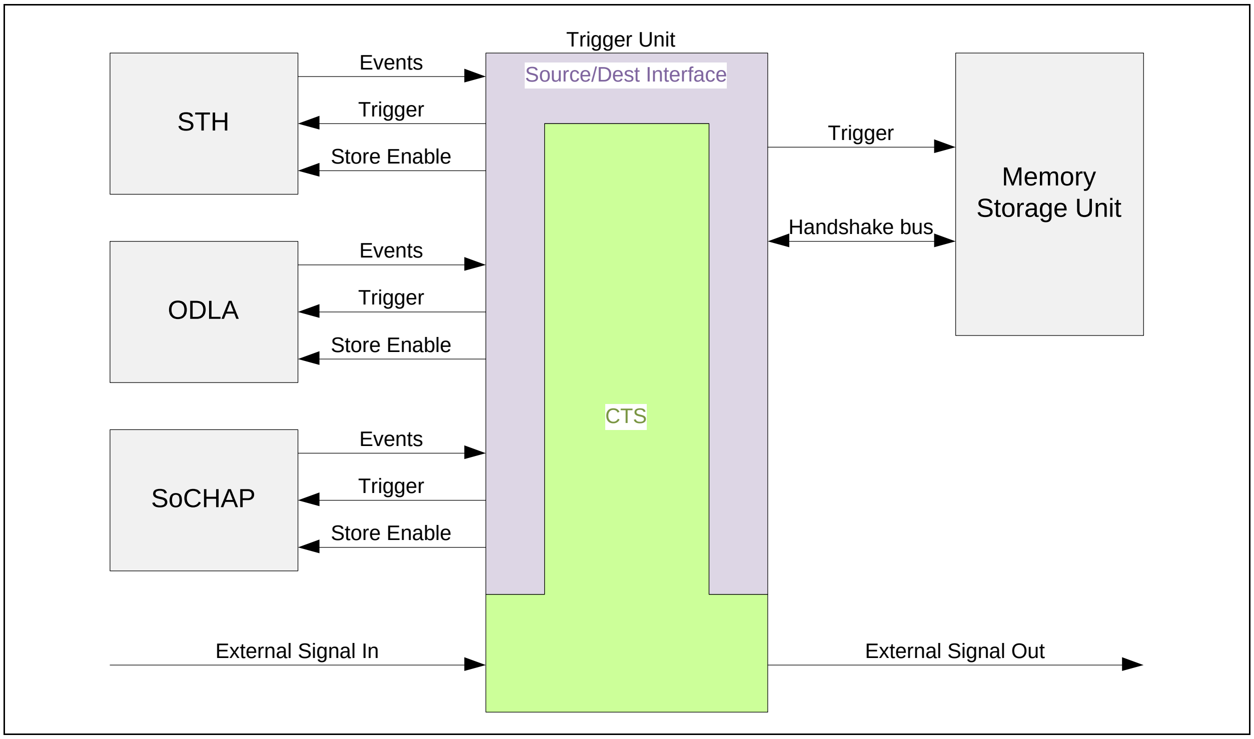
For more information, see the DCI Trace Handler chapter.

5.2.7 Trigger Unit

The triggering capability in Intel® Trace Hub takes “events” and “signals” as input, processes them, and takes actions based on sequences or combinations of these events. To state it slightly differently, events and signals are Trigger Unit state machine inputs, with each state machine state/clause taking separate, user-specified actions such as trigger, start store, stop store, start timer, and so on. The Trigger Unit takes input events from inside Intel® Trace Hub and signals from outside Intel® Trace Hub. Note that there is only one trigger output from the Trigger Unit, and it is routed simultaneously to all destinations (trace sources and MSU).

See the Common Trigger Sequencer in chapter 9   for more information on the trigger sequencer block (CTS).

Figure 27: Trigger Unit



5.2.7.1 Source/Dest Interface Shim

As shown in Figure 27, the Source/Dest Interface shim is located between the CTS and the trace sources, and between the CTS and the MSU. The shim’s purpose is to bridge the *storequal* signals from the CTS to the *storeEn* signals to the sources, and also to implement the “block and drain” functionality required for the MSU’s multi-window functionality.

The mapping of *storeQual* to *storeEn* signals is given in Table 5-1. Since Stor_Qual[0] is reserved for the special function of *captureDone*, and since the SMU and TSCU do not need a *storeEn* driven from the CTS, there is a rather unique mapping of *Stor_Qual* to *storeEn* signals, as shown in the table below.

Table 5-3: Store_Qual to Trace Source Mapping

Stor_Qual #	Destination
0	CaptureDone
1	SoCHAP
2	ODLA
3	STH

The multi-window support requires the shim to "interrupt" (de-assert temporarily) the store enable to the trace sources during a window switch, such that the entire pipeline can drain and switch to the next window. This "interrupting" is only enabled for those trace sources that have their "mode" bit set in the Source Control Register. Further, the "interrupting" of the *storeEn* signals will only happen if the trigger sequencer has reached its "trigger" state/condition. That is, if the sequencer has not yet asserted the *trigger* output, then the de-assertion of *storeQual* signal(s) will not result in a window switch operation; it will only serve to pause trace capture for the duration of *storeQual* deassertion.

When completely stopping trace storage in multi-buffer mode, the *captureDone* signal must be asserted simultaneously with the deassertion of the *storeQual[N]* signals. If the *captureDone* assertion is delayed (even for 1 clock cycle), and if the *Trigger* signal has been asserted, then (for store-to-memory) a window switch will be performed before tracing is stopped and all data is flushed.

The CTS has 16 inputs for events. Fourteen are used by the Intel® Trace Hub for its sources, as follows:

Table 5-4: Trigger Unit Event Connections

Event #	Source	Event #	Source
0	VER* 0	8	SOCHAP 0
1	VER 1	9	SOCHAP 1
2	VER 2	10	STH 0
3	VER 3	11	STH 1
4	VER 4	12	STH 2
5	VER 5	13	STH 3
6	VER 6	14	External
7	VER 7	15	External

* Note: VER = VIS Event Recognizer

The Trigger Unit makes four each Signal_In and Signal_Out signals available to the SoC for use according to the SoC usage needs. Please refer to your product-specific documentation for information on how these signals are connected.

5.2.8 Time Stamp Correlation Unit

The Time Stamp Correlation Unit is a native MIPI STPv2.1 trace source. Since TSCU data is considered “administrative” data, it is always sent using Master 0 Channel 0. The TSCU is connected to the Data Switch using an input buffer with a depth of 1, as it (TSCU) has no ability to store the data it wishes to send.

For more information, refer to chapter 6 .

5.2.9 Automatic Trace Source Enable / Disable

The SMU and TSCU, like all other trace sources, have an *Enable* input signal to control whether they send their trace data (maintenance packets, or time stamp correlation packets, respectively) or not. There is, however, no *StoreQual* signal from the trigger sequencer for these trace sources. Instead, the Intel® Trace Hub automatically controls whether the SMU and TSCU *Enable* signals are asserted or deasserted.

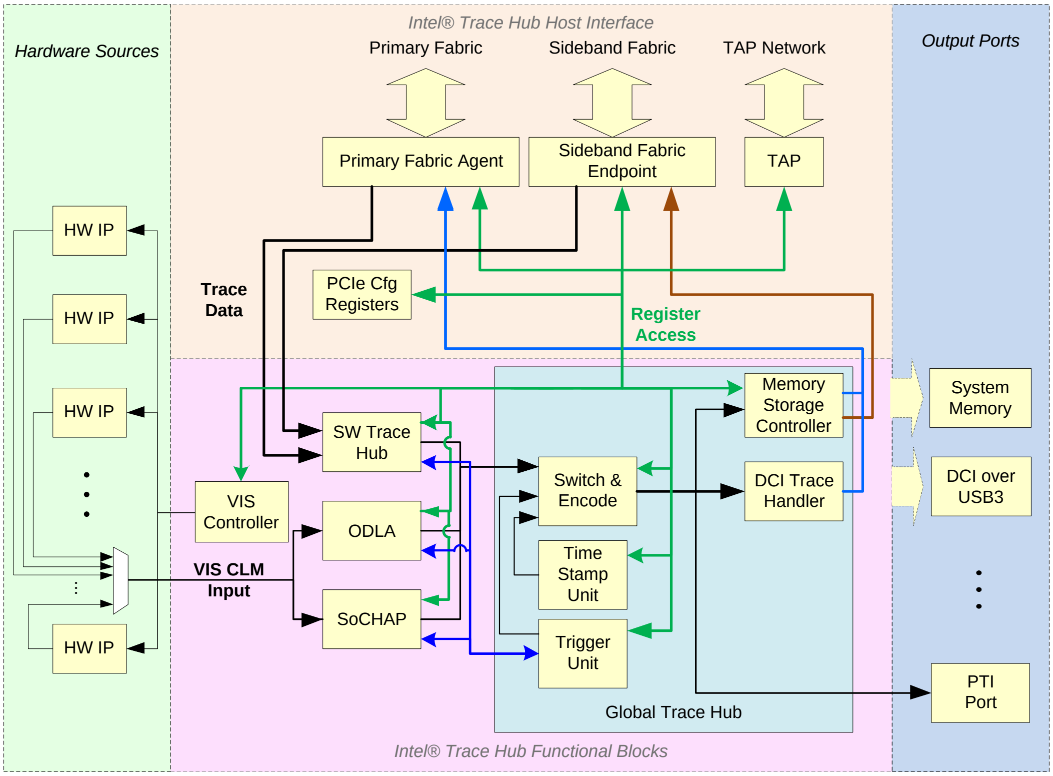
6 Timestamp Correlation Unit

The TSCU provides a common timebase and a trace source timestamp correlation mechanism for Intel® Trace Hub (Intel TH) so that captured Intel TH data can be reconstructed and correlated with CPU-timestamped data. As shown in Figure 28, the TSCU is part of the Global Trace Hub.

The TSCU provides a free running 64-bit timestamp counter that is shared throughout the Intel TH architecture. This timestamp runs on the Intel TH clock and is used for marking native MIPI STPv2 trace source packet's fields as they are created by their respective trace sources.

Intel® Processor Trace is a non-native STPv2 trace source with its own embedded timestamp information, so its data does not directly reference the Intel® Trace Hub timestamp values. Its timing is based on a common SoC timer, operating in an always-on domain, called an ART. The TSCU employs a method for synchronizing and maintaining a local copy of the ART called a Common Timer Copy (CTC). The TSCU periodically snapshots CTC values along with Intel TH timestamp values to create timestamp correlation packets. These packets are then sent to a trace destination where they can be used by post-processing software to time-correlate traces with the other captured trace sources.

Figure 28: SoC timestamp correlation architecture – system view



The TSCU provides the following features and capabilities:

- Provides a free running 64-bit timestamp value to all Intel TH logic based on a dedicated counter running on the common clock.
- Supports two Intel TH clock frequencies for SoC clock switching to a slower clock during low power states. The timestamp counter maintains a consistent timebase in terms of the faster clock frequency regardless of current Intel TH clock frequency.
- Correlation between all trace sources is relative to the shared Intel TH timestamp.
- Creates correlation packets to store snapshot values of the CTC and Intel TH timestamp. These are stored in STPv2 TIME_TS packets. Packets are generated periodically or when requested by GTH to initialize trace source storage buffers. This provides a mechanism to correlate Intel TH trace captures to IA core and software time bases.

6.1 Intel® Trace Hub timestamp

The TSCU maintains the free running 64-bit Intel TH timestamp counter. All trace sources are correlated to the timestamp counter by being explicitly timestamped at the trace source with the current Intel TH timestamp value, or via post-processing software that correlates an embedded common SoC derived timestamp value (ART) to an Intel TH timestamp value (Intel® Processor Trace, STH). The Intel TH timestamp counter is reset by a powergood reset only.

The timestamp counter logic supports switching done externally to the Intel TH. For example, the SoC will normally drive the Intel TH clock at a fixed frequency based on a PLL. During certain low power states, the SoC may switch the clock driving into the Intel TH clock input to use a slower slow clock. The Intel TH timestamp counter includes logic to provide a consistent timebase

6.2 CTC tracking unit

The CTC tracking unit in the TSCU is used to track the SoC's Always Running Timer (ART). The ART is driven by a stable, always-running clock that has a fixed frequency without SSC modulating its source. The SoC's CPUs use the ART as the reference for generating their local TSC values after clock crossing and tracking/scaling logic. Each core maintains its own local, fast-domain TSC that is synchronized to each other, even after returning from a powered down state, so that a coherent timestamp is available for reference. The core's local TSC value is what is returned when a RDTSC instruction is executed, for example. The ART is also the base for the timestamp values embedded in Intel® Processor Trace traces.

Intel TH tracks the ART value so that Intel® Processor Trace traces can be time correlated with other trace sources.

6.3 TSCU packet control logic

Samples of the current CTC, the Intel TH offset counter, and the Intel TH timestamp counter values are taken simultaneously to create snapshots of their values. These sampled values are stored in timestamp correlation packets that are sent to a trace destination by the GTH. The counter sampling occurs only after CTC initialization has been successfully performed, and only while the TSCU *storeEn* signal is asserted. Additionally, sampling occurs:

- Periodically based on when a selectable CTC bit toggles
- Whenever directed via CSR write to a SWForceSample register bit field
- Whenever the *forcesample* TSCU input signal asserts
- On the rising edge of *storeEn*

Timestamp correlation packets are sent to the GTH using the TIME_TS packet format. The resultant STPv2 packet for the TIME_TS packet is as follows:

TIME_TS = 0xF 0x0 0x5 clock_domain other_ts_format TS_other TS

Where:

clock_domain = tsClkDom = 0x1

other_ts_format = 0x4

TS_Other = npkoffset[7:0], ctc_value[55:0]

TS = npk_timestamp[63:0]

The other_ts_format is not strictly called-out or specified in the STP v2.1 specification, but it is allowed. This is because the other_ts_format is not defined for clock_domain != 0. Since the TSCU correlation packets are assigned to clock_domain = 1, we are free to use any value for the other_ts_format. To avoid confusion with existing definitions of ts_format, we chose format = 0x4.

TIME_TS packets can store the full 56-bit CTC value, 8-bit Intel TH offset and 64-bit Intel® Trace Hub TSC in a single packet. The mapping of counter values to data field bits is shown in Table 6-1.

Table 6-1 TIME_TS data field mapping

data																dsz
63:60	59:56	55:52	51:48	47:44	43:40	39:36	35:32	31:28	27:24	23:20	19:16	15:12	11:8	7:4	3:0	3:0
npkoffset		ctc_value[55:0]														F

6.4 Software processing

Trace sources such as ODLA, SoCHAP, and STH are timestamped with Intel TH timestamp values internally at the source, and will not need to use the correlation packets to determine their relative timing. However, post-processing software is required to extract the CTC, Intel TH offset, and Intel TH timestamp values from timestamp correlation packets and process them to provide timing correlation between the Intel® Processor Trace and the other trace sources. They are also needed to correlate between any RDTSC instruction results that may need to be correlated to Intel TH captures or any other non-Intel TH trace source that has a timestamp derived from ART values.

The first step involves building a timestamp correlation table that maps CTC and Intel TH clock offset values to Intel TH timestamp values as shown in Table 6-2. This table is populated with the contents of decoded timestamp correlation packets with each entry sorted in increasing time order. Additionally, a coarse core TSC value can be derived based on each table's CTC entry multiplied by scaling factor, P, which is a constant integer scaling factor corresponding to the SoC's guaranteed frequency.

Table 6-2. Example timestamp correlation table

Correlation Point	Intel® TH TSC	CTC	Intel® TH Offset	Coarse Core TSC
N-3	NTSC_Nm3	CTC_Nm3	ITHOS_Nm3	CTC_Nm3 * P
N-2	NTSC_Nm2	CTC_Nm2	ITHOS_Nm2	CTC_Nm2 * P
N-1	NTSC_Nm1	CTC_Nm1	ITHOS_Nm1	CTC_Nm1 * P
N	NTSC_N	CTC_N	ITHOS_N	CTC_N * P
N+1	NTSC_Np1	CTC_Np1	ITHOS_Np1	CTC_Np1 * P
N+2	NTSC_Np2	CTC_Np2	ITHOS_Np2	CTC_Np2 * P
N+3	NTSC_Np3	CTC_Np3	ITHOS_Np3	CTC_Np3 * P

Intel® Processor Trace TSC packets contain the core's current time stamp counter value. They are sent on the Intel® Processor Trace enabling as well as on key conditions such as core frequency changes and sleep state wakeup. The TSC in these packets are not directly associated with values stored in the Intel TH timestamp correlation packets, but they are related by an equation of the form:

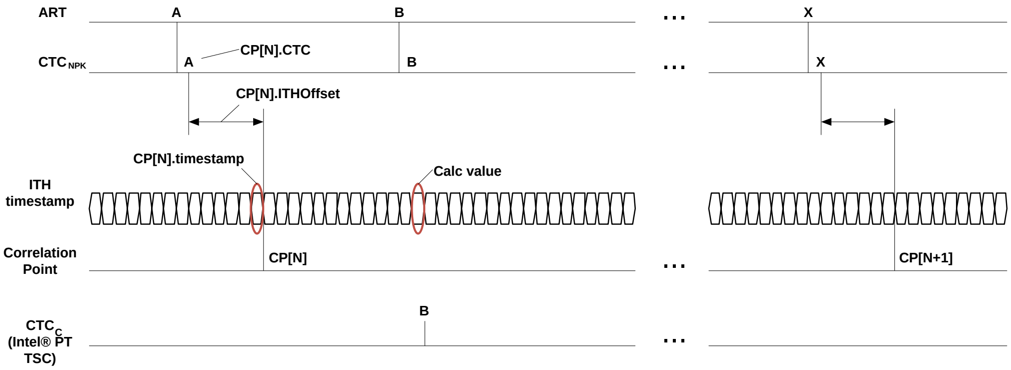
$$TSC_C = CTC_C * P + fastcounter + sw\ offsets$$

where P is a scaling factor that corresponds to the SoC's guaranteed frequency. The TSC value stored in the Intel® Processor Trace TSC packet can be correlated to an Intel TH timestamp value within the resolution of the ART/CTC given a known value of P. To convert a CTC value embedded in an Intel® Processor Trace TSC packet to an Intel TH timestamp value, find the correlation point with the closest CTC value to the CTC value stored in the Intel® Processor Trace TSC packet. If the CTC value obtained from the Intel® Processor Trace TSC packet is greater than the closest correlation point's CTC value, then use the following equation to calculate the corresponding Intel TH timestamp value.

$$ITHTSCCalc = NTSC_N - NPKOS_N + (CTC_C - CTC_N) * \frac{XCLKPER}{ITHCLKPER}$$

Where ITHCLKPER is the nominal Intel TH clock period and XCLKPER is the period of the clock that is driving the ART. Figure 29 is a waveform showing the relationship of these terms graphically.

Figure 29. Conversion from CTC values to Intel TH timestamps



If the closest matching correlation point's CTC value is greater than the CTC value obtained from the INTEL® PROCESSOR TRACE TSC packet, instead use the following equation.

$$ITHTSCCalc = NTSC_N - NPKOS_N - (CTC_N - CTC_C) * \frac{XCLKPER}{ITHCLKPER}$$

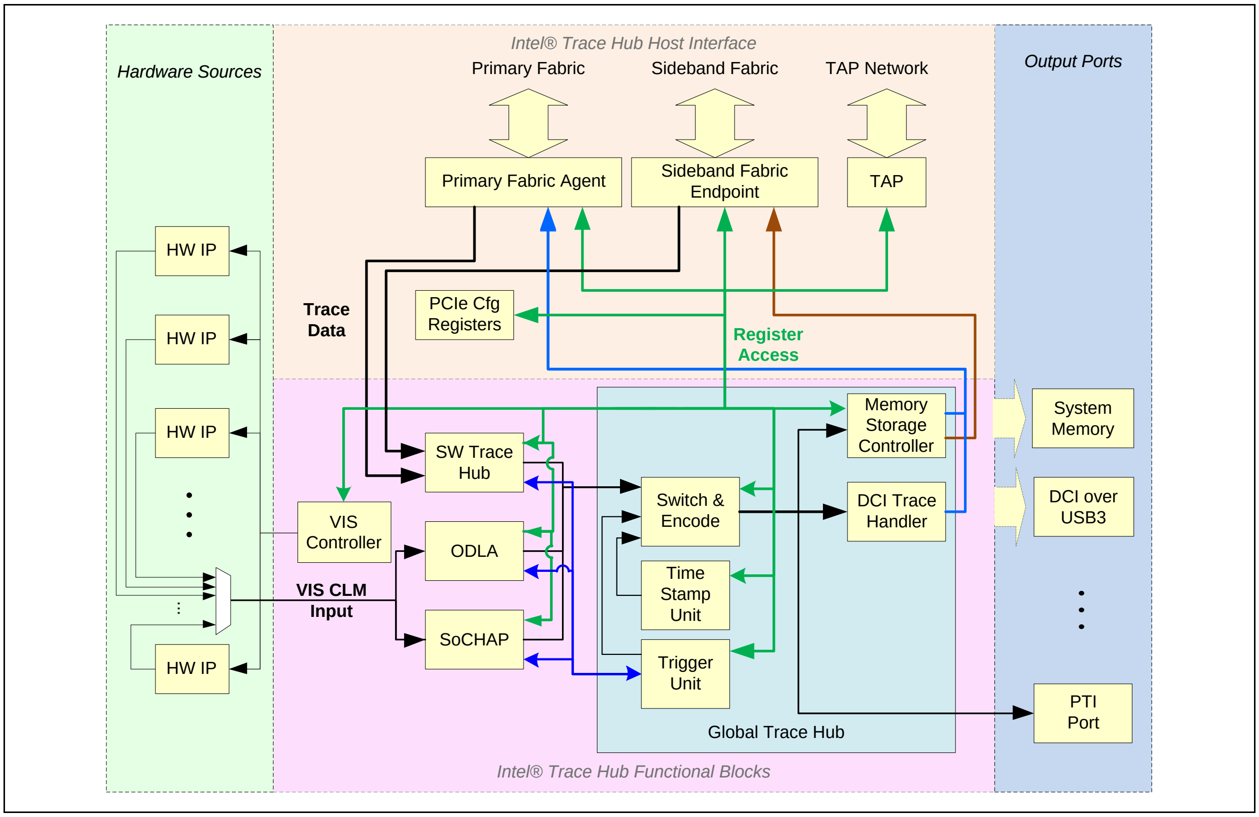
The overall calculation accuracy is a function of how frequently correlation packets are created. The TSCU's sample frequency should be increased whenever tighter time correlation is required.

Intel® Processor Trace traces also contain Mini Time Counter (MTC) packets that include a subset of bits from the core CTC. These packets are periodically inserted into the Intel® Processor Trace at the source based on a selectable rate that is a scaled value corresponding to the ART's rate. The above equations can be used with MTC packet contents given a CTC value from an Intel® Processor Trace TSC packet that has been stored prior to the MTC packet. This is required to provide the missing CTC bit values to combine with the MTC value.

7 Memory Storage Unit

This section defines the high level architecture of the Intel® Trace Hub Memory Storage Unit and its components. Figure 30 shows the MSU in the context of the Intel TH block.

Figure 30: Memory Storage Unit in Intel® Trace Hub



7.1 Functional description

7.1.1 MSU overview

The MSU is responsible for taking trace data from the Byte Packing Buffers, temporarily storing it in its own trace buffers, and routing the data to system memory.

The MSU is comprised of two Memory Storage Controllers and an MSU arbiter to arbitrate and control their access to the primary fabric. Two MSCs are required in order to meet the requirements of the “windowed” trace capture use cases supported by the Intel® Trace Hub.

To ensure good performance, the MSCs wait until a full cache line of data has accumulated in the MTBs before writing that data to system memory. Only when in a “flush” condition (e.g. a window switch, or an end-of-tracing condition), or when specifically configured via the MSCnCTL.MSCnLEN register field does an MSC write less than a cache line of data.

7.1.2 MSU operational modes

The Memory Storage Unit operates in four modes. Single block mode (or CSR-driven mode), multi-block mode (or linked-list mode), DCI trace mode (or Intel TH DCI trace handler mode), and Internal Buffer mode. All operating modes have an initialization stage, a storage stage, and a retrieval stage. In the initialization stage, software initializes the CSRs and memory blocks (if applicable) with information required by the hardware for the storage stage. When software is done initializing the MSU (i.e., one or both MSCs), it moves to the storage stage by configuring the trigger unit according to the needs of the user, and then setting the CTS control register Sequencer_Enable bit. Hardware performs the tracing and storage operations, and then sends an interrupt to software (see msu_interrupt description) to indicate it is finished (see CaptureDone description), and data is available.

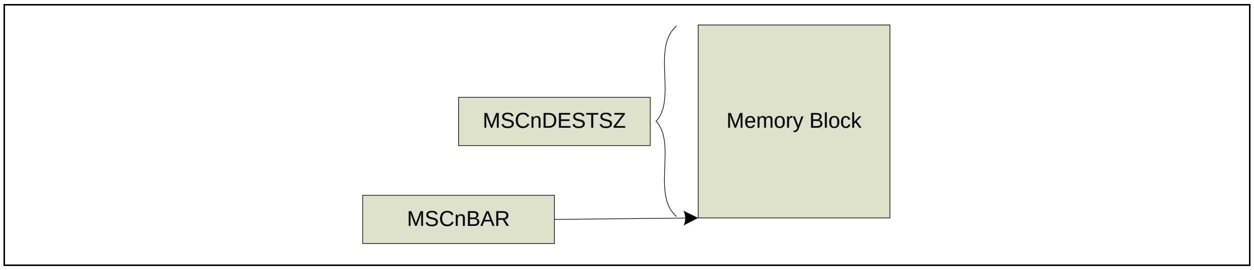
Under normal conditions, the MSCs are reset at power on. The MSCs support multiple captures in different operating modes, however there is no soft reset mechanism implemented in the MSCs to aid in this transition. For this case, the MSC requires the current capture to complete properly with no residual data, then the MSCnEN must be de-asserted, then the MSC must be re-initialized and finally re-started in the new mode by setting MSCnEN.

The four operational modes and the memory block header formats are described in the followed sections.

7.1.2.1 Single block or CSR-driven mode

The single block mode, also known as the CSR-driven mode, shown in Figure 31 is the simple and straight-forward mode. The user programs the MSC CSRs with the starting memory address, size, and wrap settings. This allows the user to store trace data to memory with a minimum of setup information. In single block (CSR-driven) mode, the MSU operation is simplified and is not responsible for performing a memory read and a memory write of header information since this information exists in CSRs. The single-block mode requires the user to dedicate a portion of memory for the trace buffer that is not visible to the OS.

Figure 31: MSC single block or CSR-driven mode



Note: If wrapping is not enabled for Single Block mode, then when the memory block/buffer is filled, the MSC will stop writing data to memory, and will apply upstream backpressure. Because the MSC will backpressure the upstream pipeline, the pipeLineEmpty bits will never be asserted, because the data pipeline cannot be emptied. Software must account for this case when stopping tracing operations.

7.1.2.2 Multi-block or linked-list mode

The multi-block or linked-list mode allows the user to store trace data to system memory while the system is under the control of an OS. For this mode, one or more blocks of memory is allocated, either by BIOS, or by a process running under the operating system. A memory “block” in this context is a single, physically contiguous portion of memory. The same software (BIOS or OS) then initializes the memory block with header information describing the block and its relationship to other blocks; notably each block contains two addresses linking the block to neighboring blocks. This is similar to descriptor tables used in/by USB and ethernet drivers. Once the linked list is initialized, the software writes the starting address of the first memory block to the MSCnBAR registers. Upon setting MSCnEN, the MSC fetches the first descriptor in preparation for storing trace data to system memory.

7.1.2.3 DCI Trace Handler mode

In the DCI trace handler mode, the two trace buffers in the MSU are utilized as a single combined buffer. In this mode, only trace destination #0 is valid (trace destination #1, or MSC #1, is an invalid destination, even though MSC1 must be set to DCI Trace Handler Mode).

The Intel TH DCI trace handler is responsible for the data once it is in the trace buffers. The DCI trace handler has two sub-modes of operation: DbC.Trace mode, and BSSB mode.

In DbC.Trace mode the two MSC’s act as a memory resource for the DCI Trace Handler, which handles all the interaction with the DbC.Trace endpoint. The DCI trace handler waits for the trace buffer depth to reach a certain threshold and then programs the DBC (using memory write cycles) to perform a DMA transfer. Once programmed, the DBC starts the DMA transfer of the trace data and these transactions appear to the MSU as inbound memory read cycles, where the trace data is then transferred to the DbC.Trace endpoint.

In BSSB mode, the DCI trace handler initiates the transfer of data from the unified trace buffer as memory-mapped write commands to the DCI Bridge.

7.1.2.4 Internal Buffer mode

The Internal Buffer mode of operation is a mode where the trace buffers will capture trace data without transporting it out to main memory. This mode is useful for debug of a system where main memory is not available. Read transactions that target the trace buffers will always return a single a QWORD (64 bits) data quantity, and bursting is not supported. These reads are only supported in Internal Buffer mode and are ignored in other modes. Writing of the trace buffers from the primary or sideband fabric is not supported in any operational mode.

The wrap configuration bit will be used to determine if the MSU should wrap or halt trace capture at the end of the trace buffer.

Note: If wrapping is not enabled for Internal Buffer mode, then when the internal buffer is filled, the MSC will stop storing data to the internal buffer. Because the MSC will backpressure the upstream pipeline, the pipeLineEmpty bits will never be asserted, because the pipeline cannot be emptied. Software must account for this case when stopping tracing operations.

The reading of the trace buffers in internal buffer mode can be from the primary interface, sideband interface, or the TAP. Address bit 15 used to select between MTB0 and MTB1.

Note: Reading of the MTBs in internal buffer mode while trace data is being stored in the MTBs is not supported. There is currently no way to prevent the MTB from under-run (reading more data than is actually present). Furthermore, the WRAPSTAT will not function properly, due to the fact that reading from the MTB changes the read pointer value, which is used in the full and empty calculations.

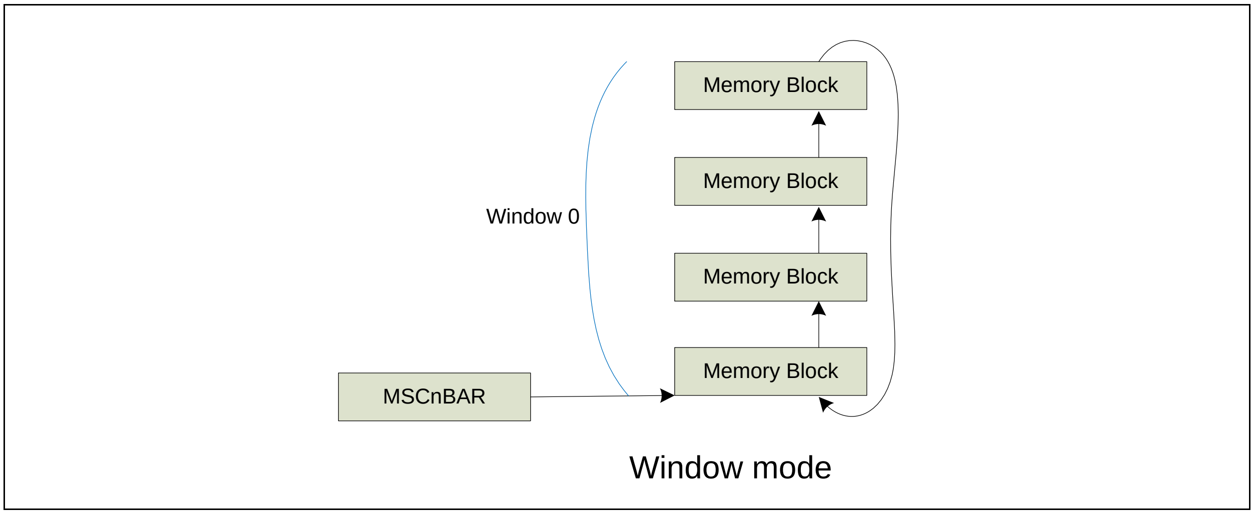
Note: When switching to internal buffer mode from another operational mode, some trace data will be lost in the “preload” buffer. This is because, in a “normal” operational mode (anything other than internal buffer mode), data from the MTB is loaded into a "preload" buffer so that it can be immediately sent to the primary bus when the it is granted to Intel® Trace Hub. The preload for each MSC buffer is one stage deep, and cannot be read. Thus, the first 8 bytes of trace data will not be available for extraction when switching from a normal operational mode to Internal Buffer mode.

7.1.3 Multi-block memory windows

Memory blocks can be chained together through programming of the software header. A collection or chain of memory blocks, all with the same next window starting address is called a memory window.

Figure 32 gives a visual representation of the multi-block memory window concept.

Figure 32: Single memory window with multiple blocks

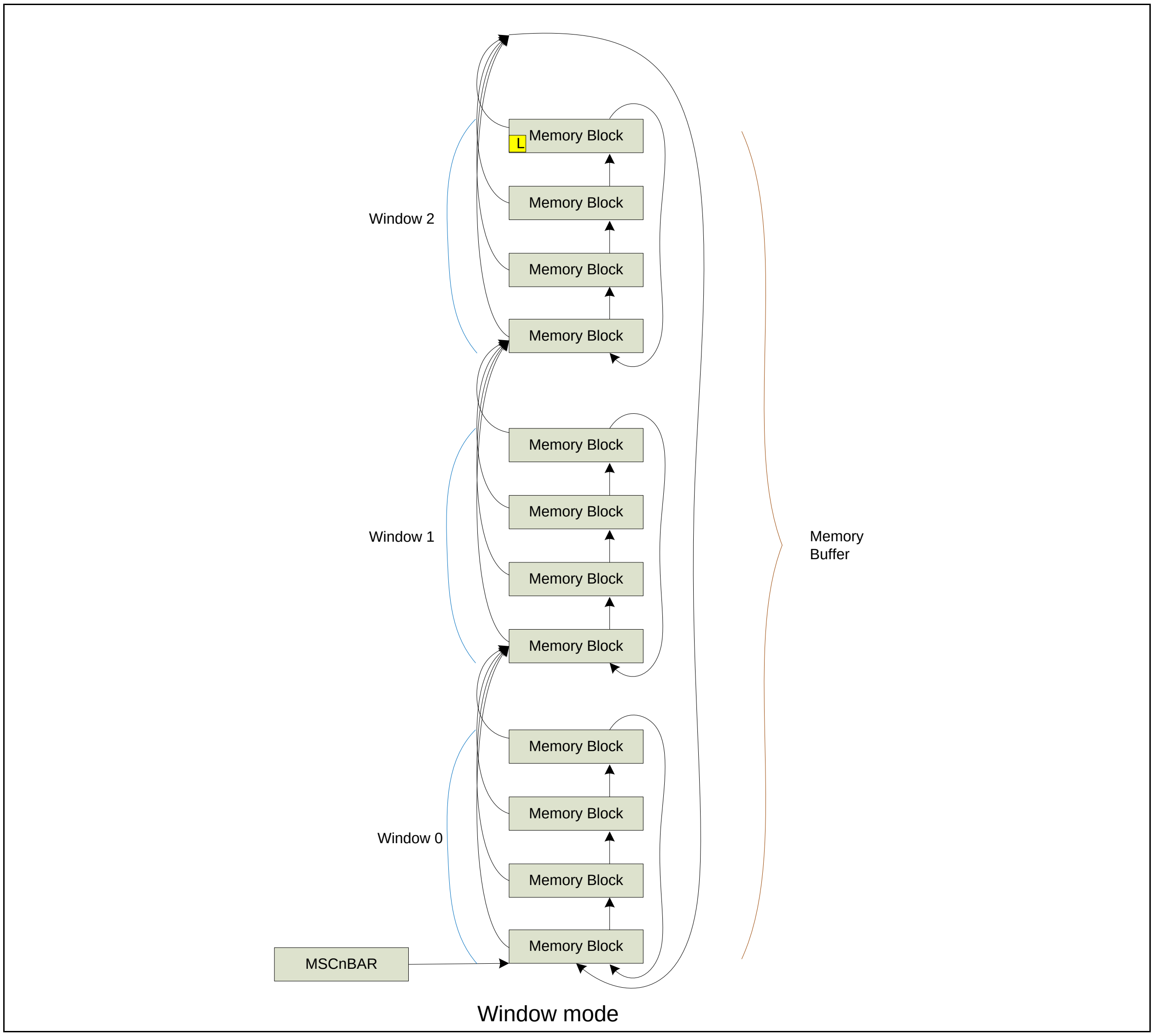


7.1.4 Multi-window memory buffer

Multiple windows can be chained together through programming of the software header. The collection of memory windows together makes up the memory buffer for a given MSC. Multi-block windows are supported to enable the Intel TH to use OS-managed memory. Contemporary operating systems allocate memory blocks in increments of 4 kB, and don’t guarantee delivery to any single block that is more than 4 kB, or to any multiple blocks that are contiguous; multi-block windows are, therefore, a necessity so that the MSC can store trace data in buffers larger than 4 kB.

Figure 33 gives a visual representation of the multiple chained windows concept.

Figure 33: Multiple memory windows



Repeated storage is the intended use case for multiple windows within a single memory buffer. That is, the user desires to store data before and/or after a given event (trigger), and to do this several times in a single capture session. To accomplish this, the Intel TH trigger unit is configured to assert the *storeEn* signal for one or more trace sources, and begin watching for the trigger event. During this time, the MSC is storing data into a single window (wrapping is always on in this mode). When the trigger condition is reached, the trigger unit begins a post-trigger fill period (the length of time of which is determined by a counter/timer). When the post-trigger fill period is reached, the trigger unit (as it was previously configured) de-asserts the *storeEn* signals to all trace sources sending data to the MSC(s) operating in multi-window mode. This de-assertion of the *storeEn* signals is interpreted by the MSU as a “switch window” command. Once all in-flight data is written to system DRAM, the MSC(s) switch to the next window (where they might, for example, begin writing to memory at MSCnNWSA, specified in the then-current memory block header).

7.1.5 Memory block header description

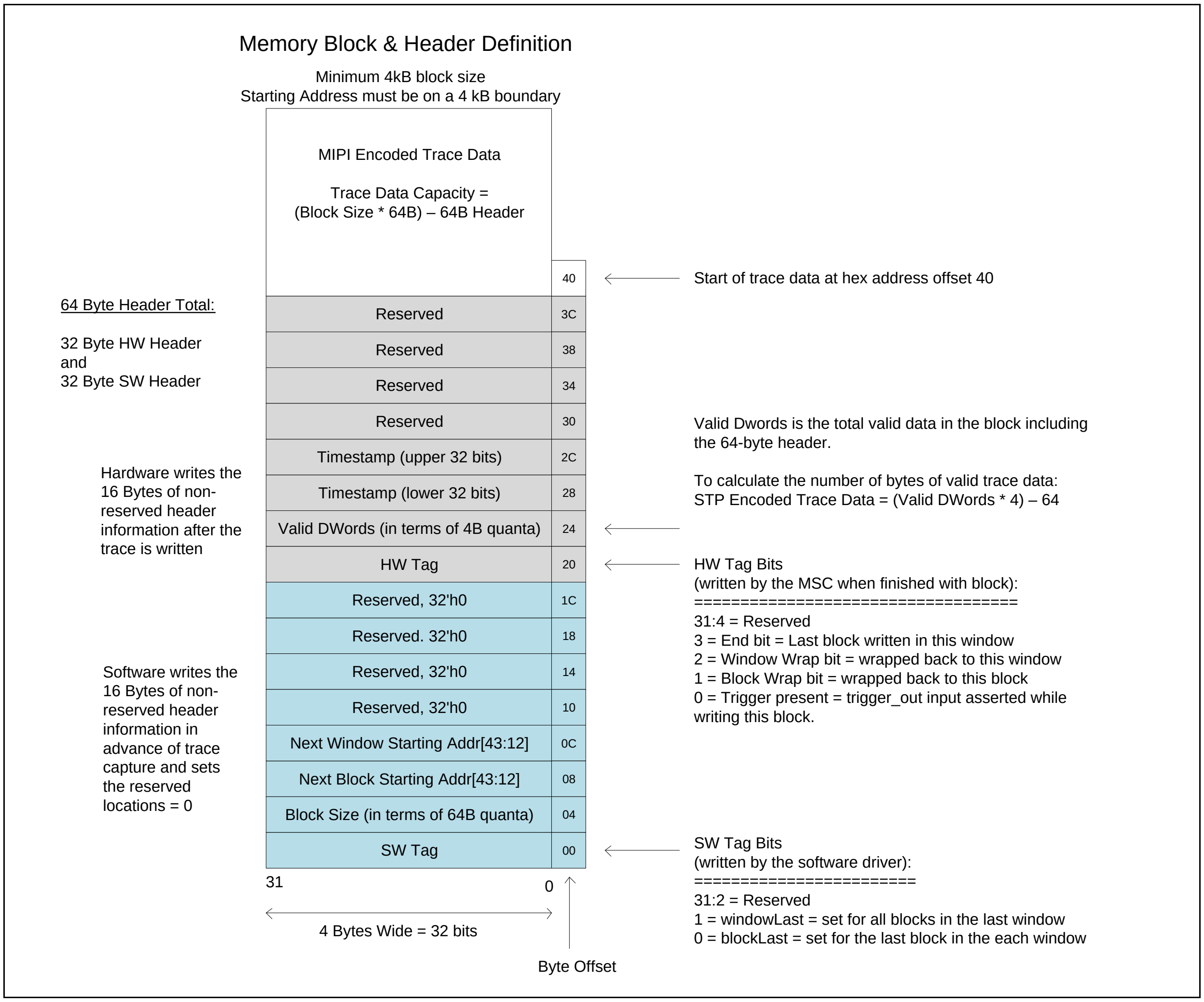
The memory block in single block (CSR-driven) mode has no header information, and can thus utilize 100 percent of its storage for storage of trace data.

For multi-block mode, the first 64 bytes of each memory block are reserved for header information. The header is divided into two areas: the first 32 bytes are reserved for software configuration, and the second 32 bytes are written by the MSC hardware. The software headers contain the information required to chain together the blocks into windows and buffers, as well as some descriptive ‘tags’ that describe the block. The driver software is responsible for gaining block allocation from the OS, and writing the software header information. The pointer to the first memory block must be put into the MSCnBAR registers.

The MSC will write the hardware header information when it is finished writing to the block.

Figure 34 shows the memory header topology within the memory block.

Figure 34: Memory Block Description showing header



Memory Block & Header Definition

Minimum 4kB block size
Starting Address must be on a 4 kB boundary

MIPI Encoded Trace Data

Trace Data Capacity =
(Block Size * 64B) – 64B Header

Field	Byte Offset
	40
Reserved	3C
Reserved	38
Reserved	34
Reserved	30
Timestamp (upper 32 bits)	2C
Timestamp (lower 32 bits)	28
Valid DWords (in terms of 4B quanta)	24
HW Tag	20
Reserved, 32'h0	1C
Reserved. 32'h0	18
Reserved, 32'h0	14
Reserved, 32'h0	10
Next Window Starting Addr[43:12]	0C
Next Block Starting Addr[43:12]	08
Block Size (in terms of 64B quanta)	04
SW Tag	00

64 Byte Header Total:

32 Byte HW Header
and
32 Byte SW Header

Hardware writes the 16 Bytes of non-reserved header information after the trace is written

Software writes the 16 Bytes of non-reserved header information in advance of trace capture and sets the reserved locations = 0

Start of trace data at hex address offset 40

Valid Dwords is the total valid data in the block including the 64-byte header.

To calculate the number of bytes of valid trace data:
STP Encoded Trace Data = (Valid DWords * 4) – 64

HW Tag Bits
(written by the MSC when finished with block):
==================================
31:4 = Reserved
3 = End bit = Last block written in this window
2 = Window Wrap bit = wrapped back to this window
1 = Block Wrap bit = wrapped back to this block
0 = Trigger present = trigger_out input asserted while writing this block.

SW Tag Bits
(written by the software driver):
========================
31:2 = Reserved
1 = windowLast = set for all blocks in the last window
0 = blockLast = set for the last block in the each window

7.1.5.1 Software header description

The software header contains the following 32-bit values:

0x0: The 32-bit **SW Tag** implements the following bits. All unused bit locations must be written with zeros, and are to be ignored when reading.

- **Window Last** in bit position 1 indicates that the given block is in the last memory window. This bit should be set for all blocks in the last window, and written with zeros for all blocks not in the last window. This is used by the MSC hardware to know when it has wrapped past the last window to know when to set the window wrap bit.

- **Block Last** in bit position 0 indicates that the given block is the last block in this window. The MSC hardware uses this to determine when to set the block wrap bit in single buffer mode.

0x4: Block Size defines the size of the current memory block. The value programmed in the block size is multiplied by 64 to represent the size of this block in bytes. The amount of trace data that can be captured is equal to this Block Size * 64, less the 64 bytes of header. The minimum block size supported by the MSU is 4 kBytes. The capacity of Valid DWords is = 2^32 DWords = 16 GB, so block sizes should be kept to a maximum of 16 GB.

0x8: Next Memory Block Starting Address is the physical address that points to the next block within this window. The value programmed in the next memory block starting address is left-shifted by 12 bits to determine the physical address. Because of this 12 bit shift, all blocks must start on 4 KB boundaries.

0xC: Next Window Starting Address is the physical address that points to the starting address of the first memory block in the next window when working in multi-window mode. Each memory block of the current window should have programmed into it the same value for Next Window Starting Address. This is because the MSU can jump to the next window from any block within the current window. The MSU will read the current block header and use this information, and does not save the header information from previous block headers. The value programmed in Next Window Starting Address is left shifted by 12 bits to determine the physical address. Because of this 12 bit shift, all blocks must start on 4 KB boundaries.

7.1.5.2 Hardware header description

The hardware header contains the following 32-bit values:

0x20: The 32-bit **HW Tag** implements the following bits. All unused bit locations must be written with zeros, and ignored when reading.

- **End bit** in bit position 3 indicates that this memory block is the last block written with trace data in this window. That is, the trace capture was halted by either SwitchBuffer or CaptureDone and the last data for the window/buffer was stored in this memory block. Software uses the end bit to know where the end of the trace is. The beginning of the trace is also in this block if the block wrap bit is set. If the block wrap bit is not set, then the beginning of the trace is at the start of the window.
- **Window Wrap bit** in bit position 2 indicates the trace that is stored wrapped back to the first memory window. Once the MSU wraps back to the first window, it sets all window wrap bits in all HW headers.
- **Block Wrap bit** in bit position 1 indicates the trace that is stored in the window wrapped back to the first block in this memory window. That is, more data was stored to the window than fits the size of the window, so the trace wrapped past the end back to the beginning of the window. Once the MSU wraps back to the start of the window, it will set all block wrap bits in all HW headers in this current window. If this bit is set, the meaning of the number of valid DWords field is altered. When block wrap = 1, valid Dwords indicates the position of the newest data in the memory window/buffer and valid Dwords + 1 indicates the position of the oldest data in the memory window/buffer. There is a corner case where the user could configure only one window. A window wrap would also be a block wrap. In this case, the MSU sets the window wrap bit.
- **Trigger Present** in bit position 0 indicates that the CTS asserted its Trigger_out output to the system while the MSU was storing data to this memory block. The

> Trigger_out is a very fast sideband signal to the MSU. Any action taken by ODLA to mark the trace data with this trigger_out indication will likely be delayed from this point in time due to the buffering of data along the Intel TH pipeline, including the trace buffer in the MSC. Due to this buffering, it is very possible that the data being captured by ODLA at the time of the Trigger_out assertion could be stored in subsequent memory blocks.

0x24: Valid DWords (in terms of DWords) indicates the total number of valid DWords in the memory block. To get a byte count, multiply valid DWords by 4. This result includes the 64 bytes of software and hardware header information. To calculate the number of bytes of valid trace data (the MIPI Encoded Trace Data) = (valid DWords * 4) – 64.

In general, if the block fills with valid trace data, and the MSU exits to the next block at the end of the current block, then valid DWords will match the block size in DWords.

The exception case is if the MSU switches windows (due to switchBuffer) thus causing it to exit before reaching the end of the block; in this case, valid DWords * 4 will be the offset to the next data that it would have written. In this case, SW can read valid DWords, and multiply it by 4 which can be added to the block's starting address to calculate the physical address of the oldest trace data if block wrap = 1, or to one past the end of the valid trace data if block wrap = 0. This latter calculation would also be used if the trace capture halted before the end of the current block due to CaptureDone.

0x28: Timestamp lower is the lower 32-bits of the TSCU timestamp. The TSCU timestamp counter is captured continuously by the MSU. This timestamp value is written by the MSU into each hardware header when it has completed writing the trace to the memory block. The timestamp information in the chain of memory blocks can be quickly scanned by the user post trace capture to navigate through the memory blocks and windows and to find a point in time that is interesting to the user.

0x2C: Timestamp upper is the upper 32-bits of the TSCU timestamp counter.

7.1.6 Operational details

7.1.6.1 MSCnMODE, wrap enable and wrap status behavior

Table 7-1 lists the relationships between the WRAPENn, WRAPSTATn, block wrap, window wrap and msu_odla_memwrap output for each operational mode of the MSC. Note that the msu_odla_memwrap signal is an output to the ODLA logic that indicates the memory buffer has wrapped, thus potentially over-writing trace data containing the last timestamp. For this reason, ODLA will wake if it's in a deep compression mode, and insert a timestamp packet into the trace.

Table 7-1: wrap enable, status, and signal relationships for each operational mode

MSCnMODE CSR	WRAPENn bit	WRAPSTATn status register behavior	BlockWrap and WindowWrap HW header bits	msu_odla_memwrap output signal
00 Single Block or CSR Driven Mode	WRAPENn = 1 enables the MMI to MSCnSIZEstore more trace data than can be stored in one pass as indicated by MSCnSIZE. The MSU will "wrap" the memory write address back to MSCnBAR thus creating a circular trace buffer that will over-write older data at the start of the buffer with newer data. The trace buffer will then contain data at the end of the trace capture. When WRAPENn = 0, the MMI will stop writing trace data at MSCnBAR + MSCnSIZE. When it stops writing, it will set the MSU_INT bit in the MSU_STS register, and (naturally) backpressure the entire upstream data path. Thus, the pipeline cannot drain (cannot reach empty) in this case. WRAPENn controls the memory address wrap in Single Block mode, and thus wrapping the Trace Buffer write pointer past MTBDEP by the MSC Source Interface is always enabled in this mode.	WRAPSTATn will be set to 1 when WRAPENn = 0 and the memory block is filled (that is, when MSCnSIZE*4k bytes or more are written). Else WRAPSTATn = 0. WRAPSTATn = 1 when WRAPENn = 1 and MSCnSIZE * 4k bytes, or more, are written to the memory buffer. WRAPSTATn will remain set to 1 until MSCnEN = 0 or until reset is asserted, which will clear WRAPSTATn.	Undefined for this mode	The MMI asserts a pulse on msu_odla_memwrap each time it wraps past MSCnBAR + MSCnSIZE
01 Multi Window Mode	Block and Window wrapping is always enabled in multi window mode.	Undefined for this mode.	If the MSU wraps past the last block of a window as indicated by the blockLast bit, then all subsequent BlockWrap HW header bits will be set while in that window. If the MSU wraps past the last	The MMI asserts a pulse on msu_odla_memwrap each time it wraps past the end of the last block in the window as indicated by the block last bit. The MMI also asserts a pulse on msu_odla_memwrap after it exits a window when the WindowLast bit is set.

MSCnMODE CSR	WRAPENn bit	WRAPSTATn status register behavior	BlockWrap and WindowWrap HW header bits	msu_odla_memwrap output signal
			window back to the first window as indicated by the windowLast bit, then all subsequent WindowWrap HW header bits will be set.	
10 DCI Trace Handler Mode	WRAPENn enables wrapping the Trace Buffer write pointer past MTBDEP by the MSC Source Interface. This should be set by the SW.	WRAPSTATn will be set to 1 when the Trace Buffer write pointer wraps past MTBDEP. This should be the normal operating condition. WRAPSTATn will remain set to 1 until MSCnEN = 0 or until reset is asserted, which will clear WRAPSTATn.	Undefined for this mode	Not used. msu_odla_memwrap should always be driven to 0 when in DbC.Trace mode.
11 Internal buffer mode	WRAPENn enables wrapping of the Trace Buffer write pointer past MTBDEP by the MSI in dubug mode.	WRAPSTATn = 1 if the Trace Buffer write pointer wraps. That is, if MTBDEP * MD_WIDTH bytes (or more) are written, WRAPSTATn will be set. WRAPSTATn will remain set to 1 until MSCnEN = 0 or until reset is asserted, which will clear WRAPSTATn. These conditions are valid only if the MTB(s) are not read while trace data is being sent to the MSC(s).	Undefined for this mode	The MSC Source Interface asserts a pulse on msu_odla_memwrap each time it wraps past the end of the Trace Buffer.

7.1.7 Intel TH DCI trace handler

The Intel TH DCI trace handler (DCI trace handler) is a trace destination, or output port within the MSU. While it is a unique output port from a capability viewpoint, it shares a number of functional requirements with the MSU, and so is tightly coupled to the MSU architecture. For example, both the MSU and the DCI trace handler must buffer a certain amount of data before sending it to its final destination. For efficiency and area savings, the MSU and DCI trace handler have a single set of memory buffers and output logic between them.

7.1.7.1 Multiple independent captures

The MSC is capable of performing multiple captures independent of the functional mode without the need for a power cycle or hardware reset. Each trace capture operation is atomic, and must complete before the next capture can be started. These independent capture operations must be delimited by setting MSCnEN = 0.

In single or multi-block modes, trace capture is normally completed when the trigger unit asserts the *captureDone* signal to the MSC, and the pipeline is flushed. These operations can also be terminated forcefully (though not without consequences) by the software by clearing MSCnEN to 0.

In DCI Trace Handler mode, both MSCs must be enabled and set to DCI Trace Handler mode to operate. Clearing either of the MSCnEN bits will halt the MSU’s DCI Trace Handler mode operation. The *captureDone* input from the CTS gasket is not used by the MSCs in this mode (though it is used by the DCI Trace Handler). More details on this mode of operation are available in the DCI Trace Handler chapter.

The captureDone input is also not used in internal buffer mode.

When the software clears MSCnEN to 0, the MSCnTBWP, MSCnTBRP (trace buffer write and read pointers) are reset to 0. The MSCnMWP (memory write pointer) is loaded with the contents of the MSCnBAR. If any of this status information is required to be used or saved, the software must read these CSRs before clearing MSCnEN. The process of clearing MSCnEN to 0 allows for the all-important state from the previous capture operation to be reset, thus allowing a new configuration (including a new starting base address and mode of operation) to be programmed. Setting MSCnEN to 1 will re-start the MSC capture operation in the new mode of operation.

When MSCnEN is cleared to 0, the trace data in the trace buffers (MTBs) is not cleared. It is possible to subsequently switch the MSC to internal buffer mode and read the trace buffer contents if this aids in debugging the system.

8 DCI Trace Handler

The Direct Connect Interface (DCI) is an emerging transport technology for closed-chassis debug access. The primary purpose of DCI is to allow existing functional I/O (e.g., a USB port) to be used for debug of the system and/or silicon, and to gain access to trace/debug features in the silicon.

The Intel® Trace Hub DCI Trace Handler (Intel® TH DCI TH, or simply DCI TH) is an ingredient of the Intel® Trace Hub Architecture. It orchestrates transfer of debug trace data collected from the Intel® Trace Hub to a USB port. The main purpose of the DCI TH is to provide high speed continuous streaming by smoothing out transport latencies and the bursty nature of debug/trace traffic through the use of a large buffer. The DCI TH supports two types of trace streaming destinations:

- USB Streaming: High-speed streaming over a USB port supporting a xHCI.DBC.Trace interface. The USB Host controller is responsible for packetizing and streaming over USB.
- BSSB Streaming: Low_speed streaming to USB port supporting a Boundary Scan Side Band (BSSB) transport protocol. The DCI Bridge is responsible for packetizing streaming over DCI.BSSB

Streaming mode is configurable in the DCI TH. Current usage modes suggest configuration of the mode by the debug host software stack.

The DCI TH is responsible for delivering data from the Global Trace Hub (GTH) to a USB Host controller or DCI Bridge.

As with the other Intel® Trace Hub components, the DCI Trace Handler can function independently of the operating system, as it is completely configurable and controllable via JTAG.

8.1.1 Functional requirements

- **Supports the following Streaming modes:**
  - "High-speed" streaming over USB through xHCI.DBC.Trace end point
  - BSSB mode streaming through the DCI Bridge.
- **Supports switching between USB and BSSB modes (Host assisted)**
  - One USB Port model: Requires Unplugging and plugging USB cable/ BSSB adaptor
  - Two USB Port model: One USB port streaming from XHCI.DBC.Trace and the other streaming in BSSB model.
    - Only One destination active at any time
    - Host assisted Switching between destinations.
- **Can back-pressure GTH for lossless trace streaming.**

- **Support data transfers through the primary and sideband fabrics**
  - All trace streaming related communication to/from DCI TH in DBC mode is on the primary fabric.
  - All trace streaming related communication to/from DCI TH in BSSB mode is on the sideband fabric.

8.1.2 Limitations:

- **No simultaneous streaming to more than one destination:**
  - Only one device (xHCI.DbC or BSSB) is supported for streaming at any time.
  - Store to memory is not supported while DCI Trace Handler is active.
- **No error detection/correction at storage buffer.**
- **Not a reliable data transmittal protocol**
  - No support for reliable protocol or retries of data transfers

8.2 Trace Streaming Modes

DCI TH can support two streaming modes: “High speed” USB mode streaming, and “low speed” BSSB mode streaming.

8.2.1 “High Speed” USB mode

In “high speed” USB mode, trace data is streamed over the primary fabric to a USB port. A USB host controller (xHCI) provides a debug trace interface (xHCI.DBC.Trace) to facilitate the data transfer to it. The xHCI.DBC.Trace interface provides a DBC.Trace.IN register (in MMIO space) to setup a DMA transfer. A write to this register with data address and size will trigger a DMA in the form of one or more memory reads to the data address. When the data address points to the DCI TH memory buffer, it results in debug/trace data streaming.

8.2.2 “Low speed” BSSB mode

In “low speed” Boundary Scan Side Band (BSSB) mode, the DCI TH writes the trace data to a buffer in the DCI Bridge. The DCI bridge packetizes this data into DCI packets and streams them over a USB port configured in BSSB mode. DCI packets are of constant 64B in size and allows up to 60B data payload. Communication between the DCI Trace Handler and the DCI Bridge is accomplished over the sideband fabric interface. Since the DCI TH handles data only in integral multiples of 8B (owing to internal microarchitecture details), and the sideband fabric is limited to 8B of payload per transaction, the DCI Trace Handler transfers streaming data to the DCI Bridge in 56B blocks over 7 sideband transactions.

8.3 Switching trace Streaming Modes

The DCI Trace Handler does not currently support dynamic switching of streaming modes. One mode must be completely finished before the DCI TH can be reconfigured for another streaming mode, and then started up.

8.4 End of Trace handling

In an end-of-the-trace scenario, the DCI Trace Handler must be flushed to ensure all data is transmitted to the destination. There are two methods to flush the DCI Trace Handler: automatic flush, or manual flush.

The automatic flush case utilizes the Intel® TH Trigger Unit to assert the captureDone output at the end of trace capture. When the DCI TH sees this assertion, it will automatically enter its “flush mode”, sending all data to the destination. The FLUSHDONE bit can be polled to determine when all flushing is done (despite the fact that the MSCs are involved in this trace destination, the interrupt is not asserted in DCI mode).

The manual flush case must be used when the Host software has not been configured the Intel® TH for automatic flushing at the end of trace capture. Host/debug software will need a way to detect the end of the trace. This can be determined from Intel® TH trigger state, perhaps the rate or flow of data, a specific message, or some other condition (exact method is outside the scope of this specification). Once the software detects this condition, it can force a manual flush of the last partial packet from the trace buffer by using the STREAMCFG1.FLUSH bit.

In both cases, if there is no data in the MTBs to send to the DbC.Trace or DCI Bridge, the DCI TH will immediately cease operation, sending no further writes to either endpoint, and will set the FLUSHDONE bit.

8.5 Error Handling

8.5.1 DbC.Trace Device Unresponsive

When the DbC.Trace device becomes un-responsive (from the DCI Trace Handler’s viewpoint – not requesting data or returning status) it is detected at the DCI TH through a time-out counter. After the counter expires, the DbC is deemed un-responsive, and the DCI TH starts a fresh cycle of setting up DMA transfers to the DbC.Trace endpoint.

After a number of such time outs occur (MAXTIMEOUTS), the DCI Trace Handler asserts its *portReset* signal to the GTH. This signal informs GTH that the trace destination is not able to accept data, allows GTH to use its PnDROP policy to drop the trace data if so configured. It is recommended to set the PnDROP policy bit, or to ensure the STH BDC is set to a low value so that data is dropped when the DCI TH is unable to make forward progress.

8.5.2 DCI Bridge Unresponsive

When the DCI Bridge becomes unresponsive (from the DCI Trace Handler’s viewpoint – not requesting data or returning status), it is detected through a time-out counter. Similar to the DbC.Trace unresponsive scenario, a time out counter triggers a “reset” and allows the DCI TH to repeat its attempt to send the data. After multiple timeouts (MAXTIMEOUTS), a port reset is asserted to GTH.

8.5.3 Unexpected reads from DbC

Unexpected reads from DbC (when the DCI Trace Handler doesn’t have any outstanding DMA) are completed with dummy data.

8.5.4 Unexpected credit returns from DbC

Credit returns from DBC when DCI Trace Handler has full credits are ignored. Credit return when DBC returns a credit before completing all the outstanding reads for a given DMA is processed normally.

8.5.5 Unexpected credit returns from DCI Bridge

Credit returns from the DCI Bridge when DCI Trace Handler has full credits are ignored. When the DCI Bridge returns credits before a full 56B transfer, it is processed normally and DCI Trace Handler starts counting 8B transfers towards a new transaction.

8.5.6 Reads to a non-sequential or unexpected address from DbC

The DCI Trace Handler assumes the reads from the DbC.Trace to come for addresses in sequential order for the DMAs advertised. As such, the DCI TH does not inspect or use the address of the read request. Rather, it supplies the data from the trace buffer in a sequential manner for each successive read. Any non-linearity in addresses of the read request is ignored.

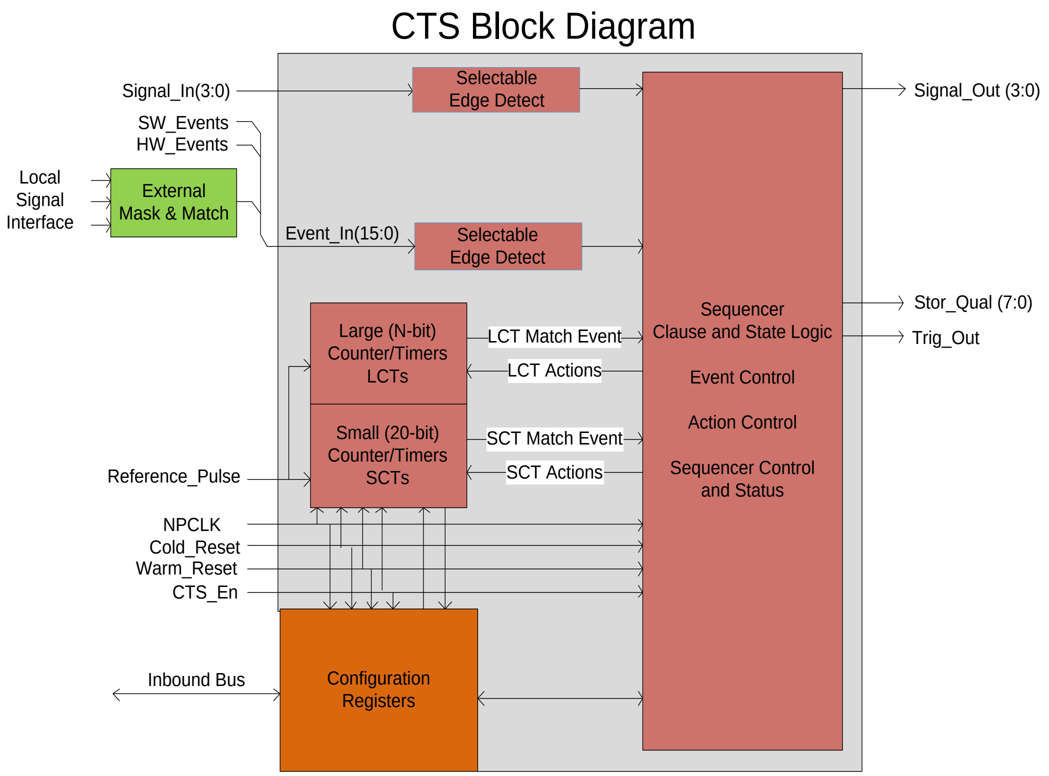
9 Common Trigger Sequencer

The Common Trigger Sequencer (CTS) provides the core triggering capability for the Intel® Trace Hub. The CTS provides the ability to detect one or more sequences of events, and to take one or more actions for each detected series of events. For each sequence of events, one is chosen to be called the “Trigger” – typically the state or condition enabling the start of data storage.

9.1 Functional Description

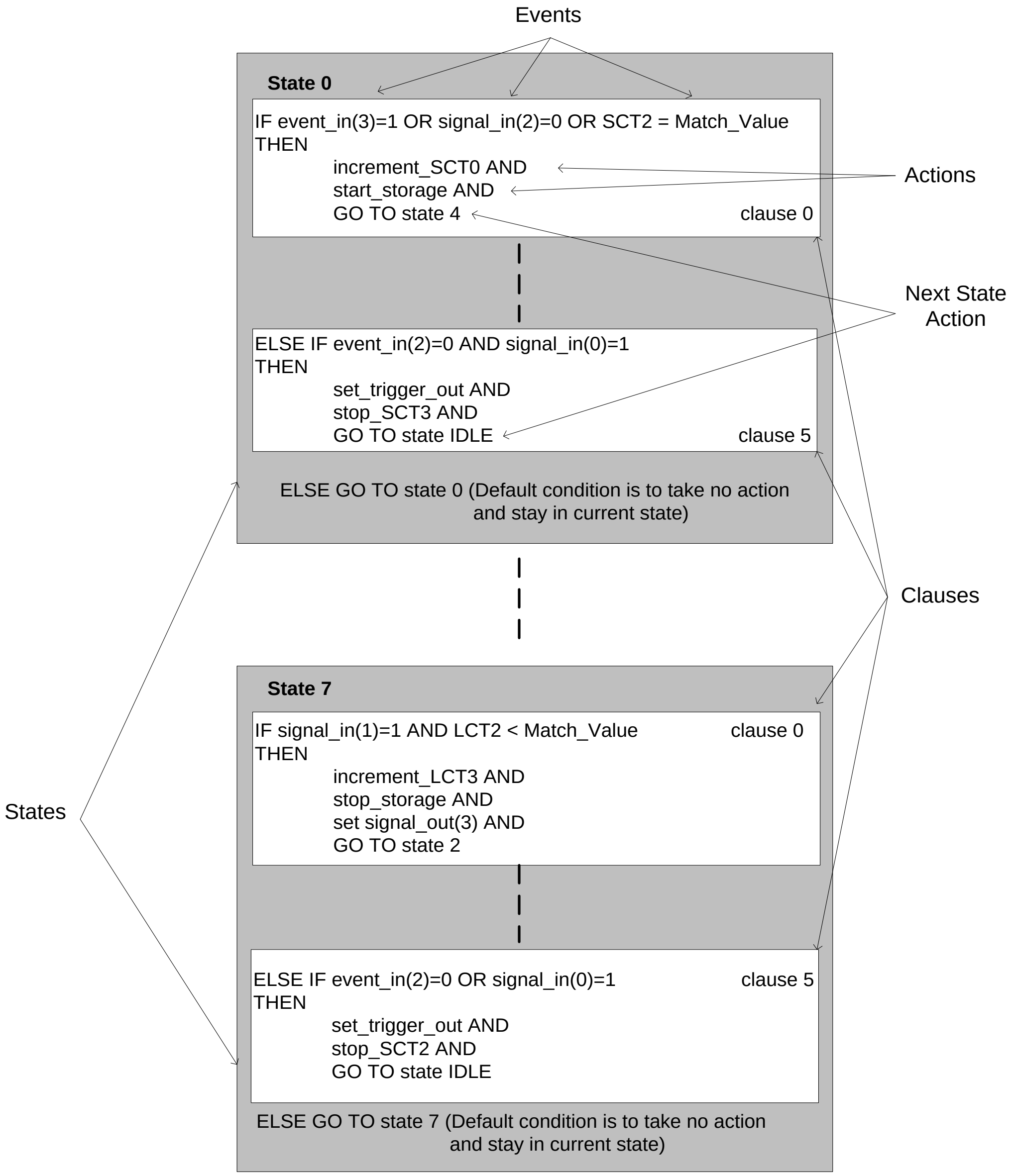
Figure 35 shows a CTS block diagram. The major blocks include the External Mask and Match logic (which is not part of the CTS specification), the Trigger Sequencer State Machine, the Global Counters, and the Timers and the Configuration Registers. Each of these sub-blocks is described in detail in later sections.

Figure 35 : CTS Block Diagram



The CTS utilizes states and clauses to process events and take actions. Figure 36 describes the terminology of the relationship between clauses, states, event inputs, action outputs and next-state indication within the CTS. A **clause** is an IF or ELSE IF pairing of events and actions within a state. A **state** represents the present condition of the system that is being monitored (or traced), and can be the result of one or more conditions, events, or sequences of events. For example, a state can represent the starting point of a system – like initial condition(s), or it can represent that a particular sequence of events has occurred. An **event** is the signal (or combinations of signals in the case of external match/mask logic), including both the assertion and de-assertion of signals, which are used as inputs to the clause. **Actions** are stimulus that can be defined to be taken by a clause when event inputs match or “hit”.

Figure 36: Textual example of sequencer terms



```
State 0
IF event_in(3)=1 OR signal_in(2)=0 OR SCT2 = Match_Value
THEN
          increment_SCT0 AND
          start_storage AND
          GO TO state 4                                        clause 0

ELSE IF event_in(2)=0 AND signal_in(0)=1
THEN
          set_trigger_out AND
          stop_SCT3 AND
          GO TO state IDLE                                     clause 5

ELSE GO TO state 0 (Default condition is to take no action
and stay in current state)

State 7
IF signal_in(1)=1 AND LCT2 < Match_Value                       clause 0
THEN
          increment_LCT3 AND
          stop_storage AND
          set signal_out(3) AND
          GO TO state 2

ELSE IF event_in(2)=0 OR signal_in(0)=1                        clause 5
THEN
          set_trigger_out AND
          stop_SCT2 AND
          GO TO state IDLE

ELSE GO TO state 7 (Default condition is to take no action
and stay in current state)
```

In order to build a trigger specification (which is the set of states, events, clauses, and actions the sequence will follow), the user must determine:

(1) the events (inputs) that will be needed,
(2) the resources that will be used, and
(3) the sequence (if any) that must be followed.

Then the trigger specification is written by groupings (states) of a series of IF … THEN … ELSE statements (clauses). Each of these concepts is described throughout the remainder of this document.

9.1.1 CTS Global Counter/Timer Resources

The CTS implements global counter/timers where all clauses of all states have access to and shared control of all counter/timers. The number and size of the counter resources are selectable via parameters. There are two types of counter/timers available. The “small” counter/timer (SCT) is 20-bits wide and the “large” counter/timer (LCT) is 45-bits wide. The maximum number of SCTs is 4 and the maximum number of LCTs is 4; the actual number is a per-project parameter (please consult your product-specific documentation for details). The SCTs and LCTs are implemented as count up (incremental) counter/timers. They are reset to 0, and compared against a programmable match value. This comparison from all counter/timers is sent as an event signal to all clauses. The value programmed for comparison is unique to each counter/timer, but shared by all clauses. Counting of events or measuring of time using the SCTs and LCTs can span across state transitions.

Operating as either a counter or a timer, the SCTs and LCTs will match against a respective *match_value* register every cycle. The result of this match translates into an event input available to all clauses of all states in that current cycle. This allows for full rate operation and sampling of the counter/timers.

When operating in timer mode, the *CTn_start_inc action will start the timer. It will operate as an incrementing timer and compare its match_value register every cycle. If a match event occurs, the SCT or LCT will stop incrementing and set the match event output to the clause logic, and remain set. It will then wait for a *CTn_clear action or hard reset to reset back to zero. While counting in timer mode, the LCTs will stop and suspend operation if the *CT_stop action is signaled. As a result, the contents of the counter will remain unchanged, and the counter will then wait for either a *CTn_start_inc action signal to continue from the current value, or a *CTn_clear action signal to clear the counter to zero. The SCTs don’t have the ability to stop and a stop action is not defined for the SCTs. In any single cycle, the timer can be instructed to clear and start indicated by the *CTn_clear and *CTn_start_inc action signals *true* simultaneously. In such a case, the timer will reset to the value *1h*. This effectively allows the counter to start counting in the same cycle that it’s cleared.

When operating in counter mode, the *CTn_start_inc action will be used to tell the counter when to increment. It will operate as an incrementing counter and compare its match_value register every cycle. If a match event occurs, the SCT or LCT will set the match event output to the clause logic. Since the user may desire to count events beyond the match event, the SCT or LCT will continue to increment if the *CTn_start_inc bit is set after the match event, instructing it to increment, allowing the user to count past the match event. Keep in mind that the match event is sticky, so it will remain set if the CT increments beyond the match value. This match event signal is cleared when the CT is cleared or reset, yielding a simpler, more flexible design. The *CTn_clear action signal will cause the counter to reset to zero. The *CTn_stop action input is valid for timer mode and undefined in counter mode.

Table 8-2 shows the counter and timer sizes translated to ranges for different example frequencies.

Table 9-2: Counter and Timer ranges

Counter or Timer Width	Maximum Count Range	Timer range at 100 MHz	Timer range at 2 GHz NPCLK (as an example frequency)
20 – bit SCT	1 M events	10.5 mSec	523.7 uSec
45 – bit LCT	32 T events	4.1 days	4.9 hours

9.1.1.1 Peak Timer Mode for LCTs

The Peak Timer mode for LCTs gives the user the ability to measure the worst-case time from one event to another event and store the maximum (peak) count value in a readable status register for inspection by the user.

This feature is enabled with the LCT_Mode(1) configuration register bit in the LCT Control Register. The comparison logic for the LCT_0 uses the peak count status value for the LCT match comparison if the LCT_Mode(1) = 1. The clause logic can be configured to match on start and stop events, and to take the LCT *clear* action upon the second event. The LCT operates in timer mode during this operation and counts on NPCLK. If the time between events exceeds the previously-stored peak count time (i.e. a match), then the peak count status is updated with the contents of the LCT functional register.

The following example highlights the value of the requested capability to repeatedly measure time between event_0 and event_1 (or any combinations of events currently available within a clause), and save as status the maximum duration.

```
State_0:
   // start LCT_0 upon first event, then free running
        If event_0 = 1 then start LCT_0 and go to State_0
        // clear LCT_0 upon second event
                  Else if event_1 = 1 then clear_LCT_0 and go to State_0
   // updates peak timer if  LCT_0_Match = 1
        Else stay in State_0
        // default clause
```

To summarize, the peak timer value for its cycle by cycle comparison, and the peak timer value is a status register that is updated with the LCT clear action that is readable via the sideband fabric. Start or Stop actions will not affect the peak timer logic, which improves the flexibility of this feature. While in the IDLE state after a trigger action the peak timer contents will be preserved to give the user software time to read its contents. The peak timer status register will be cleared when the CTS is enabled and transitions out of the IDLE state.

9.1.1.2 Dedicated Event Counter Mode for SCTs

The Dedicated Event Counter mode for SCTs gives the user the ability to count event_in events without the need for increment control by the clause logic. Additional configuration register bits to support this include the SCTn_Mode(1) bit, and the SCTn_EventSel(n:0) bit to select which event_in(15:0) to count. SCTn_EventSel(n:0) will only implement the number of bits required to select the full range of Event_In(n:0) inputs selected via parameterization. For example, if 8 Event_in(7:0) inputs are parameterized into the design, then three SCTn_EventSel(2:0) registers are required.

If the dedicated event counter mode is selected by the user, then the SCTn_EventSel(n:0) will be used as a mux select to choose what event_in(15:0) input to count. The SCTn_start_inc action will not be used as an increment action. The SCTn_clear action will still be used to clear the counter. In this mode, the SCT will only increment if the selected Event_in input wire is (high true) in the current cycle. This increment decision is made each cycle that the CTS is active, which means the SCT dedicated event counter activity starts counting from zero when CTS is enabled and exits the IDLE state, and continues until the CTS enters the IDLE state or is disabled. Additionally, the SCTn_match_value will still be compared against every cycle to create the SCTn_match event to the clause and state logic.

The user can program the SCTn_match_value to a desired value, enable the dedicated event counter mode, and select which SCT is to count which Event_In input wire. The user can then take advantage of the SCTn_match event knowing that hardware is supporting the generation of comparing for a certain event count on the configured SCT. At any time the user can clear the SCT to zero and know that the dedicated event counter will continue counting from zero. If the user clears the dedicated event counter in the same cycle that the selected Event_In is asserted, then the dedicated event counter will count that event, and actually be cleared and incremented to 1 (the SCT is actually cleared to 1 in this case).

To clarify: to use this mode the user will have to perform the following actions:

- Program SCTn to be in the Local Counter mode
- Select which event_in(15:0) to select to count
- Program the match value for SCTn
- Program the trigger clause to “If SCTn_match then trigger”

The ELSE clause would not be needed to perform the event counting as the logic to do this is automatic, and supports multiple counters operating in parallel, potentially freeing up multiple clauses. This is known as a local counter operation (CTS is calling this the dedicated-event counter mode) and matches the capabilities of external logic analyzers and other on-die methods to count events. The CTS is not designed to be a performance counter *per se*; however, it does add capabilities to the CTS small counter/timers to aid in performance counting.

9.1.1.3 Internal Counter/Timer Events

Counter/Timers are global resources that operate in a dual mode as either a counter or a timer. The mode of operation is controlled by that counter/timer’s control register settings. In both modes of operation, the counter or timer is reset to zero and increments from zero. The current value of the counter/timer is matched against a programmed match value every cycle. Having the counter/timers operate in dual modes is done to share the increment and match logic for both modes. When configured to operate in a given mode, its resources are available to all clauses of all states, but each operates in one mode at any time. Each counter/timer has an output match event signal that is available to all clauses of all states. Once a match is signaled, it is “sticky” and thus remains true until the CT is cleared, or reset. The timer or counter will match against the counter's match_value setting programmed in its control register. Each clause can individually select any combination of SCTs or LCTs match event output as *match* or *not matched* event inputs enabled for that clause. There is an option to match when the match value is not equal to the value in counter match value register (the event signal being low indicates not a match). This option is set via the SCTn_mode or LCTn_mode bits in the clause event mode registers. The counter will only count if the start_inc action is true for that cycle, and the counter will count the number of cycles that the start_inc action is true. In timer mode, the start_inc action means to start the timer free running, and will increment every cycle until it sees a stop or clear action or a reset event. In timer mode, the timer will always function at NPCLK frequency, but it may count using a prescaler to get coarse granularity. Timers have 2 modes of operation. The timer can either be free running at NPCLK frequency, or it can be configured to count Reference_Pulse input signals being high true. For Intel Trace Hub 1.0 the Reference_Pulse input is always asserted, so there is no difference between these two modes.

9.1.2 External Input Events

There are 20 external event inputs that can be combined to create a match event per clause / per state in the sequencer. These are the Event_In(15:0) and Signal_In(3:0) input pins. These external events are combined with internal counter/timer events and available in combination to all clauses of all states.

There are control bits in the CTS Control register and the clause event mode enable control registers that control the treatment of the primary inputs and their effect on each of the clauses within each state. Each clause of each state will have a unique event mode and event enable register to control the clause's behavior. The input configuration register is shared by all clauses of all states. Both Event_In(15:0) and Signal_In(3:0) inputs have rising edge detectors that can be selected with registers in the CTS Control configuration register.

To select which of the inputs should be considered for a match event per clause, there are a set of control registers for each clause for each state of the sequencer. In each clause control register, there is an OP Type bit which indicates whether all of the enabled input matches for that clause are "ORed" together or "ANDed" together. The reset or default value of all these register bits is 0. If we are over budget on area, consider that these registers can be made to be a non-reset type, and the software could then be sure to write valid values into all when it configures the CTS.

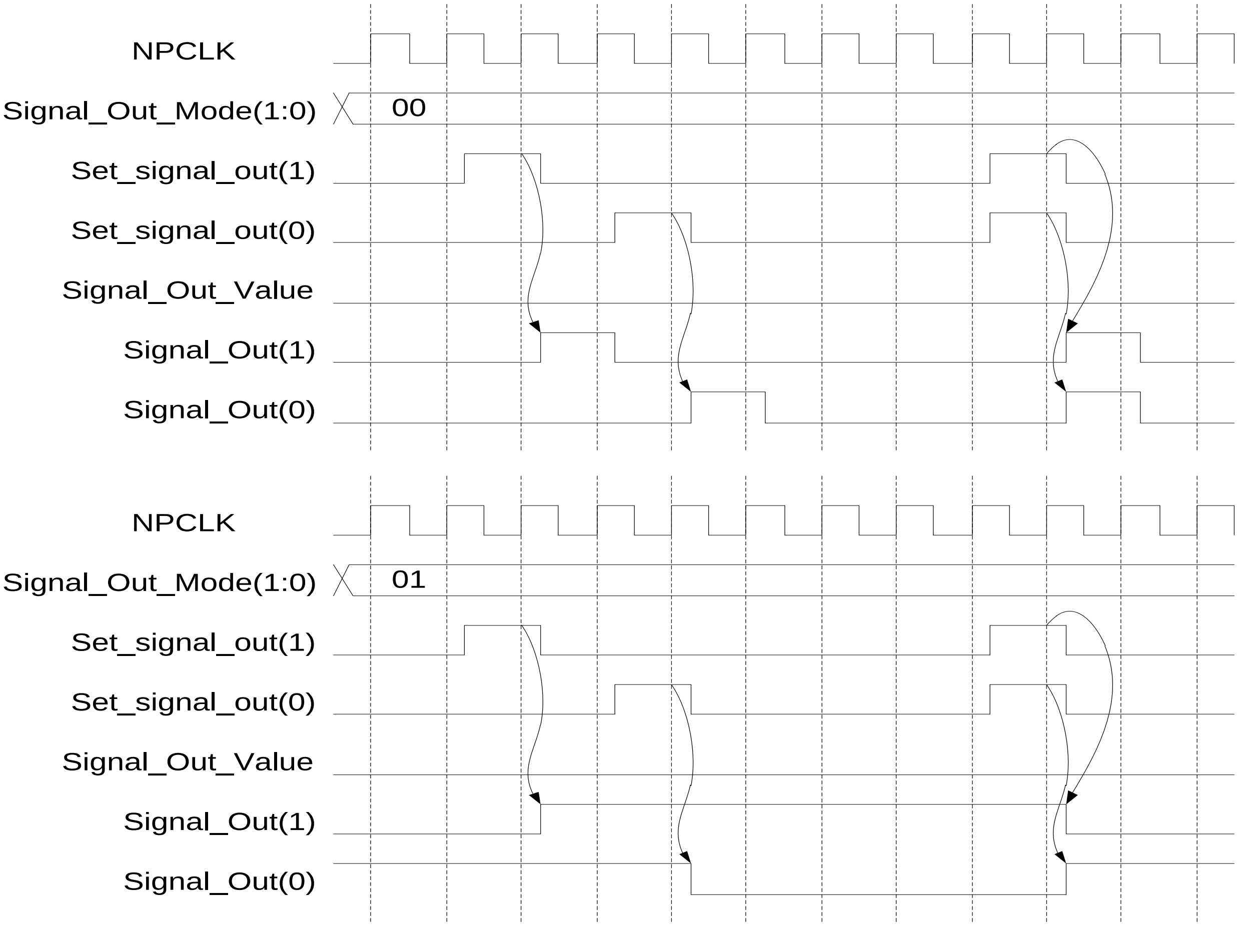
9.1.3 Sequencer Actions

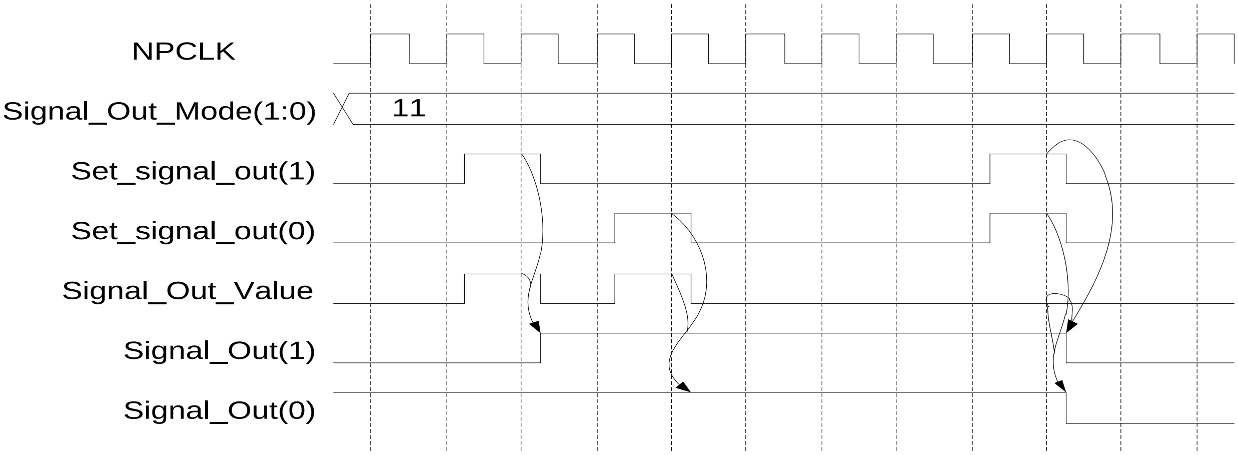
Upon a match event in any clause from any state of the sequencer, a number of output actions can be taken. There are seven external actions: Set_signal_out(3:0), Set_trig_out, Start_storage,and Stop_storage. Other internal actions are all taken upon the sequencer state machine itself, such as going to another state, starting a timer, or incrementing a counter. All actions selected in the clause output control registers will happen simultaneously upon a match event, but some selections are invalid when others are selected, such as starting and stopping a timer in the same cycle.

9.1.3.1 Signal Actions

Up to four Signal_Out(3:0) actions can be specified when a clause match event occurs. Set_signal_out(3:0) will set the Signal_Out signals that are routed as outputs of the sequencer and can be used to communicate to other circuits or blocks in the SOC where the user can select the signal_out wires to be pulsed, to toggle, or to signal a level. If the pulsed mode is selected, then the signal_out wires will be pulsed for one NPCLK on each Set_signal_out action. If the toggle mode is selected, then the signal_out wires will invert on each Set_signal_out action. If the level mode is selected, then the Signal_Out(3:0) wires can be set to the value of the Set_Signal_Value action qualifier. The signal_out wires can be used to set a marker with a timestamp in the trace, but that requires external implementation specific logic. The following diagrams show the different behaviors of the Signal_Out modes.

Figure 37: Timing Diagrams showing the Signal_Out mode behavior





9.1.3.2 Storage Control Actions

There are Stor_Qual(7:0) output signals from the CTS to the storage subsystem used to communicate storage control. The Start_storage action will set the Stor_Qual output and the Stop_storage action will clear the Stor_Qual output. These action to the Stor_Qual outputs are sticky, thus Stor_Qual is defined as level output signaling.

9.1.3.3 Trigger Action

The Set_trig_out action will pulse the Trig_Out output signal for one NPCLK cycle.

9.1.3.4 Counter Actions

There are two counter actions per counter/timer that can result from a match event within a clause. Upon a match, selected actions can cause the counter to be incremented by 1, or cleared (set to 0). It is not valid to increment and clear from the same clause – at most one may be selected.

9.1.3.5 Timer Actions

There are three timer actions per counter/timer that can result from a match event. Upon a match, selected actions can cause the timer to start counting which means it will start from ‘h0 and begin incrementing, or start from its current value if it’s previously been stopped. The timer can also be stopped with the stop event for subsequent re-start if desired (note the stop action is only available on the LCTs, and not the SCTs). The timer will set the timer_n_match bit to the mask and match logic during the cycle that it reaches the match value. The timer will halt when it reaches the match value. The timer can also be reset based off of a match event. This means the timer will be reset to zero. Another match event with start timer selected must be used to start the timer again. It is valid to start the timer and clear the timer from the same clause, so the value loaded should be ‘h1 in this case. The LCTs can also be stopped with its current value preserved waiting for another start action to resume counting. This is useful for accumulation of time.

9.1.3.6 “Go To” State Actions

Upon a match event for any clause, the output action can be configured to go to any available state, including the current state and the idle state. This allows for the sequencer to be used to look for a sequence of events (eg. ‘A’ followed by ‘B’, followed by ‘C’) among other trigger scenarios. The CTS is specified to be a one-hot state sequencer, that is, there is always one and only one active state. A requirement of the CTS is that each clause in each state must specify the next active state by having the Set_state(3:0) configured in the action control register. Any value programmed in this register must define a valid entry as there is no hardware support to avoid illegal state transitions. In other words, if 4 states are parameterized & included in a given local CTS instantiation, then the Set_state(3:0) value must be within the range of 0 – 3 or IDLE. Any illegal state transitions programmed and taken may cause a hang of the CTS. The hardware for the clauses are built as follows:

```
If (condition) then
    Take Actions, and go to state W
Else if (condition) then
    Take Actions and go to state X
Else if (condition) then
    Take Actions and go to state Y
Else if (condition) then
    Take Actions and go to state Z
Else
    Stay in current state and take no actions
```

What is implied and will be supported by hardware is the final "***Else stay in current state and take no actions***" choice. The user need not define this, and the software can assume this action is hard wired.

9.1.4 Sequencer Control and Status

9.1.4.1 Sequencer Control

The Sequencer_Enable configuration bit has a rising edge detector built on it to form a "start" bit internally. When this start bit is true, the CTS logic will use the Force_State configuration register setting to inform it if it's a normal start condition where the state machine will proceed to state 0, or if it's a power state re-start condition where all RW_V shadow status registers will be "jammed" back into the functional registers. These registers jammed are the current state, Stor_Qual state, signal_out state, all counter timer values, and the currently running indication of any counter/timer that is programmed into timer mode. Clearing the Sequencer_Enable bit will reset all the counter and timer resources and the state machine will go to idle. All of the clocks for this block of logic can be gated when the enable is cleared. The enable should be cleared on a cold reset of the part, but if possible should survive a warm reset so that the triggering logic can be used during reset cycling of the part. The enable bit must be set to turn on the clocks to this logic.

9.1.4.2 Sequencer Status

This section describes the status registers available as status in the CTS. All status register can be written by the software to be used to "jam" (force) the sequencer state by using the Force_State register during a power state restart. During normal operation, an internal connection from the CTS to the register block will be generated that is the Update_Status configuration bit tied to the update input to the register block. Update is used as a status update enable for all shadow status registers. The software can clear the Update_Status register in advance of reading all the numerous status registers. This allows for a “snapshot” of the constantly changing state of the CTS to be held for observation while the software reads all the status registers.

9.1.4.2.1 Sequencer State

The four Sequencer_State(3:0) bits are a binary encoding of the current state the sequencer is in and can be read by software to give a real time update of the current state of the sequencer. There is a maximum of 8 parameterized states available, plus IDLE. The Sequencer_State(3:0) bits can be read real time reliably because the register access interface is synchronous to NPCLK as is the sequencer itself.

0000 = State 0

0001 = State 1

0010 = State 2

0011 = State 3

0100 = State 4

0101 = State 5

0110 = State 6

0111 = State 7

1000 = IDLE

9.1.4.2.2 Trigger Status

The trigger status bit indicates that a trigger action on Trig_Out has occurred.

9.1.4.2.3 Stor_Qual Status

The Stor_Qual_Initial_Value status register will be used by the CTS to set the state of the Stor_Qual output signal upon a sequencer start event as signaled by the rising edge detect logic on the Sequencer_start configuration bit. Once the CTS has started operation as indicated by the sequencer_running signal, the the Stor_Qual_Initial_Value register will act as a shadow status register of the Stor_Qual output bit. The Stor_Qual_Initial_Value register is similar to other status registers, but differs in one way, that is, the Stor_Qual output is always forced with the Stor_Qual_Initial_Value register upon a sequencer start up regardless of the state of the Force_State register contents.

9.1.4.2.4 Counter/Timer Status

Each of the Large and Small Counter/Timers has a current status register that reflects the real time contents of the counter/timer during operation. An additional status bit is available if in timer mode called *CT*_Running is a status indication that the timer is currently running due to a previous start action. The software can reliably read the counter or timer’s current status registers because the register-read interface is also synchronous to NPCLK within this block of logic.

9.1.4.3 Conditions that affect use cases

The following table is a compilation of internal events and a description of how they affect the internal logic relating to usability. The intent is to enumerate the different corner cases and have a table describing what internal actions are taken.

Table 9-1: Corner case actions

Condition or Event	Cause of Event	Default Action Taken	Description

Sequencer Start	Sequencer_Enable = 1 &
	Force_State = 0
		State transitions to state 0
		All counter/timers reset to 0h
		Trig_out = 0
		Signal_Out (3:0) = 0h
		Stor_Qual(7:0) = Stor_Qual_Initial_Value(7:0)
			This is a normal startup condition.

Sequencer Start with forced state	Sequencer_Enable = 1 &
	Force_State = 1
		State transitions to contents in Sequencer_State(3:0)
		All counter/timers are loaded with contents of the status shadow registers, including the running status
		Trig_Out = 0
		Signal_out(3:0) = 0h
		Stor_Qual(7:0) = Stor_Qual_Initial_Value(7:0)
			Restoring the state after a power event. This is identified using the Force_State configuration register.

Sequencer Stops	Clause event match causing next state transition defined as IDLE	All counter/timers stop at current value Trig_out = 0 Signal_Out(3:0) = 0h Stor_Qual(7:0) holds the current value, unless there is a stop_storage or start_storage action in this cycle, then it takes that action and holds last value when IDLE.	This is normally due to a trigger condition, and is part of normal operation
Cold_Reset	Cold_Reset Input pin assertion	All resettable registers in the design are reset to their default values. This includes the reset on the flops for all inputs and outputs, they must all reset to 0. The state will reset to IDLE = 1000.	This is a normal cold reset that is normally due to a power cycle event.
Software Reset	Software writes Sequencer_Enable = 0 configuration register	Will not affect the configuration registers except for the status registers. State is reset to IDLE, Counter/Timers will halt and reset to 0, Outputs are all reset to 0.	

9.1.5 Clause and State Operation

All states are identical except that the sequencer will always progress to state 0 from IDLE when the Sequencer_Enable = 1 and Force_State = 0. This begins normal operation of the trigger sequencer operation. The independent states will share the primary event inputs after the synchronization and edge detect logic. The states will also share the internal events resulting from the counter and timer match or not matching. Each clause of each state has an Input Control Register associated with it that determines which events in combination can cause a TRUE or Logical “1” on the output of its event qualifier. This translates to a TRUE on its “if” or the “else if” clause. Multiple events could be AND’d or OR’d together, but no other combination of AND/OR is possible. This AND/OR function is selected by the OP_type configuration registers.

The CTS supports AND/OR of all enabled events on a per clause basis. Additionally the inverse of the event can be chosen using the clause event mode registers. Other AND/OR event combinations can be supported with the ELSE/IF implied within the clause logic. For example:

```
IF ((A AND B) OR (C AND D)) THEN perform action X
```

is not supported by the CTS, but can be performed with the CTS by programming as follows:

```
IF (A AND B) THEN perform action X                    // clause 0
ELSE IF (C AND D) THEN perform action X               // clause 1
ELSE do nothing
    // default clause
```

It is important to note that a priority encoder is built to form the “if / else if / else if ...” clause sequence for the state. The clause 0 event qualifier result will be the first “if” clause. If clause 0’s conditional match logic (hit) is true, then it will gate off the actions from all later clauses via its inverted value being an input to the later states AND. It follows that clause 1’s conditional “else if” operator is only enabled if clause 0’s didn’t “hit”, and if true will disable clause 2 and clause 3, etc. The S*_active input indicates that this is the current active state and is used to enable action outputs from this state if the sequencer has this state as it’s currently active state.

At most one clause can signal a positive high true result to the action qualifier block. In this block, the single input will cause possibly multiple output events to be set. This is all controlled by each clause’s Action Control Register settings.

At the top level, each state’s action outputs will be OR’d together to create the final action bits that go to either the counter/timers, the sequencer (indication of next state) or the primary outputs.

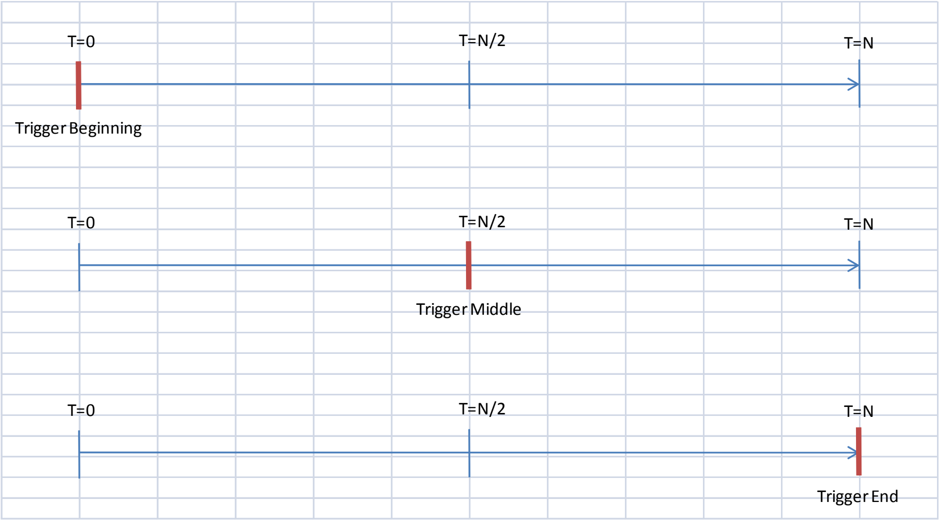
9.1.6 Global Triggering Considerations

9.1.6.1 Trigger Location within Trace Capture

In most trace capture situations, the user wants to be able to place a trigger point at some programmable place within the trace. There are three baseline trigger locations that are useful: trigger at the beginning of trace, trigger in the middle, and trigger at the end. CTS supports each of these three, and any variation in between.

Trigger at beginning of trace is when a trigger event occurs and storage is starts and fills up the entire selected storage space. Therefore the trace shows a trigger location at time = 0 and the rest of the trace follows. Trigger at the middle of the trace is when a trigger event occurs and storage continues for ½ the storage size, which allows for ½ the data to be previous to the event and ½ the data after the trigger event. Trigger at the end of the trace is when a trigger event occurs and storage halts immediately, such that all data stored is data previous to the trigger event.

Figure 38 : Trigger locations



9.1.6.2 Circular Trace Buffer

Trace capture systems normally support the concept of a circular trace buffer, which is a buffer of a given size that is allocated and used in a circular manner until a trigger occurs. The circular trace buffer allows capture and storage of information prior to the trigger (trigger at end of trace), and then optionally continue capturing after the trigger until the post-trigger storage counter has counted down.

9.1.6.3 Trigger or Marker Indications in Trace

The user needs to know the trigger location’s position in the trace. The Trig_Out signal action output signal is used to communicate this trigger from the CTS to the storage logic. Trace Sources can then insert a TRIG packet into their trace stream to indicate the position of the trigger relative to their trace stream.

9.1.6.4 Capture Abort

A mechanism to abort the capture after triggering is required. For cases where post-trigger storage is selected, but due to the nature of the traffic, source selection or filtering, storage is not proceeding or proceeding slowly. The tool would set the bit (or bits, or sequence of actions) necessary to stop the trace collection in response to user GUI selection. This feature is typically used when a post-trigger storage counter value is set but no progress is being made toward filling the storage memory (user knows this as tool indicates fill progress by querying and displaying the value of the post-storage counter). This would be a function of the storage subsystem rather than the CTS.

9.1.6.5 Repetitive Triggering

A typical LA feature, repetitive triggering provides storage at each occurrence of a trigger. After a trigger occurs, the storage completes, and the trigger re-enables. This feature can be simulated using storage qualification by enabling storage and starting a timer based on the event of interest, stopping storage when the timer expires, and then going back to searching for the event of interest. This too would be a function of the storage subsystem. The CTS can support this by configuring the state sequence and matching to provide multiple Trig_Out indications. This is because the Trig_Out action is independent of the next_state action due to any clause's match. This means that a clause could set Trig_Out, then go back to state 0 for example (not forced to go to IDLE) to continue sequencing. Subsequent Trig_Out conditions could then be matched upon.

9.1.6.6 Power State Survival & Power concerns

There are multiple low power states in a system and they are all implementation-specific. Low power states that reduce the voltage or frequency should be survivable by the CTS. The CTS does not provide the hardware hooks to self-correct from loss of power. Re-starting the CTS after a loss of power requires software or firmware support. Hooks have been implemented in the CTS to allow capturing of the state, forcing back in this state, and re-starting after a power down event.

The procedure would be that the software/firmware will capture the state of the CTS's status registers before power is lost and save the state somewhere in non-volatile memory. A snapshot of the state of the CTS can be captured by setting the Update_Status register then clearing it. In the cycle that Update_Status is cleared, the state of the CTS will be held in the shadow status registers as they will quit updating. The software/firmware can then read all the CTS status registers without them changing during the process.

Upon resumption of power, the software/firmware writes the previously-saved state back into CTS. The CTS status registers are unique in that they are shadow copies of the functional registers that act as both readable status registers during normal operation, and as writeable registers that will force the state of the functional registers opportunistically. The software can set the Force_State bit along with the Sequencer_Enable register to force the state of the CTS and bypass the normal startup to State 0. Forcing the state of the CTS will also force the state of the counters, timers, and output signals to the value of their status registers. While going through this restore sequence, the Update_Status register bit must be cleared to ensure the integrity of the status CSRs that will be forced back into the CTS.

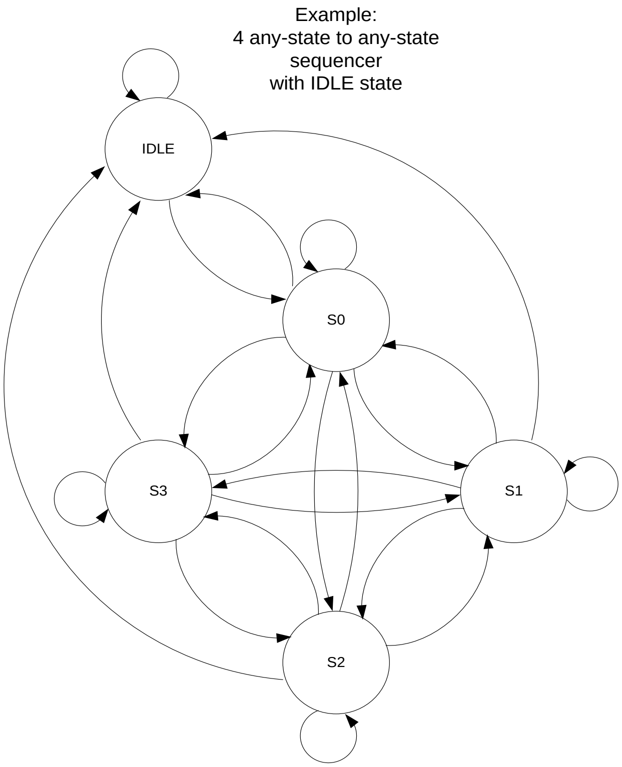
This concept of re-starting after a power state is tricky, and there are many corner cases. One caution is that any CTS activity that occurs after the snapshot of the CTS state is taken will not be visible because the status registers quit updating when the Update_Status bit is cleared. In other words, the state of the CTS will be restored to this snapshot taken of the state.

9.2 CTS State Machine

The promoted re-use and flexibility of the CTS is supported through modularization and parameterization. The number of states and clauses of the CTS State Machine are parameters that can be set to meet the target requirements. The maximum number of functional states for the CTS is 8 (plus an idle state) and the maximum number of clauses per state is 6 (plus the default clause). The logic for each clause of each state is identical. They are configured in a way to create a prioritized if/then/else if structure per state, as well as a state sequencer at the top level.

Figure 39 shows an example four-state trigger sequencer machine. It is a Finite State Machine programmed through a set of control registers that is used to create complex triggers based on a number of input variable match conditions. Each clause of each state of the sequencer has all input events and all the internal counter/timer events available as input conditions to match against. Upon a match condition, any number of available actions can be taken and any next state can be defined. All states of the sequencer share the global counter/timers. These counter/timers can be configured to either count events or measure time. Configured as a counter, the counter/timer can be used to count match events before either moving on to the next state of the sequencer, or signaling one of the selectable output actions. Configured as a timer, the counter/timer can be started, stopped or re-started based on any match event and used to measure or accumulate time values (note that stopping and subsequent re-starting of a timer is only available with LCTs). Because of the possibility for trigger scenarios to look for a number of different conditions per state, there will be up to six clauses per sequencer state – an “if event W, else if event X, else if event Y, else if event Z … etc.”. Because the user will most likely not want to use the same exit condition for each of the clause conditions, each clause has the ability to choose any or all of the actions and next state arc on the clause match (shown as W, X, Y, and Z above). Each clause of each state of the sequencer can go to any other state of the sequencer, including back to the same state upon a match event.

Figure 39: Sequencer Block Diagram

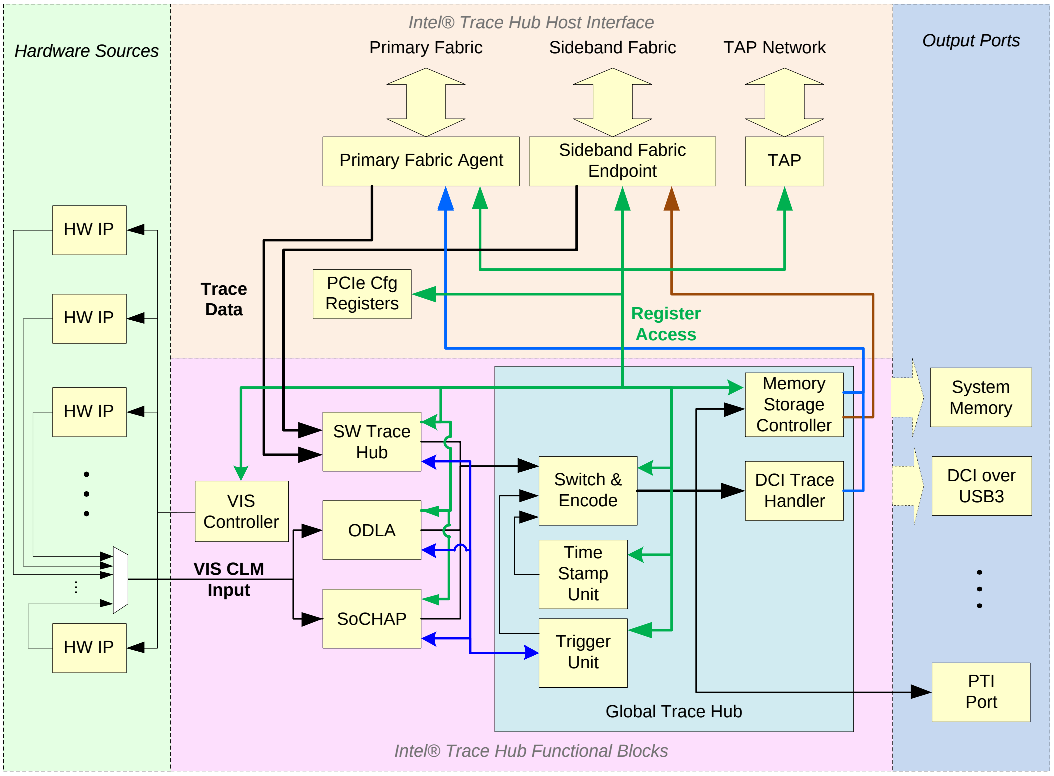


10 Parallel Trace Interface Port

The Parallel Trace Interface (PTI) port is an ingredient of the Intel® Trace Hub Architecture shown in Figure 40. The Intel Trace Hub architecture is designed to support multiple trace destinations for the collected trace data. Some destinations are on-chip, like system memory, while others are off-chip, like the PTI port which is the topic of this chapter.

The MIPI-PTI is a standard interface defined by the MIPI Alliance[10]; it is composed of a parallel output port used to send trace data off-chip for capture by external hardware tools.

Figure 40: Intel Trace Hub architecture high-level diagram



10.1 PTI Port Software Support

The Intel Trace Hub's PTI port configuration registers are visible to software through Intel Trace Hub's PCI device. They are accessible through all interfaces supported by the Intel Trace Hub, including the primary fabric interface, sideband fabric interface, and TAP network.

[10] *The Parallel Trace Interface standard is defined and maintained by the Mobile Industry Processor Interface (MIPI) alliance working group.*

The PTI port can be completely enabled or disabled via software running on target, or through software running a Debug and Test System (DTS) on a JTAG-based interface.

10.2 PTI Port Calibration/Training Sequence

Support for MIPI-PTI specified calibration/training patterns generated by an internal pattern generation block to be used for calibrating external capture devices.

10.3 Functional Description

10.3.1 Intel Trace Hub Trace Transfer through the PTI Port

The Global Trace Hub (GTH) is the main trace collector for the Intel Trace Hub architecture. It is responsible for collecting data from the various sources, encoding the data into industry standard STPv2.1 (System Trace Protocol version 2.1)[11] and optionally time stamping it with the Time Stamp Correlation Unit. If not already time stamped, it can enable correlation of the occurrence of software and hardware events.

The GTH supports multiple trace-destinations for transferring the collected data to either an on-chip or off-chip location. The trace destination addressed in this document is the PTI port. To this end, the GTH will transmit the *trace data*[12] it has collected to the PTI port controller, which will format it into PTI packets as explained below. This data is then forwarded to the General Purpose Input/Output (GPIO) pins for transmission. Moreover, the PTI port controller/formatter can also down-scale the Intel Trace Hub clock frequency if required by the specific GPIO technology.

On the Debug and Test System (DTS) side, the debug host PC or workstation is responsible for processing trace data received by the DTS to obtain an actual reconstruction of the system’s behavior. It will include all the necessary APIs and the software stack to perform this task.

In addition, like other Intel Trace Hub off-chip output ports, the PTI port supports an additional mode known as the *Bypass Mode* where the input data (most likely clock synchronous VIS data from the *CLM*[13]), is forwarded directly to the pins without any modification or processing. This mode will also be explained later in this document.

10.3.2 MIPI-PTI Protocol and Terminology

The MIPI-PTI protocol is an industry standard interface defined to export relatively high-bandwidth debug and trace data from the target system to the DTS through the DTC.

A PTI is a unidirectional, Double Data Rate (DDR), interface with a set of *PTI_BPBWIDTH/2* single-ended data lanes. The strobe/clock output in the MIPI-PTI specification is, in the

[11] *The MIPI STPv2 is the industry standard tracing protocol developed by the MIPI Alliance.*

[12] *In the context of the Intel Trace Hub architecture, trace data and debug data are used interchangeably to refer to the data collected from various hardware and software sources that enable an accurate reconstruction of system behavior for debug and performance optimization purposes.*

[13] *CLM stands for Chip Level Mux where all the VIS signals are collected from their various sources throughout the chip.*

remainder of this document, referred to as a dedicated output called *Trace Clock*. In addition to the above outputs, Intel Trace Hub PTI defines an additional output known as the 'Trace Valid' signal (*trace_vld*), which is asserted as long as there is valid data on the output pins. In VIS-bypass mode, the two dedicated outputs are used to drive the two VIS clocks while the data pins drive the VIS data.

It is also important to mention that in most of the use cases the DTS is the master controller for the PTI port and is responsible for providing control and coordination with the PTI port logic through one of the standard interfaces (e.g. JTAG interface, sideband fabric, or primary fabric). There do exist, however, some use cases where the SUT can configure itself for trace collection – in this case the DTS will have to be configured *a priori* to accept all trace data sent to it. Details of how the PTI port controller is configured, however, are beyond the scope of this document.

Finally, the PTI specification defines a PTI *packet* as the data transferred on the data lanes, *trace_data*, on a single toggle of the trace clock or trace strobe (note that the *trace_vld* signal is not required for a valid PTI packet).

10.3.2.1 Clock Divider Logic

Since the Intel Trace Hub clock is usually one of the fastest clocks on an SoC, a slower clock may be required for data transfer on the GPIO pins. To this end, a clock divider is embedded in the PTI port controller logic that will generate such a clock. The PTI port controller clock divider supports ratios of 1, 2, 4, and 8.

10.3.2.2 PTI Mode Mux

Intel Trace Hub's PTI port supports 4 modes where each uses a subset of the GPIO pins to drive the trace data. A summary of these 4 modes and the GTH trace data to pin mapping is presented in Table 10-1.

If the GTH interface width is not an integer multiple of the number of data lanes enabled for trace data transfer, additional NULL packets will be added to the trace data under certain conditions. One such occurrence would be if the GTH is quiescent for a long enough period of time such that the PTI port internal FIFO does not have enough data to fill all of the enabled data lanes.

Table 10-1: PTI port modes and signal mapping

Mode/Pins	PTI 4-bit Mode	PTI 8-bit Mode	PTI 16-bit Mode
trace_data[3:0]	pti_data[3:0] pti_data[7:4] pti_data[11:8] pti_data[15:12] pti_data[19:16] pti_data[23:20] pti_data[27:24] pti_data[31:28]	pti_data[3:0] pti_data[7:4] pti_data[19:16] pti_data[23:20]	pti_data[3:0] pti_data[19:16]
trace_data[7:4]	N/A	pti_data[11:8] pti_data[15:12] pti_data[27:24] pti_data[31:28]	pti_data[7:4] pti_data[23:20]
trace_data[11:8]	N/A	N/A	pti_data[11:8] pti_data[27:24]
trace_data[15:12]	N/A	N/A	pti_data[15:12] pti_data[31:28]

10.3.2.3 PTI Transmission Controller

The PTI transmission controller is responsible for controlling the data flow and the data multiplexer based on the programmed configuration. It is responsible for generating the index signals driving the mux and generating the *trace_vld* signal which is also used for generating the PTI trace clock (or strobe).

10.3.2.4 PTI Pattern Generator

The MIPI-PTI standard requires a minimum of six training/calibration patterns that have to be supported by a PTI port as well as DTS. There is a dedicated block in the PTI port controller that is responsible for generating those five patterns based on the input configuration. A list of those patterns, their configuration bits, and use cases directly copied from MIPI-PTI specification are given in Table 10-2. Also note that the pattern will only be driven on the pins corresponding to the current configuration mode.

Table 10-2: MIPI-PTI training/calibration patterns

Pattern Number	Patten Generation Config Mode	Packet Sequence	Use Case

Pattern Number	Patten Generation Config Mode	Packet Sequence	Use Case
0	3'b001	• trace_data[] all ones • trace_data[] all ones • trace_data[] all zeros • trace_data[] all ones • trace_data[] all ones • trace_data[] all zeros • …	• Stresses power supply/ground of the board/chip by switching from all highs to all lows on the same clock (looking for ground bounces). • Stresses pin to pin skew by driving all signals high for at least 2 bit periods (by ensuring the signals attain the maximum "high logic" voltages before a transition.
1	3'b010	• trace_data[] all zeros • trace_data[] all ones • Invert trace_data[0] • Invert trace_data[] • Invert trace_data[] • Invert trace_data[1] and trace_data[0] • Invert trace_data[] • Invert trace_data[] • Invert trace_data[2] and trace_data[1] • Invert trace_data[] • Invert trace_data[] • … • Invert trace_data[N] and trace_data[N-1] • Invert trace_data[] • Invert trace_data[] • …	• Stresses chip electrical characteristics by driving a pattern that is dominated by high signals. • Verifies signal mapping by walking an inverted bit across the interface.

Pattern Number	Patten Generation Config Mode	Packet Sequence	Use Case
2	3’b011	• trace_data[] all ones • trace_data[] all zeros • Invert trace_data[0] • Invert trace_data[] • Invert trace_data[] • Invert trace_data[1] and trace_data[0] • Invert trace_data[] • Invert trace_data[] • Invert trace_data[2] and trace_data[1] • Invert trace_data[] • Invert trace_data[] • … • Invert trace_data[N] and trace_data[N-1] • Invert trace_data[] • Invert trace_data[] • …	• Stresses chip electrical characteristics by driving a pattern that is dominated by low signals. • Verifies signal mapping by walking an inverted bit across the interface.
3	3’b100	• trace_data[] all zeros • trace_data[] all zeros • trace_data[] all ones • trace_data[] all zeros • trace_data[] all zeros • …	• Stresses power supply/ground of the board/chip by switching from all highs to all lows on the same clock (looking for ground bounces). • Stresses pin to pin skew by driving all signals low for at least 2 bit periods (by ensuring the signals attain the minimum “low logic” voltages before a transition.
4	3’b101	• Up counter from ‘0’s to all ‘1’s and back to ‘0’s	• Pattern is more representative of real data. • Useful for timing training since it’s very easy to look at for increments and wrap arounds.
5	3’b110	• Alternating odd and even bits set/clear	• Every bit switches every clock • Recommended for timing training since the check can be made right at the interface with simple logic.

Figure 41: PTI packets (4-bit mode)

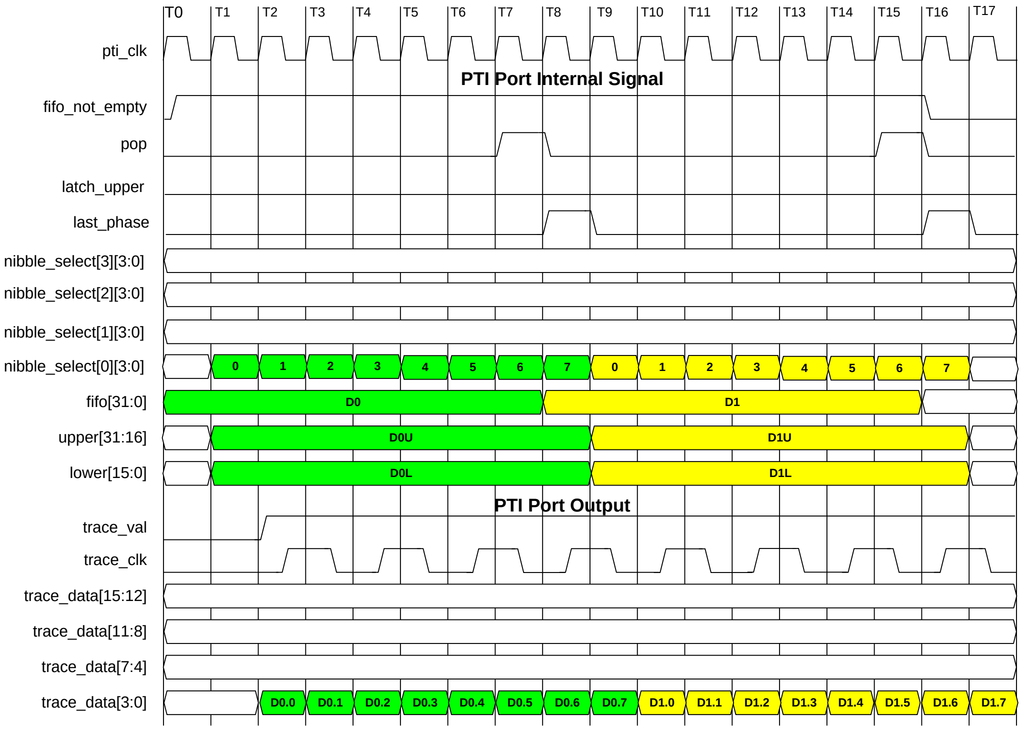


Figure 42: PTI packets (8-bit mode)

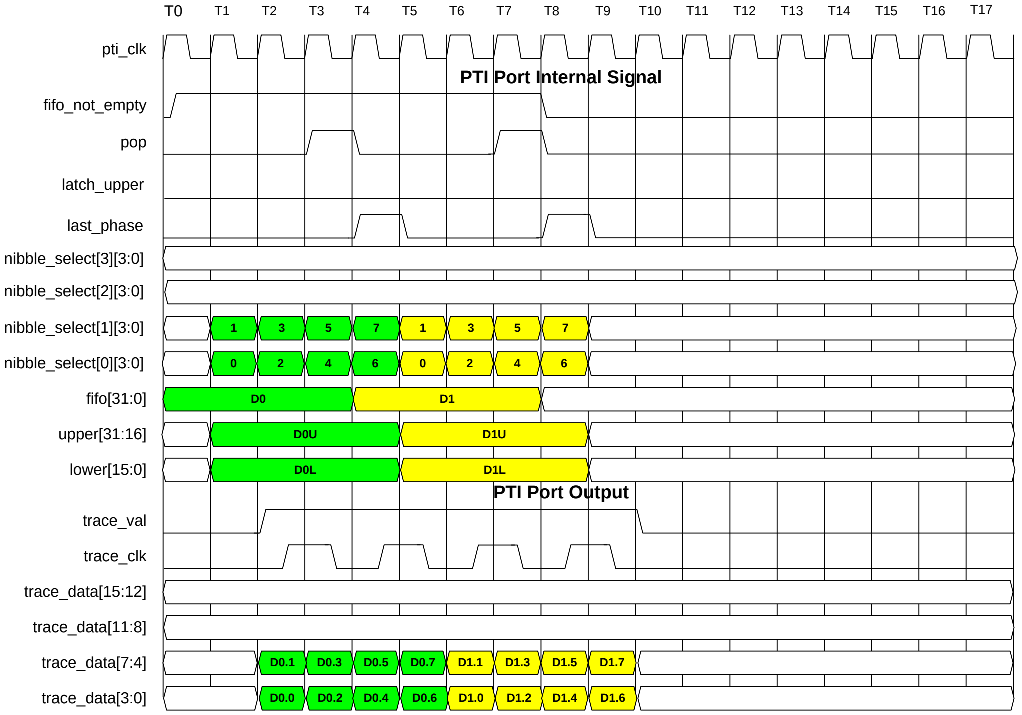
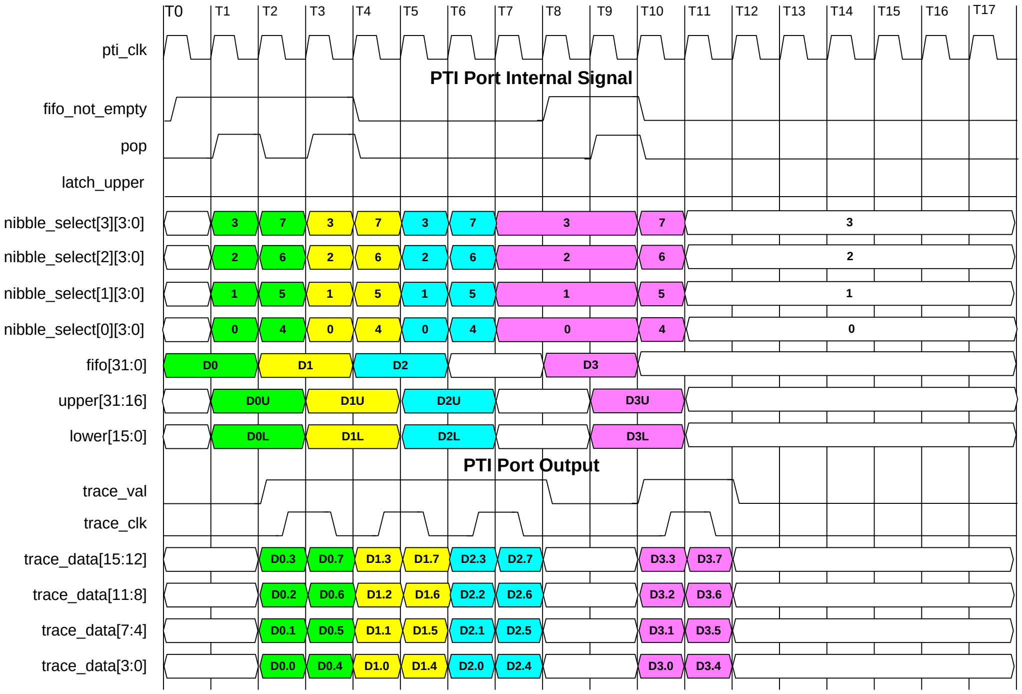


Figure 43: PTI packets (16-bit mode)



10.3.3 Programming Model

The PTI port has to be enabled before sending any trace data to it, or before using it to generate any of the calibration patterns. After enabling, it is recommend driving one or more of the six calibration patterns supported to allow the DTC to adjust its delays for data integrity on the PTI port. Once calibrated, the PTI port is ready to transfer data between the GTH and the DTC.

10.3.4 Performance/Bandwidth Analysis

The PTI port is capable of operating at the Intel Trace Hub clock frequency for a maximum possible bandwidth of (PTI_BPBWID/2)-bits per Intel Trace Hub clock cycles. For example, if running Intel Trace Hub clock at 400MHz with 16 data lanes, the PTI port can transfer data up to a bandwidth of 800MB/s. However, the actual output bandwidth of the PTI port is dependent on the GPIO capabilities and technology.

11 Host Interface

The Host Interface block provides the PCI device presence and memory-mapped IO (MMIO) interface of the Intel® Trace Hub device. The primary fabric interface provides a high-bandwidth interface to the SOC, for both receiving high-bandwidth trace data, and writing STP-encoded trace data to memory (or to the USB3 DbC.Trace endpoint), while the sideband fabric provides an alternative, though lower-bandwidth, interface. The Host Interface block also provides a TAP block, allowing an external JTAG host device access to the same resources (PCI Configuration space, and MMIO space).

11.1 Fabric Multi-Root Space Support

The internal fabric (internal bus) provides support for multiple independent root spaces with their independent address spaces. In this context, a root space is a collection of resources using an independent address space that is orthogonal to all other root spaces. A typical agent is referred to as a single-root space aware is only accessible in its own root space and can only communicate with agents in that root space. On the other hand, an agent that is multi-root space aware is accessible from multiple root spaces (i.e. is mapped to each of the orthogonal address spaces), and can communicate with agent across root spaces. The mechanism by which an agent communicates with other agents across root spaces and how a transaction is routed is beyond the scope of this specification The Intel® Trace Hub is a multi-root space accessible device, accessible in two root spaces, with different addresses in each space.

For the *Host Space*[14] or Root Space 0 (RS0), the Intel® Trace Hub implements four programmable BARs summarized in Table 11-1. For Root Space 1 (RS1), the Intel® Trace Hub can be accessed via two fixed memory windows (“BARs”) as summarized in Table 11-2. In addition, the Intel® Trace Hub can target (perform memory writes) to devices in root space 0. Specifically, the Intel® Trace Hub is designed to write to system memory, and also to a USB controller with a DbC.Trace endpoint. Because of this functionality, the Intel Trace Hub can function as a “bridge” between RS1 and RS0. This may have security or isolation implications, in the case of running multiple virtual machines on a single physical machine. Operating Systems and Virtual Machine Monitors should take appropriate precautions when assigning the Intel Trace Hub to a single virtual machine.

Table 11-1: Host Space Base Address Registers

Intel® Trace Hub MMIO Region	Host Space Base Address
CSR and MSC Trace Buffer (MTB) Region: The CSR/MTB region is a fixed 1MB of MMIO space that includes the memory mapped CSRs, plus the MSC Trace Buffers (see Figure 2 for more details).	CSR_MTB_BAR programmable configuration space register – **BAR0**

[14] *The Host Space is the name used to denote the main IA CPU root space.*

Intel® Trace Hub MMIO Region	Host Space Base Address
Software Trace Memory Region (STMR): the STMR is a variable size (project specific) MMIO space used by software/firmware to send debug messages to the Intel® Trace Hub. For more details please refer to Chapter 4	SW_BAR programmable configuration space register – **BAR1**
Intel® PT Trace Memory Region (RTMR): RTMR is a variable size (project specific) MMIO space used by Intel® PT hardware to send debug messages to the Intel® Trace Hub. For more details please refer to Chapter 4 .	RTIT_BAR programmable configuration space register – **BAR2**
Firmware Trace Memory Region (FTMR): FTMR is a variable size (project specific) MMIO space used by firmware to send debug messages to the Intel® Trace Hub. For more details please refer to Chapter 4	FW_BAR programmable configuration space register – **BAR3**

Table 11-2: Root-Space 1 (RS1) Base Address Registers

Intel® Trace Hub MMIO Region	RS1 Space Base Address
CSR and MSC Trace Buffer (MTB) Region: The CSR/MTB region is a fixed 1MB of MMIO space that includes the memory mapped CSRs, plus the MSC Trace Buffers (see Figure 2 for more details).	0xF740_0000 (**BAR0**)
Firmware Trace Memory Region (FTMR): the FTMR is a variable size (project specific) MMIO space used by firmware to send debug messages to the Intel® Trace Hub. For more details please refer to Chapter 4 .	0xF700_0000 (**BAR3**)

11.2 Error Handling and Reporting

The Intel® Trace Hub is a standard PCI device that supports error logging and reporting per the PCI and internal bus specifications. In this section, error handling and reporting are presented.

From a transaction-level point of view, the Intel® Trace Hub is responsible for handling three types of erroneous transaction types. Those three types can be summarized as follows:

1. ***Unsupported transactions that are always unsupported:*** Any transaction that does not have the *type*, *format*, and *traffic class*, supported by the Intel® Trace Hub is considered always unsupported.

2. ***Supported transactions that violate the Intel® Trace Hub programming model:*** These transactions are supported and have the correct type, format, and traffic class, but might be violating the Intel® Trace Hub programming model or some specific Intel® Trace Hub requirement based on the current Intel® Trace Hub device configuration and state.

3. ***Erroneous downstream completions:*** This is a completion transaction received by the Intel® Trace Hub in response to an outbound non-posted request with a completion status other than successful indicating some error has occurred.

Depending on which of the above transaction type is received, the Intel® Trace Hub will respond and handle the exceptions differently as explained below.

11.2.1 Always Unsupported Requests

Unsupported Requests (UR)s are the category of transactions that are always unsupported by the Intel® Trace Hub independent of the state of the device. The action taken by the Intel® Trace Hub in response to the fabric issuing a command put for an always unsupported request depends on the transaction type as listed in Table 11-3.

Examples of always unsupported requests:

1. A command put for an IOWr or IORd transaction.
2. A command put for a transaction with an IMPS larger than 64B.
3. A command put for a CfgWr transaction with wrong bus/device/function numbers.

Table 11-3: Unsupported Request Error Handling

Request Type	Error Condition	Error Logging	Error Reporting
Posted	Unsupported Request	Sets Unsupported Request Detected (URD) bit in Device Specific PCI Status register	If both the Unsupported Request Reporting Enable (URRE) bit in Device Specific Control register and the SERR Enable bit in PCI Command register are set, then the Signaled System Error (SSE) bit in PCI Status register will be set and a DO_SERR message will be sent on the sideband interface
Non-Posted	Unsupported Request	No	The Completion will be returned with an Unsupported Request (UR) status

11.2.2 Programming Model Unsupported Requests

If the Intel® Trace Hub receives a command put for a transaction targeting the Intel® Trace Hub (it asserts hit or would have asserted hit if using target decode) but the transaction violates the Intel® Trace Hub's programming model, it will have to be handled as a Completer Abort (CA).

Examples of supported requests that violate the Intel® Trace Hub programming model:

1. A command put for a Cfg transaction with a payload larger than 1DW.
2. A command put for a MWr/MRd transaction with MSE not set.
3. A command put for a MWr/MRd to a 32-bit CSR that is not DW aligned.

It is also important to note that even when using target decode a transaction that violates the Intel® Trace Hub's programming model might still be received due to the target pipelined architecture. For example, consider two back-to-back transactions: the first is a CfgWr that clears the MSE bit, and the second is a MWr. Due to the pipelined nature of the agent, both requests would be accepted, but by the time the device is ready to service the MWr, the memory space-enable bit would have been cleared and the second transaction has to be a CA. The action taken by the Intel® Trace Hub in response to a transaction that violates its programming model depends on the type of the transaction as listed in Table 11-4.

Table 11-4: Completer Abort Error Handling

Request Type	Error Condition	Error Logging	Error Reporting
Posted	Completer Abort	Sets Signaled Target Abort (STA) bit in PCI Status register	In PCI Command register, if SERR Enable bit is set, then the Signaled System Error (SSE) bit in PCI Status register will be set and a DO_SERR message will be sent on the sideband interface
Non-Posted	Completer Abort	Sets Signaled Target Abort (STA) bit in PCI Status register	The Completion will be returned with a Completer Abort (CA) status

11.2.3 Erroneous Downstream Completions

The third type of erroneous transaction that is handled by the Intel® Trace Hub relates to downstream completion transactions (generated in response to outbound non-posted transactions from the MSU). In this case, any completion with a status other than "successful" is considered erroneous and has to be handled by the Intel Trace Hub Host Interface. This is necessary to avoid corrupting system memory and even possibly hanging the entire system.

To this end, once a completion transaction is received by the Intel® Trace Hub in response to an issued non-posted transaction with any status other than successful, the primary fabric target agent will immediately signal the MSU to take the appropriate action to prevent any catastrophic events from occurring. The detailed MSU behavior is documented in the MSU chapter of this specification. From an Intel® Trace Hub device perspective, Table 11-5 shows how such an error will be logged in PCI configuration space based on the completion status.

Table 11-5: Unsuccessful completion error handling as a requestor

Request Type	Completion Status	Error Logging	Error Reporting
Completion	Completion Abort	Sets Receive Target Abort (RTA) bit in PCI Status register.	MSU specific. Please consult the MSU specification for a detailed description.

Request Type	Completion Status	Error Logging	Error Reporting
Completion	Unsupported Request	Sets Receive Master Abort (RMA) bit in PCI Status register.	MSU specific. Please consult the MSU specification for a detailed description.

11.2.4 Miscellaneous Downstream Erroneous Transactions

If a downstream transaction does not fall in any of the above categories but can still cause some functional problem, like trying to access a non-existent register or reading a write-only register, the internal bus and PCI specifications require that the device still service the transaction. If this erroneous transaction is a posted transaction, it will silently be dropped. If it is a non-posted transaction, then a data payload of zero will be returned along with a status of Successful Completion (SC).

Examples of miscellaneous erroneous transactions:

1. A MWr transaction targeting the RO MSC Trace Buffer (MTB).
2. A MRd to an empty region in one of the Intel® Trace Hub’s MMIO regions.

To deal with the above types of erroneous transactions, a different approach was needed; the internal fabric agent does not have this level of knowledge about the other Intel® Trace Hub components. For example, the primary fabric target agent does not know if the MRd transaction it received is targeting a non-existent memory region. Hence, the Intel® Trace Hub implements a timeout mechanism that is used to comply with the PCI specification requirements, while also acting as a survivability feature that can prevent internal hangs and blockings.

The timeout mechanism is composed of two timers at the main points of arbitration on the Intel® Trace Hub internal bus architecture. The first is located at the Inbound Switch arbiter and the second at the Inbound Clock Crossing Bridge arbiter.

Once a request is put on the corresponding bus (IPCLK Inbound Bus for the Inbound Switch and the NPCLK Inbound Bus for the Inbound CCB), the timer starts to be decremented. For posted requests, if the timer expires before the targeted block accepts the request, it is immediately dropped to free the bus and accept the next transaction. For non-posted requests if the timer expires before the response has been returned by the targeted unit, the timer will return a successful completion on behalf of the targeted unit and accept the next non-posted request (since there can only be one outstanding non-posted request at any given time).

Both timers are programmable through the Intel® Trace Hub PCI configuration space to accommodate for different functional block latencies. For example, the VIS Register Controller block can be running much slower than other units and might require a much longer time to respond to a given request. For this reason it is crucial that software users make sure to program the values for those timeout timers to a large enough value that would prevent any false timeouts, yet a small enough value that wouldn’t unnecessarily degrade the overall system performance.

For some use cases, the Intel® Trace Hub needs to backpressure the internal bus fabric which, in turn, can backpressure the source of the transaction. In this case, the Intel®

Trace Hub also provides the ability to turn off one or both of those timeout timers by programming them to zero.

It is also important to note that the larger the values programmed into those timeout timers, the larger performance degradation that will be sensed by software. This is, however, a good indication that software is not performing the correct operation and should be debugged.

12 Programming Model

A detailed programming model of all of Intel® Trace Hub features for all use cases is beyond the scope of this document. In this section, we present a suggested programming sequence.

In general, the recommended programming/configuration sequence for the Intel® Trace Hub hardware is:

1. Trace Destinations
2. Trace Sources
3. Mapping sources to destinations (configuring Global Trace Hub)
4. Triggering

More details for each of the above steps are given in the following subsections.

12.1.1 Configuring Trace Destinations

Depending on the use-case, one or more Intel® Trace Hub tracing destinations must be configured as described below.

12.1.1.1 Trace to PTI

Configuring PTI as a trace source is relatively simple. The user will select the operational mode (i.e. 4-bit, 8-bit, or 16-bit), set the SMU maintenance packet frequency for this port, GTH data retention policy, NULL packet generation policy, and then enable PTI port.

If training is also required, the PTI port controller supports generating some test patterns for testing the capture tool. Training is a fully manual operation. That is, there is not a pre-defined training sequence (in hardware terms) for PTI. The user/software can select a training pattern to send to the PTI port. The user would then verify proper capture of that training pattern. Once verified, the user can select a different training pattern. Once the user is satisfied with PTI port training, he/she can disable training and proceed with trace capture.

12.1.1.2 Trace to DCI, BSSB Mode

If trace data will be sent through BSSB, the Intel® Trace Hub DCI Handler has to be configured to BSSB mode (refer to section 13.6 for details). The BSSB-related hardware (DCI Bridge and USB PHY) must also be initialized, though that is out of scope for this document. Both MSCs must be put into DCI mode, and enabled, since they control the source side of the local trace buffer that stores the trace data. As with other trace destinations, the SMU packet frequency should be selected, as well as NULL packet generation and GTH data retention policy (drop/no-drop).

Generally, the configuration sequence for this mode is:

1. Configure the DCI Bridge and BSSB components to stream trace out of the BSSB port (details are outside the scope of this specification).
2. Program MTB Base pointer and DCI TH Credit return register offsets into the appropriate registers of the DCI Bridge
3. Configure the DCI TH with DBC and DCI Base pointers / offsets

a. EXIBASEHI, EXIBASELO registers

b. DBCBASEHI, DBCBASELO registers

c. STREAMCFG1, STREAMCFG2 registers

4.Set DCI TH Timeout values

5.Configure both MSCs for DCI Mode

6.Clear the SETSTRMMODE bit (write 0), and set the ENABLE bit in the STREAMCFG1 register

12.1.1.3 Trace to DCI, DbC (USB) Mode

When tracing to USB (specifically, the DbC.Trace endpoint within the xHCI controller), the USB controller should be discovered and configured first. Next, the Intel® Trace Hub DCI Trace Handler must be enabled to run in DbC.Trace mode, and initialized with pointers to the DbC.Trace endpoint; see section 13.6 for details. As with other trace destinations, the SMU packet frequency should be selected, as well as NULL packet generation and GTH data retention policy (drop/no-drop).

Generally, the configuration sequence for this mode is:

1.Configure the XHCI.DBC Controller for operation as a DbC.Trace endpoint (details are outside the scope of this specification)

2.Write MTB Base address pointer, DCI TH Credit return register offset into the appropriate registers of the XCHI.DbC Controller.

3.Configure the DCI TH with DBC Base pointers / offsets

a. EXIBASEHI, EXIBASELO registers

b. DBCBASEHI, DBCBASELO registers

c. STREAMCFG1, STREAMCFG2 registers

4.Set DCI TH Timeout values

5.Configure both MSCs for DCI Mode

6.Set the SETSTRMMODE bit and set the ENABLE bit in the STREAMCFG1 register.

12.1.1.4 Trace to System Memory, Single-block/CSR-driven mode

2. When tracing to system memory, the first thing that must be done is to allocate system DRAM. For single-buffer, or CSR-driven, mode, a single large, physically contiguous block of system memory must be allocated (typically by the system BIOS).

3. For each MSC, software should write the base address and buffer size for its corresponding memory block in the MSCnBAR and MSCnSIZE register.

4. If interrupt-on-complete operation is desired, software must set up an ISR for the MSU error handling MSI/INTx.

5. As with other trace destinations, the SMU packet frequency should be selected, as well as NULL packet generation and GTH data retention policy (drop/no-drop).

6. Enable the desired MSC(s) by setting the MSCnEN bit, and writing the corresponding value for single-block/CSR-driven mode to the MSCnMODE bits. Options such as wrap_enable may also be programmed prior/simultaneous to setting the enable.

7. When the msu_interrupt is received indicating capture done, software should read the current memory write pointer value from CMWPn and the wrap status from the WRAPSTATn bit to determine the valid trace data in the memory block.

12.1.1.5 Trace to System Memory, Multi-block/Linked-list mode

8. Software should allocate one or more regions of memory to hold the memory blocks for the trace data/linked-list. Software writes the first 32 bytes in the header to configure the address, size and other attributes for each memory block. Software should also clear to 0 the 32 bytes of the hardware header for each memory block so that it can easily identify non-zero values as those written by the MSU.

9. Software should write the address for the first memory block of the first memory window (the head of the linked-list) to the MSCnBAR register.

10. If interrupt-on-complete operation is desired, software must set up an ISR for the MSU error handling MSI/INTx.

11. As with other trace destinations, the SMU packet frequency should be selected, as well as NULL packet generation and GTH data retention policy (drop/no-drop).

12. After setting up all the block headers in memory and programming the first memory block address, software should enable one or more desired MSC(s) by setting the MSCnEN bit, and writing the corresponding value for multi-block/linked-list mode to the MSCnMODE bits.

13. When the msu_interrupt is received indicating capture done, software can read the hardware header in each block of each linked-list/window to determine where the valid trace data and other attributes have been linked.

12.1.1.6 Internal buffer mode

14. Enable the desired MSC(s) by setting the MSCnEN bit, and writing the corresponding value for internal buffer mode to the MSCnMODE bits. Options such as wrap_enable may also be programmed prior/simultaneous to setting the enable.

15. When the capture is complete, software can read the current write pointer from the MSCnTBWP register and the wrap status from the WRAPSTATn bit to determine the valid trace data in each MTB. Software can then directly read the trace data content from each MTB (which is memory-mapped).

16. In internal buffer mode, the MSC responds to inbound read transactions with the addressed trace buffer data. The MSC supports a burst length of 1 on these read transactions (that is, the MSC will not respond to a burst read to the trace buffer). Each trace buffer is 256 entries deep. Each entry is either 8 bytes or 16 bytes wide, depending on MD_WIDTH. If MD_WIDTH = 64 (8 bytes), then MTB0 is addressed at address offset 0x000 to 0x7FF and MTB1 is addressed at address offset 0x8000 to 0x87FF. If MD_WIDTH = 128 (16 bytes), then MTB0 is addressed at address offset 0x000 to 0xFFF and MTB1 is addressed at address offset 0x8000 to 0x8FFF. Note there are offsets not shown that are based on the CSR_MTB_BAR (see Chapter 11).

Note: Reading the MTBs in internal buffer mode while trace data is being stored in the MTBs is not supported. There is no way to prevent the MTB from under-run (reading more data than is actually present). Furthermore, the WRAPSTAT function will not function properly, due to the fact that reading from the MTB changes the read pointer value, which is used in the full and empty calculations.

12.1.2 Configuring Trace Sources

Depending on the use-case, one or more Intel® Trace Hub trace sources must be enabled and configured to capture/forward trace data to the desired trace destination.

12.1.2.1 Visualization of Internal Signals (VIS)

The VIS Register Controller is used to configure the VIS mux tree to route signals from their sources into the Intel® Trace Hub.

12.1.2.2 On-Die Logic Analyzer (ODLA)

The ODLA captures VIS signals into ODLA packets.

12.1.2.3 SoC CHAP (SoCHAP)

The SoCHAPs take VIS signals as input set up in a previous step, and count them according to how they are configured

12.1.2.4 Software Trace Hub (STH)

There is little to set up in the STH to enable tracing software and firmware trace sources; most of the “enabling” is done when configuring the trigger unit. Event generation is the key item to configure, as it will inform the trigger unit of interesting data coming through the STH.

The user/software should set event match/mask registers as needed for event(s) of interest that will come into the STH. Then set up the TRIG_TS Packet payload (TRIGTSPP), if such will be utilized. Thirdly, set up the Backpressure Duration Counter (see STH HAS for more details). Finally, clear the DPLC.DPLCO if necessary.

12.1.3 Mapping Trace Sources to Trace Destinations

Mapping sources to destinations involves two fundamental steps: choosing which source(s) to send to which destination(s), and then writing SWDEST bits in GTH CSRs accordingly. The algorithm or method by which software does this is outside the scope of this specification.

Within the GTH itself, the TSCU destination needs to be chosen to be useful to decode software: it should be sent to same destination as software trace data that will use the TSCU time correlation packets (i.e., trace data that is time-stamped with the CPU’s time stamp).

Next, Enable or Disable trace sources for masters 0-256, as appropriate for the debug scenario.

Set Grant Duration for each source. Also set the Low Water Mark for each source; the LWM must be set to one less than Grant Duration.

Set storage mode for each source (single or multi-buffer).

Set BPB Transmit Threshold (BPBTXLV) for each output port.

12.1.4 Configuring the Trigger Unit

Configuring the Trigger unit can be as simple or as complex as the user desires, from a simple enable-all-sources-immediately, to a complex trigger with multiple start and stop conditions. A full treatise on Trigger Unit configuration is outside the scope of this document. Nonetheless, the following is a brief summary of the two basic methods:

12.1.4.1 Using the Trigger Unit

Configure the states, events, and clauses, according to the sequence needed for achieving the desired trigger condition. Actions in the various states and clauses will likely include enabling one or more trace sources, counting a certain number of clocks before or after taking a certain action, and counting clocks after a trigger to limit the amount of post-trigger data that is captured, among others.

If storing trace data to memory or USB.DbC.Trace, ensure the “Capture Done” signal (store qualifier #0) is asserted at the end of tracing, so that all trace data can be flushed out of the Intel® Trace Hub’s internal pipe and buffers, and sent to the destination. In the case of memory, captureDone assertion can also interrupt the main CPU, indicating it is time to post-process and display the captured data.

12.1.4.2 Manual Override

The “manual override” mode is the equivalent of setting the trigger specification to “If Anything, start storing immediately”, and “stop storing when I tell you.” This is accomplished by software setting the *storeEn* override bits (GTH register SCR) for each trace source to be enabled, and starts the flow of the trace data.

Stopping the flow of trace data is most easily accomplished by software setting the appropriate ForceStoreEnOff bits in the SCR2 register. Using this method also allows software to manually assert the “Capture Done” signal, which will cause all data in the Intel® Trace Hub pipeline and buffers to be flushed to the destination.

12.1.4.3 Intel® Trace Hub Software Reset

The Intel® Trace Hub supports a software-visible configuration register that enables resetting the Intel® Trace Hub logic to an uninitialized state through a software command. The logic blocks that will be reset if this register is written by software are the STH, GTH, Trigger Unit, TSCU, ODLA, SoCHAP, and portions of the MSU.

The software reset can be performed through any of the three interfaces, the primary fabric, the sideband fabric, or TAP. Once the “reset command” is received by the Intel Trace Hub, the reset takes place immediately.

To summarize, the following will occur when a software reset is detected:

- Reset all MMIO control registers.
- Reset all functional logic in the Intel® Trace Hub Functional Blocks.
- Reset all the PCI configuration space registers except the following:
    - MTB_CSR_LBAR and MTB_CSR_UBAR registers.
    - SW_LBAR and SW_UBAR registers.

- RTIT_LBAR and RTIT_UBAR registers.
- FW_LBAR and FW_UBAR registers.
- Intel® Trace Hub Device Specific Control (NPKSDC) register.

Please note that the user of the software reset needs to re-program the Intel® Trace Hub entirely, including its PCI command register, to re-enable it. This software reset is mainly for debug purposes and should not be visible to the Intel® Trace Hub PCI driver since it changes the PCI header contents.

13 Register Description

This chapter provides details for all the registers within the Intel® Trace Hub, including PCI Configuration space registers, memory-mapped I/O registers, and TAP data registers.

13.1 Register Attributes Definitions

The register attributes used in this chapter are listed in Table 13-1.

Most registers within the Intel Trace Hub are reset to their hardware default value by the main power-up reset, called npk_rst_b. The PCI configuration space registers are generally reset by the main system or CPU reset, called “host_rst_b”. The host_rst_b signal is typically only asserted when the “host” (main IA CPU) subsystem experiences a reset (e.g., warm reset).

Table 13-1: Register Attributes

Tag	Name	Description
RW	Read/Write	Bit is readable and writable
WO	Write Only	Bit is only writeable with reads returning zeros.
RO	Read Only	Bit is only readable, but writes have no effects. Read-only should not be used for reserved fields.
RV, Rsvd	Reserved	Bit is reserved. Reads will return 0, and writes have no effect. Reserved tag may be either RV or Rsvd. The “Rsvd” is used to reduce potential confusion with RW because a ‘V’ and ‘W’ may be mistaken when the font is small and .pdf compresses the image of the text.
RW1C	Read/Clear	Reads will return current value. A write with logic 1 causes the bit to be cleared.

13.2 STH Registers

13.2.1 STH Registers Summary

STH Registers			
Offset Start	Offset End	Symbol	Register Name/Function
04000	04003	STHCAP0	STH Capability 0
04004	04007	STHCAP1	STH Capability 1
04008	0400B	TRIG	Create Trigger Packet
0400C	0400F	TRIG_TS	Create Trigger Packet with Time Stamp
04010	04013	XSYNC	Create Cross-Synchronization Packet
04014	04017	XSYNC_TS	Create Cross-

STH Registers

Offset Start	Offset End	Symbol	Register Name/Function
			Synchronization Packet w Time Stamp
04018	0401B	GERR	Create General Error Packet
0401C	0401F	TRIGTSPP	TRIG_TS packet payload
04020	04023	BDC	Backpressure Duration Counter
04024	04027	DPLC	Data Packet Lost Counter
04028	0402B	EV0AMCH	Event 0 Address Match Register
0402C	0402F	EV0AMSK	Event 0 Address Mask Register
04030	04033	EV0DMCH	Event 0 Data Match Register
04034	04037	EV0DMSK	Event 0 Data Mask Register
04038	0403B	EV0CTRL	Event 0 Match/Mask Control Register
0403C	0403F	EV1AMCH	Event 1 Address Match Register
04040	04043	EV1AMSK	Event 1 Address Mask Register
04044	04047	EV1DMCH	Event 1 Data Match Register
04048	0404B	EV1DMSK	Event 1 Data Mask Register
0404C	0404F	EV1CTRL	Event 1 Match/Mask Control Register
04050	04053	EV2AMCH	Event 2 Address Match Register
04054	04057	EV2AMSK	Event 2 Address Mask Register
04058	0405B	EV2DMCH	Event 2 Data Match Register
0405C	0405F	EV2DMSK	Event 2 Data Mask Register
04060	04063	EV2CTRL	Event 2 Match/Mask Control Register
04064	04067	EV3AMCH	Event 3 Address Match

STH Registers			
Offset Start	**Offset End**	**Symbol**	**Register Name/Function**
			Register
04068	0406B	EV3AMSK	Event 3 Address Mask Register
0406C	0406F	EV3DMCH	Event 3 Data Match Register
04070	04073	EV3DMSK	Event 3 Data Mask Register
04074	04077	EV3CTRL	Event 3 Match/Mask Control Register
04078	0407B	STHMISC1	STH MISC Control register 1
0407C	0407F	STHCAP2	STH Capability 2

13.2.2 STHCAP0: STH Capability 0

CSR Register Name: STHCAP0: STH Capability 0				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 04000	**Offset End:** 04003
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:16	RO	FF	STHMNUM	STH Highest Software Master: Indicates the highest numbered software master. Total number of masters is STHMNUM - STHMSTR
15:0	RO	20	STHMSTR	SW Master Start: Indicates the starting master number for the Software Trace Hub as exposed by the PCI BAR SW_BAR (SW_UBAR + SW_LBAR). This is the same as the value of the hardware parameter SW_MSTR_STRT.

13.2.3 STHCAP1: STH Capability 1

CSR Register Name: STHCAP1: STH Capability 1			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 04004	**Offset End:** 04007

Bits	Access	Default	Label	Bit Description
31:24	RW	1F	FW_MSTR	**Firmware Master Stop.** This field indicates the highest Master number that is allocated to firmware trace sources. The default (reset) value of FW_MSTR_STP is set by the synthesis parameter FW_MSTR_STP. Thereafter, it can be written by any software or firmware to adjust the number of masters allocated to firmware. This field is intended to be used in two ways: When the STH is a single PCI BAR, this register field can be changed according to whatever software model best suits the use case(s) for the implementation. When the STH is split into a fixed ACPI BAR for firmware, and a movable PCI BAR for software, then this register field should be only read, and not written. This is because the value in this register reflects the dividing line between the ACPI BAR and the PCI BAR. If the dividing line is moved, software will have an inaccurate view of the STH.
23:20	RO	0	RSVD	STHCAP1 Reserved
19:16	RO	2	RTITCNT	**Number of RTIT Masters.** Indicates the number of RTIT Masters supported in this instantiation of the STH. 0h: RTIT is not implemented and not supported 1h: 1 RTIT master (1 CPU or thread) 2h: 2 RTIT masters 3h: 3 RTIT masters etc. 15h: 15 RTIT masters
15:8	RO	0	RSVD1	STHCAP1 Reserved
7:0	RO	80	CHLCNT	Software Channels Per Master: Indicates the number of channels per master for STH.

13.2.4 TRIG: Create Trigger Packet

CSR Register Name: **TRIG**: Create Trigger Packet				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 04008	**Offset End:** 0400B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:8	RO	0	RSVD	TRIG Reserved

CSR Register Name: TRIG: Create Trigger Packet				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 04008 **Offset End:** 0400B
Bits	**Access**	**Default**	**Label**	**Bit Description**
7:0	WO	0	TRIGDAT	Create Trigger Packet,Trigger Data/Identifier.

13.2.5 TRIG_TS: Create Trigger Packet with Time Stamp

CSR Register Name: TRIG_TS: Create Trigger Packet with Time Stamp				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 0400C **Offset End:** 0400F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:8	RO	0	RSVD	TRIG_TS Resereved
7:0	WO	0	TRIGDAT	Create Trigger Packet with Time Stamp, Trigger Data/Identifier.

13.2.6 XSYNC: Create Cross-Synchronization Packet

CSR Register Name: XSYNC: Create Cross-Synchronization Packet				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 04010 **Offset End:** 04013
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:8	RO	0	RSVD	XSYNC Reserved
7:0	WO	0	XSYNCDAT	Cross-Synchronization Data/Identifier.

13.2.7 XSYNC_TS: Create Cross-Synchronization Packet w Time Stamp

CSR Register Name: XSYNC_TS: Create Cross-Synchronization Packet w Time Stamp				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 04014 **Offset End:** 04017
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:8	RO	0	RSVD	XSYNC_TS Reserved

CSR Register Name: XSYNC_TS: Create Cross-Synchronization Packet w Time Stamp				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 04014	**Offset End:** 04017
Bits	**Access**	**Default**	**Label**	**Bit Description**
7:0	WO	0	XSYNCDAT	Cross-Synchronization Data/Identifier. With Tith Time stamp

13.2.8 GERR: Create General Error Packet

CSR Register Name: GERR: Create General Error Packet				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 04018	**Offset End:** 0401B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:8	RO	0	RSVD	GERR Reserved
7:0	WO	0	GERRDAT	Create General Error Packet,Error Data/Identifier.

13.2.9 TRIGTSPP: TRIG_TS packet payload

CSR Register Name: TRIGTSPP: TRIG_TS packet payload				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 0401C	**Offset End:** 0401F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:9	RO	0	RSVD	TRIGTSPP Reserved
8	RW	0	TRIGEN	**Trigger packet Enable.** This bit determines if a TRIG_TS packet is generated when the trigger input from the CTS is asserted 0: do not create a TRIG packet 1: create a TRIG packet with payload as specified in TRIGVAL.
7:0	RW	0	TRIGVAL	TRIG Value: Specifies the payload in the TRIG_TS packet when the STH detects the trigger input from the CTS is asserted.

13.2.10 BDC: Backpressure Duration Counter

CSR Register Name: **BDC**: Backpressure Duration Counter				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 04020	**Offset End:** 04023
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RW	0	BDC	**Backpressure Duration Count value.** Indicates the number of clock cycles that STH is allowed to backpressure the host interface. Note that the default value indicates that STH will start dropping as soon as it cannot process any more data.

13.2.11 DPLC: Data Packet Lost Counter

CSR Register Name: **DPLC**: Data Packet Lost Counter				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 04024	**Offset End:** 04027
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:17	RO	0	RSVD	DPLC Reserved
16	RO/V	0	DPLCO	Data Packets Lost Counter Overflow: this is a sticky bit that indicates that the DPLC value has wrapped around at least once. This will be sent as the most significant bit of the number of packets lost after existing Drop Mode
15:0	RO	0	DPLCV	Data Packets Lost Counter Value: The number of packets (not including triggers initiated by CTS) that have been dropped during a backpressure release period.

13.2.12 EV0AMCH: Event 0 Address Match Register

CSR Register Name: **EV0AMCH**: Event 0 Address Match Register			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 04028	**Offset End:** 0402B

Bits	Access	Default	Label	Bit Description
31:0	RW	0	ADDRMT	**Event 0 Address Match Register:Address match.** Specifies the match value for the address. This effectively maps this event detector to a specific master, channel, and packet type.

13.2.13 EV0AMSK: Event 0 Address Mask Register

CSR Register Name: EV0AMSK: Event 0 Address Mask Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 0402C **Offset End:** 0402F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RW	0	ADDRMK	**Event 0 Address Mask Register:Address mask.** Specifies the mask value for the address. 0: Means the corresponding bit is not included in the match. 1: Means the corresponding bit is included in the match. Note: To trigger an event at least 1-bit of this register has to be set

13.2.14 EV0DMCH: Event 0 Data Match Register

CSR Register Name: EV0DMCH: Event 0 Data Match Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 04030 **Offset End:** 04033
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:8	RO	0	RSVD	Event 0 Data Match Register: Reserved
7:0	RW	0	DATAMT	**Event 0 Data Match Register:Data match.** Specifies the match value for the 8bits of data.

13.2.15 EV0DMSK: Event 0 Data Mask Register

CSR Register Name: EV0DMSK: Event 0 Data Mask Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 04034 / **Offset End:** 04037
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:8	RO	0	RSVD	Event 0 Data Mask Register: Reserved
7:0	RW	0	DATAMSK	**Event 0 Data Mask Register:Data Mask.** Specifies the mask value for the data. 0: Means the corresponding bit is not included in the match. 1: Means the corresponding bit is included in the match. Note: To trigger an event at least 1-bit of this register has to be set

13.2.16 EV0CTRL: Event 0 Match/Mask Control Register

CSR Register Name: EV0CTRL: Event 0 Match/Mask Control Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 04038 / **Offset End:** 0403B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:16	RO	0	RSVD	EV0CTRL Reserved
15:8	RO	0	RSVD2	EV0CTRL Reserved
7:6	RW	0	BAR_MCH	**BAR Match bits.** Specifies the match value for the BAR (Base Address Register). 00: BAR 0, CSR_MTB_BAR 01: BAR 1, STMR BAR. 10: BAR 2, RTIT BAR 11: BAR 3, Firmware BAR (FW_BAR)
5:4	RW	0	BAR_MSK	**BAR Mask bits.** Specifies the mask value for the BAR (Base Address Register). 0: Means the corresponding bit is not included in the match. 1: Means the corresponding bit is included in the match.
3	RW	0	INVMATCH	**Invert Match: When set, makes the trigger match function either NAND or NOR, depending upon the state of bit 2.** 0: The match function type is as specified by bit 2. 1: The match function type is inverted from the specification of bit 2.

CSR Register Name: EV0CTRL: Event 0 Match/Mask Control Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 04038	**Offset End:** 0403B

Bits	Access	Default	Label	Bit Description
2	RW	0	ANDMATCH	**AND Match: Sets the type of the trigger match function.** 0:The match function is OR. When any bit enabled by its mask bit matches the value of its match bit the Match output is asserted. 1: The match function is AND. All bits enabled by their mask bits must match the value of their match bits for the Match output to be asserted.
1	RO	0	RSVD3	EV0CTRL Reserved
0	RW	0	ENEVT0	**Enable Event 0.** 0: do not generate Event 0 signal to CTS when Event 0 is detected 1: generate Event 0 signal to CTS when Event 0 is detected.

13.2.17 EV1AMCH: Event 1 Address Match Register

CSR Register Name: EV1AMCH: Event 1 Address Match Register			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 0403C	**Offset End:** 0403F

Bits	Access	Default	Label	Bit Description
31:0	RW	0	ADDRMT	**Event 1 Address Match Register:Address match.** Specifies the match value for the address. This effectively maps this event detector to a specific master, channel, and packet type.

13.2.18 EV1AMSK: Event 1 Address Mask Register

CSR Register Name: EV1AMSK: Event 1 Address Mask Register			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 04040	**Offset End:** 04043

Bits	Access	Default	Label	Bit Description
31:0	RW	0	ADDRMK	**Event 1 Address Mask Register:Address mask.** Specifies the mask value for the address. 0: Means the corresponding bit is not included in the match. 1: Means the corresponding bit is included in the match. Note: To trigger an event at least 1-bit of this register has to be set

13.2.19 EV1DMCH: Event 1 Data Match Register

CSR Register Name: **EV1DMCH**: Event 1 Data Match Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 04044 / **Offset End:** 04047
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:8	RO	0	RSVD	EV1DMCH Reserved
7:0	RW	0	DATAMT	**Event 1 Data Match Register:Data match.** Specifies the match value for the 8bits of data.

13.2.20 EV1DMSK: Event 1 Data Mask Register

CSR Register Name: **EV1DMSK**: Event 1 Data Mask Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 04048 / **Offset End:** 0404B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:8	RO	0	RSVD	EV1DMSK Reserved
7:0	RW	0	DATAMSK	**Event 1 Data Mask Register:Data Mask.** Specifies the mask value for the data. 0: Means the corresponding bit is not included in the match. 1: Means the corresponding bit is included in the match. Note: To trigger an event at least 1-bit of this register has to be set

13.2.21 EV1CTRL: Event 1 Match/Mask Control Register

CSR Register Name: EV1CTRL: Event 1 Match/Mask Control Register				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 0404C	**Offset End:** 0404F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:16	RO	0	RSVD	EV1CTRL Reserved
15:8	RO	0	RSVD2	EV1CTRL Reserved
7:6	RW	0	BAR_MCH	**Event 1 Match/Mask Control Register: BAR Match bits.** Specifies the match value for the BAR (Base Address Register). 00: BAR 0, CSR_MTB_BAR 01: BAR 1, STMR BAR. 10: BAR 2, RTIT BAR 11: BAR 3, Firmware BAR (FW_BAR)
5:4	RW	0	BAR_MSK	**BAR Mask bits.** Specifies the mask value for the BAR (Base Address Register). 0: Means the corresponding bit is not included in the match. 1: Means the corresponding bit is included in the match.
3	RW	0	INVMATCH	**Invert Match: When set, makes the trigger match function either NAND or NOR, depending upon the state of bit 2.** 0: The match function type is as specified by bit 2. 1: The match function type is inverted from the specification of bit 2.
2	RW	0	ANDMATCH	**AND Match: Sets the type of the trigger match function.** 0:The match function is OR. When any bit enabled by its mask bit matches the value of its match bit the Match output is asserted. 1: The match function is AND. All bits enabled by their mask bits must match the value of their match bits for the Match output to be asserted.
1	RO	0	RSVD3	EV1CTRL Reserved
0	RW	0	ENEVT1	**Enable Event 1.** 0: do not generate Event 1 signal to CTS when Event 1 is detected 1: generate Event 1 signal to CTS when Event 1 is detected.

13.2.22 EV2AMCH: Event 2 Address Match Register

CSR Register Name: EV2AMCH: Event 2 Address Match Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 04050	**Offset End:** 04053
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RW	0	ADDRMT	**Event 2 Address Match Register:Address match.** Specifies the match value for the address. This effectively maps this event detector to a specific master, channel, and packet type.

13.2.23 EV2AMSK: Event 2 Address Mask Register

CSR Register Name: EV2AMSK: Event 2 Address Mask Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 04054	**Offset End:** 04057
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RW	0	ADDRMK	**Event 2 Address Match Register:Address mask.** Specifies the mask value for the address. 0: Means the corresponding bit is not included in the match. 1: Means the corresponding bit is included in the match. Note: To trigger an event at least 1-bit of this register has to be set

13.2.24 EV2DMCH: Event 2 Data Match Register

CSR Register Name: EV2DMCH: Event 2 Data Match Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 04058	**Offset End:** 0405B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:8	RO	0	RSVD	EV2DMCH Reserved

CSR Register Name: **EV2DMCH**: Event 2 Data Match Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 04058	**Offset End:** 0405B
Bits	**Access**	**Default**	**Label**	**Bit Description**
7:0	RW	0	DATAMT	**Event 2 Data Match Register: Data match.** Specifies the match value for the 8bits of data.

13.2.25 EV2DMSK: Event 2 Data Mask Register

CSR Register Name: **EV2DMSK**: Event 2 Data Mask Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 0405C	**Offset End:** 0405F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:8	RO	0	RSVD	EV2DMSK Reserved
7:0	RW	0	DATAMSK	**Data Mask.** Specifies the mask value for the data. 0: Means the corresponding bit is not included in the match. 1: Means the corresponding bit is included in the match. Note: To trigger an event at least 1-bit of this register has to be set

13.2.26 EV2CTRL: Event 2 Match/Mask Control Register

CSR Register Name: **EV2CTRL**: Event 2 Match/Mask Control Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 04060	**Offset End:** 04063
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:16	RO	0	RSVD	EV2CTRL Reserved
15:8	RO	0	RSVD2	EV2CTRL Reserved

CSR Register Name: EV2CTRL: Event 2 Match/Mask Control Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 04060	**Offset End:** 04063
Bits	**Access**	**Default**	**Label**	**Bit Description**
7:6	RW	0	BAR_MCH	**Event 2 Match / Mask Control Register: BAR Match bits.** Specifies the match value for the BAR (Base Address Register). 00: BAR 0, CSR_MTB_BAR 01: BAR 1, STMR BAR. 10: BAR 2, RTIT BAR 11: BAR 3, Firmware BAR (FW_BAR)
5:4	RW	0	BAR_MSK	**BAR Mask bits.** Specifies the mask value for the BAR (Base Address Register). 0: Means the corresponding bit is not included in the match. 1: Means the corresponding bit is included in the match.
3	RW	0	INVMATCH	**Invert Match: When set, makes the trigger match function either NAND or NOR, depending upon the state of bit 2.** 0: The match function type is as specified by bit 2. 1: The match function type is inverted from the specification of bit 2.
2	RW	0	ANDMATCH	**AND Match: Sets the type of the trigger match function.** 0:The match function is OR. When any bit enabled by its mask bit matches the value of its match bit the Match output is asserted. 1: The match function is AND. All bits enabled by their mask bits must match the value of their match bits for the Match output to be asserted.
1	RO	0	RSVD3	EV2CTRL Reserved
0	RW	0	ENEVT2	**Enable Event 2.** 0: do not generate Event 1 signal to CTS when Event 1 is detected 1: generate Event 1 signal to CTS when Event 1 is detected.

13.2.27 EV3AMCH: Event 3 Address Match Register

CSR Register Name: EV3AMCH: Event 3 Address Match Register					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 04064	**Offset End:** 04067
Bits	**Access**	**Default**	**Label**	**Bit Description**	
31:0	RW	0	ADDRMT	**Event 3 Address Match Register:Address match.** Specifies the match value for the address. This effectively maps this event detector to a specific master, channel, and packet type.	

13.2.28 EV3AMSK: Event 3 Address Mask Register

CSR Register Name: EV3AMSK: Event 3 Address Mask Register					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 04068	**Offset End:** 0406B
Bits	**Access**	**Default**	**Label**	**Bit Description**	
31:0	RW	0	ADDRMK	**Event 3 Address Match Register:Address mask.** Specifies the mask value for the address. 0: Means the corresponding bit is not included in the match. 1: Means the corresponding bit is included in the match. Note: To trigger an event at least 1-bit of this register has to be set	

13.2.29 EV3DMCH: Event 3 Data Match Register

CSR Register Name: EV3DMCH: Event 3 Data Match Register					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 0406C	**Offset End:** 0406F
Bits	**Access**	**Default**	**Label**	**Bit Description**	
31:8	RO	0	RSVD	EV3DMCH Reserved	

CSR Register Name: EV3DMCH: Event 3 Data Match Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 0406C **Offset End:** 0406F
Bits	**Access**	**Default**	**Label**	**Bit Description**
7:0	RW	0	DATAMT	**Event 3 Data Match Register: Data match.** Specifies the match value for the 8bits of data.

13.2.30 EV3DMSK: Event 3 Data Mask Register

CSR Register Name: EV3DMSK: Event 3 Data Mask Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 04070 **Offset End:** 04073
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:8	RO	0	RSVD	EV3DMSK Reserved
7:0	RW	0	DATAMSK	**Data Mask.** Specifies the mask value for the data. 0: Means the corresponding bit is not included in the match. 1: Means the corresponding bit is included in the match. Note: To trigger an event at least 1-bit of this register has to be set

13.2.31 EV3CTRL: Event 3 Match/Mask Control Register

CSR Register Name: EV3CTRL: Event 3 Match/Mask Control Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 04074 **Offset End:** 04077
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:16	RO	0	RSVD	EV3CTRL Reserved
15:8	RO	0	RSVD2	EV3CTRL Reserved

<table>
<tr><td colspan="6"><b>CSR Register Name:</b> <b>EV3CTRL</b>: Event 3 Match/Mask Control Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b>
04074</td><td><b>Offset End:</b>
04077</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>7:6</td><td>RW</td><td>0</td><td>BAR_MCH</td><td colspan="2"><b>Event 3 Match/Mask Control Register: BAR Match bits.</b> Specifies the match value for the BAR (Base Address Register).
00: BAR 0, CSR_MTB_BAR
01: BAR 1, STMR BAR.
10: BAR 2, RTIT BAR
11: BAR 3, Firmware BAR (FW_BAR)</td></tr>
<tr><td>5:4</td><td>RW</td><td>0</td><td>BAR_MSK</td><td colspan="2"><b>BAR Mask bits.</b> Specifies the mask value for the BAR (Base Address Register).
0: Means the corresponding bit is not included in the match.
1: Means the corresponding bit is included in the match.</td></tr>
<tr><td>3</td><td>RW</td><td>0</td><td>INVMATCH</td><td colspan="2"><b>Invert Match: When set, makes the trigger match function either NAND or NOR, depending upon the state of bit 2.</b>
0: The match function type is as specified by bit 2.
1: The match function type is inverted from the specification of bit 2.</td></tr>
<tr><td>2</td><td>RW</td><td>0</td><td>ANDMATCH</td><td colspan="2"><b>AND Match: Sets the type of the trigger match function.</b>
0:The match function is OR. When any bit enabled by its mask bit matches the value of its match bit the Match output is asserted.
1: The match function is AND. All bits enabled by their mask bits must match the value of their match bits for the Match output to be asserted.</td></tr>
<tr><td>1</td><td>RO</td><td>0</td><td>RSVD3</td><td colspan="2">EV3CTRL Reserved</td></tr>
<tr><td>0</td><td>RW</td><td>0</td><td>ENEVT3</td><td colspan="2"><b>Enable Event 2.</b>
0: do not generate Event 1 signal to CTS when Event 1 is detected
1: generate Event 1 signal to CTS when Event 1 is detected.</td></tr>
</table>

13.2.32 STHMISC1: STH MISC Control register 1

CSR Register Name: STHMISC1: STH MISC Control register 1				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 04078	**Offset End:** 0407B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RO	0	RSVD	STHMISC1 Reserved

13.2.33 STHCAP2: STH Capability 2

CSR Register Name: STHCAP2: STH Capability 2				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 0407C	**Offset End:** 0407F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:16	RO	0	RSVD	STHCAP2 Reserved
15:0	RO	8	FW_MSTR_STRT	**Firmware Master Start.** Specifies the starting Master number for the firmware ACPI BAR If the firmware ACPI BAR is not present, this field will read all zeros.

13.3 Global Trace Hub Registers

13.3.1 Global Trace Hub Registers Summary

Global Trace Hub Registers			
Offset Start	**Offset End**	**Symbol**	**Register Name/Function**
0	3	GTHOPT0	GTH Output Ports 0-3
4	7	GTHOPT1	GTH Output Ports 4-7
8	B	SWDEST_0	Switching Destination [0]
C	F	SWDEST_1	Switching Destination [1]
10	13	SWDEST_2	Switching Destination [2]
14	17	SWDEST_3	Switching Destination

Global Trace Hub Registers			
Offset Start	Offset End	Symbol	Register Name/Function
			[3]
18	1B	SWDEST_4	Switching Destination [4]
1C	1F	SWDEST_5	Switching Destination [5]
20	23	SWDEST_6	Switching Destination [6]
24	27	SWDEST_7	Switching Destination [7]
28	2B	SWDEST_8	Switching Destination [8]
2C	2F	SWDEST_9	Switching Destination [9]
30	33	SWDEST_10	Switching Destination [10]
34	37	SWDEST_11	Switching Destination [11]
38	3B	SWDEST_12	Switching Destination [12]
3C	3F	SWDEST_13	Switching Destination [13]
40	43	SWDEST_14	Switching Destination [14]
44	47	SWDEST_15	Switching Destination [15]
48	4B	SWDEST_16	Switching Destination [16]
4C	4F	SWDEST_17	Switching Destination [17]
50	53	SWDEST_18	Switching Destination [18]
54	57	SWDEST_19	Switching Destination [19]
58	5B	SWDEST_20	Switching Destination [20]
5C	5F	SWDEST_21	Switching Destination [21]
60	63	SWDEST_22	Switching Destination [22]
64	67	SWDEST_23	Switching Destination [23]

Global Trace Hub Registers			
Offset Start	**Offset End**	**Symbol**	**Register Name/Function**
68	6B	SWDEST_24	Switching Destination [24]
6C	6F	SWDEST_25	Switching Destination [25]
70	73	SWDEST_26	Switching Destination [26]
74	77	SWDEST_27	Switching Destination [27]
78	7B	SWDEST_28	Switching Destination [28]
7C	7F	SWDEST_29	Switching Destination [29]
80	83	SWDEST_30	Switching Destination [30]
84	87	SWDEST_31	Switching Destination [31]
88	8B	GSWDEST	General Software Trace Destination
8C	8F	LWM0	Low WaterMark for Sources 0-7
90	93	LWM1	Low WaterMark for Sources 7-15
94	97	GTH_INFO_1	GTH Parameter Info 1
98	9B	GTH_MISC	GTH Miscellaneous Register
9C	9F	SMCR0	STP Maintenance Control Register 0
A0	A3	SMCR1	STP Maintenance Control Register 1
A4	A7	SMCR2	STP Maintenance Control Register 2
A8	AB	SMCR3	STP Maintenance Control Register 3
AC	AF	PGD0	Programmable Grant Duration Register 0
B0	B3	PGD1	Programmable Grant Duration Register 1
C8	CB	SCR	Source Control Register
CC	CF	GTHFRQ	GTH Frequency Register

Global Trace Hub Registers			
Offset Start	**Offset End**	**Symbol**	**Register Name/Function**
D0	D3	GTHRSVD	GTH Reserved
D4	D7	GTHSTAT	GTH Status
D8	DB	SCR2	Source Control Register 2
DC	DF	DESTOVR	Destination Override Register
E0	E3	SCRPD0	ScratchPad 0 Register
E4	E7	SCRPD1	ScratchPad 1 Register
E8	EB	SCRPD2	ScratchPad 2 Register
EC	EF	SCRPD3	ScratchPad 3 Register

13.3.2 GTHOPT0: GTH Output Ports 0-3

CSR Register Name: **GTHOPT0**: GTH Output Ports 0-3				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 0	**Offset End:** 3
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	0	P3FLUSH	**Port 3 Flush.** Setting this bit will assert the flush signal to the byte packing buffer for port 3
30	RW	0	RSVD	Reserved for Future Use
29	RO/V	0	P3RST	**Port 3 in Reset.** When asserted, indicates the port is in reset, and is not accepting data. When cleared, indicates the port is operational, and can accept trace data.
28	RW	0	P3DRP	**GTH Data Retention Policy for Port 3.** See P0DRP for details.
27	RW	0	P3NULL	NULL Packet Generation for output port 3
26:24	RO	0	P3TYPE	GTH Output Port 3 type. 0h: No port 1h - System Memory / USB (DCI) 2h - reserved 3h - reserved 4h - MIPI-PTI Others - reserved
23	RW	0	P2FLUSH	**Port 2 Flush.** Setting this bit will assert the flush signal to the byte packing buffer for port 2

CSR Register Name: GTHOPT0: GTH Output Ports 0-3				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 0	**Offset End:** 3
Bits	**Access**	**Default**	**Label**	**Bit Description**
22	RW	0	RSVD	Reserved for Future Use
21	RO/V	0	P2RST	**Port 2 in Reset.** When asserted, indicates the port is in reset, and is not accepting data. When cleared, indicates the port is operational, and can accept trace data.
20	RW	0	P2DRP	**GTH Data Retention Policy for Port 2.** See P0DRP for details.
19	RW	0	P2NULL	NULL Packet Generation for output port 2.
18:16	RO	4	P2TYPE	GTH Output Port 2 type. 0h: No port 1h - System Memory / USB (DCI) 2h - reserved 3h - reserved 4h - MIPI-PTI Others - reserved
15	RW	0	P1FLUSH	**Port 1 Flush.** Setting this bit will assert the flush signal to the byte packing buffer for port 1.
14	RW	0	RSVD	Reserved for Future Use
13	RO/V	0	P1RST	**Port 1 in Reset.** When asserted, indicates the port is in reset, and is not accepting data. When cleared, indicates the port is operational, and can accept trace data.
12	RW	0	P1DRP	**GTH Data Retention Policy for Port 1.** See P0DRP for details.
11	RW	0	P1NULL	NULL Packet Generation for output port 1.
10:8	RO	1	P1TYPE	GTH Output Port 1 type. 0h: No port 1h - System Memory / USB (DCI) 2h - reserved 3h - reserved 4h - MIPI-PTI Others - reserved
7	RW	0	P0FLUSH	**Port 0 Flush.** Setting this bit will assert the flush signal to the byte packing buffer for port 0.
6	RW	0	RSVD	Reserved for Future Use

<table>
<tr><td colspan="6">**CSR Register Name: GTHOPT0**: GTH Output Ports 0-3</td></tr>
<tr><td colspan="2">**Bar:** CSR_MTB_BAR</td><td colspan="2">**Reset:** npk_rst_b</td><td>**Offset Start:** 0</td><td>**Offset End:** 3</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>5</td><td>RO/V</td><td>0</td><td>P0RST</td><td colspan="2">**Port 0 in Reset.** When asserted, indicates the port is in reset, and is not accepting data. When cleared, indicates the port is operational, and can accept trace data.</td></tr>
<tr><td>4</td><td>RW</td><td>0</td><td>P0DRP</td><td colspan="2">**GTH Data Retention Policy for Port 0.** This bit defines the behavior of the GTH when Port 0 is in a not ready or reset condition. Specifically, it is when portReset is asserted. The conditions under which the portReset signal is asserted for a given port is defined by the port's logic, and is outside the scope of this specification. This condition might occur when the output port is unconfigured, held in reset, or otherwise not fully operational. When the portReset is asserted, the BPB will take action based on the setting of the PnDRP bit. Specifically:
0: Hold/retain data. In this mode, the GTH will hold, or retain, all the trace data it has. The Byte Packing Buffer will very quickly fill up, and stall its data path. When the affected Input Buffers fill up, they will de-assert their get signal, indicating to their trace source(s) that they cannot accept any more input data.
1: Drop data. In this mode, the Byte Packing Buffer will ignore the deassertion of its get input, behaving as if it is asserted continuously. This will have the net effect of unloading data from the BPB and dropping it on the floor (it is permanently lost).</td></tr>
<tr><td>3</td><td>RW</td><td>0</td><td>P0NULL</td><td colspan="2">**NULL Packet Generation for output port 0.**
0: NULL Packets are suppressed
1: NULL Packets are generated</td></tr>
<tr><td>2:0</td><td>RO</td><td>1</td><td>P0TYPE</td><td colspan="2">GTH Output Port 0 type.
0h: No port
1h - System Memory / USB (DCI)
2h - reserved
3h - reserved
4h - MIPI-PTI
Others - reserved</td></tr>
</table>

13.3.3 GTHOPT1: GTH Output Ports 4-7

CSR Register Name: GTHOPT1: GTH Output Ports 4-7				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 4 | **Offset End:** 7
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	0	P7FLUSH	**Port 7 Flush.** Setting this bit will assert the flush signal to the byte packing buffer for port 7.
30	RW	0	RSVD	Reserved for Future Use
29	RO/V	0	P7RST	**Port 7 in Reset.** When asserted, indicates the port is in reset, and is not accepting data. When cleared, indicates the port is operational, and can accept trace data.
28	RW	0	P7DRP	**GTH Data Retention Policy for Port 7.** See P0DRP for details.
27	RW	0	P7NULL	**NULL Packet Generation for output port 7.** 0: NULL Packets are suppressed 1: NULL Packets are generated
26:24	RO	0	P7TYPE	GTH Output Port 7 type. 0h: No port 1h - System Memory / USB (DCI) 2h - reserved 3h - reserved 4h - MIPI-PTI Others - reserved
23	RW	0	P6FLUSH	**Port 6 Flush.** Setting this bit will assert the flush signal to the byte packing buffer for port 6.
22	RW	0	RSVD	Reserved for Future Use
21	RO/V	0	P6RST	**Port 6 in Reset.** When asserted, indicates the port is in reset, and is not accepting data. When cleared, indicates the port is operational, and can accept trace data.
20	RW	0	P6DRP	**GTH Data Retention Policy for Port 6.** See P0DRP for details.
19	RW	0	P6NULL	**NULL Packet Generation for output port 6.** 0: NULL Packets are suppressed 1: NULL Packets are generated
18:16	RO	0	P6TYPE	GTH Output Port 6 type. 0h: No port 1h - System Memory / USB (DCI) 2h - reserved 3h - reserved 4h - MIPI-PTI Others - reserved

<table>
<tr><td colspan="6"><b>CSR Register Name: GTHOPT1</b>: GTH Output Ports 4-7</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 4</td><td><b>Offset End:</b> 7</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>15</td><td>RW</td><td>0</td><td>P5FLUSH</td><td colspan="2"><b>Port 5 Flush.</b> Setting this bit will assert the flush signal to the byte packing buffer for port 5.</td></tr>
<tr><td>14</td><td>RW</td><td>0</td><td>RSVD</td><td colspan="2">Reserved for Future Use</td></tr>
<tr><td>13</td><td>RO/V</td><td>0</td><td>P5RST</td><td colspan="2"><b>Port 5 in Reset.</b> When asserted, indicates the port is in reset, and is not accepting data. When cleared, indicates the port is operational, and can accept trace data.</td></tr>
<tr><td>12</td><td>RW</td><td>0</td><td>P5DRP</td><td colspan="2"><b>GTH Data Retention Policy for Port 5.</b> See P0DRP for details.</td></tr>
<tr><td>11</td><td>RW</td><td>0</td><td>P5NULL</td><td colspan="2"><b>NULL Packet Generation for output port 5.</b>
0: NULL Packets are suppressed
1: NULL Packets are generated</td></tr>
<tr><td>10:8</td><td>RO</td><td>0</td><td>P5TYPE</td><td colspan="2">GTH Output Port 5 type.
0h: No port
1h - System Memory / USB (DCI)
2h - reserved
3h - reserved
4h - MIPI-PTI
Others - reserved</td></tr>
<tr><td>7</td><td>RW</td><td>0</td><td>P4FLUSH</td><td colspan="2"><b>Port 4 Flush.</b> Setting this bit will assert the flush signal to the byte packing buffer for port 4.</td></tr>
<tr><td>6</td><td>RW</td><td>0</td><td>RSVD</td><td colspan="2">Reserved for Future Use</td></tr>
<tr><td>5</td><td>RO/V</td><td>0</td><td>P4RST</td><td colspan="2"><b>Port 4 in Reset.</b> When asserted, indicates the port is in reset, and is not accepting data. When cleared, indicates the port is operational, and can accept trace data.</td></tr>
<tr><td>4</td><td>RW</td><td>0</td><td>P4DRP</td><td colspan="2"><b>GTH Data Retention Policy for Port 4.</b> See P0DRP for details.</td></tr>
<tr><td>3</td><td>RW</td><td>0</td><td>P4NULL</td><td colspan="2"><b>NULL Packet Generation for output port 4.</b>
0: NULL Packets are suppressed
1: NULL Packets are generated</td></tr>
<tr><td>2:0</td><td>RO</td><td>0</td><td>P4TYPE</td><td colspan="2">GTH Output Port 4 type.
0h: No port
1h - System Memory / USB (DCI)
2h - reserved
3h - reserved
4h - MIPI-PTI
Others - reserved</td></tr>
</table>

13.3.4 SWDEST_0: Switching Destination [0]

CSR Register Name: SWDEST_0: Switching Destination [0]

Bar: CSR_MTB_BAR | **Reset:** npk_rst_b | **Offset Start:** 8 | **Offset End:** B

Bits	Access	Default	Label	Bit Description
31	RW	1	MAST7EN	**Master 7 Enable.** See MAST0EN for definition.
30:28	RW	0	MAST7DEST	**Master 7 Destination.** See MAST1DEST for definition.
27	RW	1	MAST6EN	**Master 6 Enable.** See MAST0EN for definition.
26:24	RW	0	MAST6DEST	**Master 6 Destination.** See MAST1DEST for definition.
23	RW	1	MAST5EN	**Master 5 Enable.** See MAST0EN for definition.
22:20	RW	0	MAST5DEST	**Master 4 Enable.** See MAST0EN for definition.
19	RW	1	MAST4EN	**Master 4 Enable.** See MAST0EN for definition.
18:16	RW	0	MAST4DEST	**Master 4 Destination.** See MAST1DEST for definition.
15	RW	1	MAST3EN	**Master 3 Enable.** See MAST0EN for definition.
14:12	RW	0	MAST3DEST	**Master 3 Destination.** See MAST1DEST for definition.
11	RW	1	MAST2EN	**Master 2 Enable.** See MAST0EN for definition.
10:8	RW	0	MAST2DEST	**Master 2 Destination.** See MAST0DEST for definition.
7	RW	1	MAST1EN	**Master 1 Enable.** See MAST0EN for definition.
6:4	RW	0	MAST1DEST	**Master 1 Destination.** See MAST0DEST for definition.
3	RW	1	MAST0EN	**Master 0 Enable.** Enables tracing for Master 0. 0: tracing for Master 0 is disabled. All data received from Master 0 is received at the Input Buffer and immediately dropped. 1: tracing for Master 0 is enabled (Default)

CSR Register Name: SWDEST_0: Switching Destination [0]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 8 **Offset End:** B
Bits	**Access**	**Default**	**Label**	**Bit Description**
2:0	RW	0	MAST0DEST	**Master 0 Destination.** Specifies the destination port number (refer to GTHOPT register for details) for Master 0 trace data. 0x0: Master 0 is routed to Output Port 0 0x1: Master 0 is routed to Output Port 1 ... 0x7: Master 0 is routed to Output Port 0x7 The default value of this bit field is controlled by the corresponding MDEST[N] parameter.

13.3.5 SWDEST_1: Switching Destination [1]

CSR Register Name: SWDEST_1: Switching Destination [1]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** C **Offset End:** F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	1	MAST15EN	**Master 15 Enable.** See MAST8EN for definition.
30:28	RW	0	MAST15DEST	**Master 15 Destination.** See MAST8DEST for definition.
27	RW	1	MAST14EN	**Master 14 Enable.** See MAST8EN for definition.
26:24	RW	0	MAST14DEST	**Master 14 Destination.** See MAST8DEST for definition.
23	RW	1	MAST13EN	**Master 13 Enable.** See MAST8EN for definition.
22:20	RW	0	MAST13DEST	**Master 13 Destination.** See MAST8DEST for definition.
19	RW	1	MAST12EN	**Master 12 Enable.** See MAST8EN for definition.
18:16	RW	0	MAST12DEST	**Master 12 Destination.** See MAST8DEST for definition.
15	RW	1	MAST11EN	**Master 11 Enable.** See MAST8EN for definition.
14:12	RW	0	MAST11DEST	**Master 11 Destination.** See MAST8DEST for definition.
11	RW	1	MAST10EN	**Master 10 Enable.** See MAST8EN for definition.

CSR Register Name: SWDEST_1: Switching Destination [1]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** C	**Offset End:** F
Bits	**Access**	**Default**	**Label**	**Bit Description**
10:8	RW	0	MAST10DEST	**Master 10 Destination.** See MAST8DEST for definition.
7	RW	1	MAST9EN	**Master 9 Enable.** See MAST8EN for definition.
6:4	RW	0	MAST9DEST	**Master 9 Destination.** See MAST8DEST for definition.
3	RW	1	MAST8EN	**Master 8 Enable.** Enables tracing for Master 8. 0: tracing for Master 8 is disabled. All data received from Master 8 is received at the Input Buffer and immediately dropped. 1: tracing for Master 8 is enabled (Default)
2:0	RW	0	MAST8DEST	**Master 8 Destination.** Specifies the destination port number (refer to GTHOPT register for details) for Master 8 trace data. 0x0: Master 8 is routed to Output Port 0 0x1: Master 8 is routed to Output Port 1 ... 0x7: Master 8 is routed to Output Port 0x7 The default value of this bit field is controlled by the corresponding MDEST[N] parameter.

13.3.6 SWDEST_2: Switching Destination [2]

CSR Register Name: SWDEST_2: Switching Destination [2]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 10	**Offset End:** 13
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	1	MAST23EN	**Master 23 Enable.** See MAST16EN for definition.
30:28	RW	0	MAST23DEST	**Master 23 Destination.** See MAST16DEST for definition.
27	RW	1	MAST22EN	**Master 22 Enable.** See MAST16EN for definition.
26:24	RW	0	MAST22DEST	**Master 22 Destination.** See MAST16DEST for definition.

CSR Register Name: SWDEST_2: Switching Destination [2]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 10 / **Offset End:** 13
Bits	**Access**	**Default**	**Label**	**Bit Description**
23	RW	1	MAST21EN	**Master 21 Enable.** See MAST16EN for definition.
22:20	RW	0	MAST21DEST	**Master 21 Destination.** See MAST16DEST for definition.
19	RW	1	MAST20EN	**Master 20 Enable.** See MAST16EN for definition.
18:16	RW	0	MAST20DEST	**Master 20 Destination.** See MAST16DEST for definition.
15	RW	1	MAST19EN	**Master 19 Enable.** See MAST16EN for definition.
14:12	RW	0	MAST19DEST	**Master 19 Destination.** See MAST16DEST for definition.
11	RW	1	MAST18EN	**Master 18 Enable.** See MAST16EN for definition.
10:8	RW	0	MAST18DEST	**Master 18 Destination.** See MAST16DEST for definition.
7	RW	1	MAST17EN	**Master 17 Enable.** See MAST16EN for definition.
6:4	RW	0	MAST17DEST	**Master 17 Destination.** See MAST16DEST for definition.
3	RW	1	MAST16EN	**Master 16 Enable.** Enables tracing for Master 16. 0: tracing for Master 16 is disabled. All data received from Master 16 is received at the Input Buffer and immediately dropped. 1: tracing for Master 16 is enabled (Default)
2:0	RW	0	MAST16DEST	**Master 16 Destination.** Specifies the destination port number (refer to GTHOPT register for details) for Master 16 trace data. 0x0: Master 16 is routed to Output Port 0 0x1: Master 16 is routed to Output Port 1 … 0x7: Master 16 is routed to Output Port 0x7 The default value of this bit field is controlled by the corresponding MDEST[N] parameter.

13.3.7 SWDEST_3: Switching Destination [3]

CSR Register Name: SWDEST_3: Switching Destination [3]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 14	**Offset End:** 17
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	1	MAST31EN	**Master 31 Enable.** See MAST24EN for definition.
30:28	RW	0	MAST31DEST	**Master 31 Destination.** See MAST24DEST for definition.
27	RW	1	MAST30EN	**Master 30 Enable.** See MAST24EN for definition.
26:24	RW	0	MAST30DEST	**Master 30 Destination.** See MAST24DEST for definition.
23	RW	1	MAST29EN	**Master 29 Enable.** See MAST24EN for definition.
22:20	RW	0	MAST29DEST	**Master 29 Destination.** See MAST24DEST for definition.
19	RW	1	MAST28EN	**Master 28 Enable.** See MAST24EN for definition.
18:16	RW	0	MAST28DEST	**Master 28 Destination.** See MAST24DEST for definition.
15	RW	1	MAST27EN	**Master 27 Enable.** See MAST24EN for definition.
14:12	RW	0	MAST27DEST	**Master 27 Destination.** See MAST24DEST for definition.
11	RW	1	MAST26EN	**Master 26 Enable.** See MAST24EN for definition.
10:8	RW	0	MAST26DEST	**Master 26 Destination.** See MAST24DEST for definition.
7	RW	1	MAST25EN	**Master 25 Enable.** See MAST24EN for definition.
6:4	RW	0	MAST25DEST	**Master 25 Destination.** See MAST24DEST for definition.
3	RW	1	MAST24EN	**Master 24 Enable.** Enables tracing for Master 24. 0: tracing for Master 24 is disabled. All data received from Master 24 is received at the Input Buffer and immediately dropped. 1: tracing for Master 24 is enabled (Default)

<table>
<tr><td colspan="6"><b>CSR Register Name: SWDEST_3</b>: Switching Destination [3]</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 14</td><td><b>Offset End:</b> 17</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>2:0</td><td>RW</td><td>0</td><td>MAST24DEST</td><td colspan="2"><b>Master 24 Destination.</b> Specifies the destination port number (refer to GTHOPT register for details) for Master 24 trace data.
0x0: Master 24 is routed to Output Port 0
0x1: Master 24 is routed to Output Port 1
...
0x7: Master 24 is routed to Output Port 0x7
The default value of this bit field is controlled by the corresponding MDEST[N] parameter.</td></tr>
</table>

13.3.8 SWDEST_4: Switching Destination [4]

<table>
<tr><td colspan="6"><b>CSR Register Name: SWDEST_4</b>: Switching Destination [4]</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 18</td><td><b>Offset End:</b> 1B</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31</td><td>RW</td><td>1</td><td>MAST39EN</td><td colspan="2"><b>Master 39 Enable.</b> See MAST32EN for definition.</td></tr>
<tr><td>30:28</td><td>RW</td><td>0</td><td>MAST39DEST</td><td colspan="2"><b>Master 39 Destination.</b> See MAST32DEST for definition.</td></tr>
<tr><td>27</td><td>RW</td><td>1</td><td>MAST38EN</td><td colspan="2"><b>Master 38 Enable.</b> See MAST32EN for definition.</td></tr>
<tr><td>26:24</td><td>RW</td><td>0</td><td>MAST38DEST</td><td colspan="2"><b>Master 38 Destination.</b> See MAST32DEST for definition.</td></tr>
<tr><td>23</td><td>RW</td><td>1</td><td>MAST37EN</td><td colspan="2"><b>Master 37 Enable.</b> See MAST32EN for definition.</td></tr>
<tr><td>22:20</td><td>RW</td><td>0</td><td>MAST37DEST</td><td colspan="2"><b>Master 37 Destination.</b> See MAST32DEST for definition.</td></tr>
<tr><td>19</td><td>RW</td><td>1</td><td>MAST36EN</td><td colspan="2"><b>Master 36 Enable.</b> See MAST32EN for definition.</td></tr>
<tr><td>18:16</td><td>RW</td><td>0</td><td>MAST36DEST</td><td colspan="2"><b>Master 36 Destination.</b> See MAST32DEST for definition.</td></tr>
<tr><td>15</td><td>RW</td><td>1</td><td>MAST35EN</td><td colspan="2"><b>Master 35 Enable.</b> See MAST32EN for definition.</td></tr>
<tr><td>14:12</td><td>RW</td><td>0</td><td>MAST35DEST</td><td colspan="2"><b>Master 35 Destination.</b> See MAST32DEST for definition.</td></tr>
</table>

CSR Register Name: **SWDEST_4**: Switching Destination [4]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 18	**Offset End:** 1B
Bits	**Access**	**Default**	**Label**	**Bit Description**
11	RW	1	MAST34EN	**Master 34 Enable.** See MAST32EN for definition.
10:8	RW	0	MAST34DEST	**Master 34 Destination.** See MAST32DEST for definition.
7	RW	1	MAST33EN	**Master 33 Enable.** See MAST32EN for definition.
6:4	RW	0	MAST33DEST	**Master 33 Destination.** See MAST32DEST for definition.
3	RW	1	MAST32EN	**Master 32 Enable.** Enables tracing for Master 32. 0: tracing for Master 32 is disabled. All data received from Master 32 is received at the Input Buffer and immediately dropped. 1: tracing for Master 32 is enabled (Default)
2:0	RW	0	MAST32DEST	**Master 32 Destination.** Specifies the destination port number (refer to GTHOPT register for details) for Master 32 trace data. 0x0: Master 32 is routed to Output Port 0 0x1: Master 32 is routed to Output Port 1 ... 0x7: Master 32 is routed to Output Port 0x7 The default value of this bit field is controlled by the corresponding MDEST[N] parameter.

13.3.9 SWDEST_5: Switching Destination [5]

CSR Register Name: **SWDEST_5**: Switching Destination [5]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 1C	**Offset End:** 1F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	1	MAST47EN	**Master 47 Enable.** See MAST40EN for definition.
30:28	RW	0	MAST47DEST	**Master 47 Destination.** See MAST40DEST for definition.
27	RW	1	MAST46EN	**Master 46 Enable.** See MAST40EN for definition.

CSR Register Name: SWDEST_5: Switching Destination [5]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 1C **Offset End:** 1F
Bits	**Access**	**Default**	**Label**	**Bit Description**
26:24	RW	0	MAST46DEST	**Master 46 Destination.** See MAST40DEST for definition.
23	RW	1	MAST45EN	**Master 45 Enable.** See MAST40EN for definition.
22:20	RW	0	MAST45DEST	**Master 45 Destination.** See MAST40DEST for definition.
19	RW	1	MAST44EN	**Master 44 Enable.** See MAST40EN for definition.
18:16	RW	0	MAST44DEST	**Master 44 Destination.** See MAST40DEST for definition.
15	RW	1	MAST43EN	**Master 43 Enable.** See MAST40EN for definition.
14:12	RW	0	MAST43DEST	**Master 43 Destination.** See MAST40DEST for definition.
11	RW	1	MAST42EN	**Master 42 Enable.** See MAST40EN for definition.
10:8	RW	0	MAST42DEST	**Master 42 Destination.** See MAST40DEST for definition.
7	RW	1	MAST41EN	**Master 41 Enable.** See MAST40EN for definition.
6:4	RW	0	MAST41DEST	**Master 41 Destination.** See MAST40DEST for definition.
3	RW	1	MAST40EN	**Master 40 Enable.** Enables tracing for Master 40. 0: tracing for Master 40 is disabled. All data received from Master 40 is received at the Input Buffer and immediately dropped. 1: tracing for Master 40 is enabled (Default)
2:0	RW	0	MAST40DEST	**Master 40 Destination.** Specifies the destination port number (refer to GTHOPT register for details) for Master 40 trace data. 0x0: Master 40 is routed to Output Port 0 0x1: Master 40 is routed to Output Port 1 ... 0x7: Master 40 is routed to Output Port 0x7 The default value of this bit field is controlled by the corresponding MDEST[N] parameter.

13.3.10 SWDEST_6: Switching Destination [6]

CSR Register Name: SWDEST_6: Switching Destination [6]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 20	**Offset End:** 23
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	1	MAST55EN	**Master 55 Enable.** See MAST48EN for definition.
30:28	RW	0	MAST55DEST	**Master 55 Destination.** See MAST48DEST for definition.
27	RW	1	MAST54EN	**Master 54 Enable.** See MAST48EN for definition.
26:24	RW	0	MAST54DEST	**Master 54 Destination.** See MAST48DEST for definition.
23	RW	1	MAST53EN	**Master 53 Enable.** See MAST48EN for definition.
22:20	RW	0	MAST53DEST	**Master 53 Destination.** See MAST48DEST for definition.
19	RW	1	MAST52EN	**Master 52 Enable.** See MAST48EN for definition.
18:16	RW	0	MAST52DEST	**Master 52 Destination.** See MAST48DEST for definition.
15	RW	1	MAST51EN	**Master 51 Enable.** See MAST48EN for definition.
14:12	RW	0	MAST51DEST	**Master 51 Destination.** See MAST48DEST for definition.
11	RW	1	MAST50EN	**Master 50 Enable.** See MAST48EN for definition.
10:8	RW	0	MAST50DEST	**Master 50 Destination.** See MAST48DEST for definition.
7	RW	1	MAST49EN	**Master 49 Enable.** See MAST48EN for definition.
6:4	RW	0	MAST49DEST	**Master 49 Destination.** See MAST48DEST for definition.
3	RW	1	MAST48EN	**Master 48 Enable.** Enables tracing for Master 48. 0: tracing for Master 48 is disabled. All data received from Master 48 is received at the Input Buffer and immediately dropped. 1: tracing for Master 48 is enabled (Default)

CSR Register Name: SWDEST_6: Switching Destination [6]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 20	**Offset End:** 23
Bits	**Access**	**Default**	**Label**	**Bit Description**
2:0	RW	0	MAST48DEST	**Master 48 Destination.** Specifies the destination port number (refer to GTHOPT register for details) for Master 48 trace data. 0x0: Master 48 is routed to Output Port 0 0x1: Master 48 is routed to Output Port 1 ... 0x7: Master 48 is routed to Output Port 0x7 The default value of this bit field is controlled by the corresponding MDEST[N] parameter.

13.3.11 SWDEST_7: Switching Destination [7]

CSR Register Name: SWDEST_7: Switching Destination [7]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 24	**Offset End:** 27
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	1	MAST63EN	**Master 63 Enable.** See MAST56EN for definition.
30:28	RW	0	MAST63DEST	**Master 63 Destination.** See MAST56DEST for definition.
27	RW	1	MAST62EN	**Master 62 Enable.** See MAST56EN for definition.
26:24	RW	0	MAST62DEST	**Master 62 Destination.** See MAST56DEST for definition.
23	RW	1	MAST61EN	**Master 61 Enable.** See MAST56EN for definition.
22:20	RW	0	MAST61DEST	**Master 61 Destination.** See MAST56DEST for definition.
19	RW	1	MAST60EN	**Master 60 Enable.** See MAST56EN for definition.
18:16	RW	0	MAST60DEST	**Master 60 Destination.** See MAST56DEST for definition.
15	RW	1	MAST59EN	**Master 59 Enable.** See MAST56EN for definition.
14:12	RW	0	MAST59DEST	**Master 59 Destination.** See MAST56DEST for definition.

CSR Register Name: SWDEST_7: Switching Destination [7]					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 24	**Offset End:** 27
Bits	**Access**	**Default**	**Label**	**Bit Description**	
11	RW	1	MAST58EN	**Master 58 Enable.** See MAST56EN for definition.	
10:8	RW	0	MAST58DEST	**Master 58 Destination.** See MAST56DEST for definition.	
7	RW	1	MAST57EN	**Master 57 Enable.** See MAST56EN for definition.	
6:4	RW	0	MAST57DEST	**Master 57 Destination.** See MAST56DEST for definition.	
3	RW	1	MAST56EN	**Master 56 Enable.** Enables tracing for Master 56. 0: tracing for Master 56 is disabled. All data received from Master 56 is received at the Input Buffer and immediately dropped. 1: tracing for Master 56 is enabled (Default)	
2:0	RW	0	MAST56DEST	**Master 56 Destination.** Specifies the destination port number (refer to GTHOPT register for details) for Master 56 trace data. 0x0: Master 56 is routed to Output Port 0 0x1: Master 56 is routed to Output Port 1 ... 0x7: Master 56 is routed to Output Port 0x7 The default value of this bit field is controlled by the corresponding MDEST[N] parameter.	

13.3.12 SWDEST_8: Switching Destination [8]

CSR Register Name: SWDEST_8: Switching Destination [8]					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 28	**Offset End:** 2B
Bits	**Access**	**Default**	**Label**	**Bit Description**	
31	RW	1	MAST71EN	**Master 71 Enable.** See MAST64EN for definition.	
30:28	RW	0	MAST71DEST	**Master 71 Destination.** See MAST64DEST for definition.	
27	RW	1	MAST70EN	**Master 70 Enable.** See MAST64EN for definition.	

CSR Register Name: SWDEST_8: Switching Destination [8]			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 28	**Offset End:** 2B

Bits	Access	Default	Label	Bit Description
26:24	RW	0	MAST70DEST	**Master 70 Destination.** See MAST64DEST for definition.
23	RW	1	MAST69EN	**Master 69 Enable.** See MAST64EN for definition.
22:20	RW	0	MAST69DEST	**Master 69 Destination.** See MAST64DEST for definition.
19	RW	1	MAST68EN	**Master 68 Enable.** See MAST64EN for definition.
18:16	RW	0	MAST68DEST	**Master 68 Destination.** See MAST64DEST for definition.
15	RW	1	MAST67EN	**Master 67 Enable.** See MAST64EN for definition.
14:12	RW	0	MAST67DEST	**Master 67 Destination.** See MAST64DEST for definition.
11	RW	1	MAST66EN	**Master 66 Enable.** See MAST64EN for definition.
10:8	RW	0	MAST66DEST	**Master 66 Destination.** See MAST64DEST for definition.
7	RW	1	MAST65EN	**Master 65 Enable.** See MAST64EN for definition.
6:4	RW	0	MAST65DEST	**Master 65 Destination.** See MAST64DEST for definition.
3	RW	1	MAST64EN	**Master 64 Enable.** Enables tracing for Master 64. 0: tracing for Master 64 is disabled. All data received from Master 64 is received at the Input Buffer and immediately dropped. 1: tracing for Master 64 is enabled (Default)
2:0	RW	0	MAST64DEST	**Master 64 Destination.** Specifies the destination port number (refer to GTHOPT register for details) for Master 64 trace data. 0x0: Master 64 is routed to Output Port 0 0x1: Master 64 is routed to Output Port 1 ... 0x7: Master 64 is routed to Output Port 0x7 The default value of this bit field is controlled by the corresponding MDEST[N] parameter.

13.3.13 SWDEST_9: Switching Destination [9]

CSR Register Name: SWDEST_9: Switching Destination [9]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 2C	**Offset End:** 2F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	1	MAST79EN	**Master 79 Enable.** See MAST72EN for definition.
30:28	RW	0	MAST79DEST	**Master 79 Destination.** See MAST72DEST for definition.
27	RW	1	MAST78EN	**Master 78 Enable.** See MAST72EN for definition.
26:24	RW	0	MAST78DEST	**Master 78 Destination.** See MAST72DEST for definition.
23	RW	1	MAST77EN	**Master 77 Enable.** See MAST72EN for definition.
22:20	RW	0	MAST77DEST	**Master 77 Destination.** See MAST72DEST for definition.
19	RW	1	MAST76EN	**Master 76 Enable.** See MAST72EN for definition.
18:16	RW	0	MAST76DEST	**Master 76 Destination.** See MAST72DEST for definition.
15	RW	1	MAST75EN	**Master 75 Enable.** See MAST72EN for definition.
14:12	RW	0	MAST75DEST	**Master 75 Destination.** See MAST72DEST for definition.
11	RW	1	MAST74EN	**Master 74 Enable.** See MAST72EN for definition.
10:8	RW	0	MAST74DEST	**Master 74 Destination.** See MAST72DEST for definition.
7	RW	1	MAST73EN	**Master 73 Enable.** See MAST72EN for definition.
6:4	RW	0	MAST73DEST	**Master 73 Destination.** See MAST72DEST for definition.
3	RW	1	MAST72EN	**Master 72 Enable.** Enables tracing for Master 72. 0: tracing for Master 72 is disabled. All data received from Master 72 is received at the Input Buffer and immediately dropped. 1: tracing for Master 72 is enabled (Default)

CSR Register Name: SWDEST_9: Switching Destination [9]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 2C	**Offset End:** 2F
Bits	**Access**	**Default**	**Label**	**Bit Description**
2:0	RW	0	MAST72DEST	**Master 72 Destination.** Specifies the destination port number (refer to GTHOPT register for details) for Master 72 trace data. 0x0: Master 72 is routed to Output Port 0 0x1: Master 72 is routed to Output Port 1 ... 0x7: Master 72 is routed to Output Port 0x7 The default value of this bit field is controlled by the corresponding MDEST[N] parameter.

13.3.14 SWDEST_10: Switching Destination [10]

CSR Register Name: SWDEST_10: Switching Destination [10]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 30	**Offset End:** 33
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	1	MAST87EN	**Master 87 Enable.** See MAST80EN for definition.
30:28	RW	0	MAST87DEST	**Master 87 Destination.** See MAST80DEST for definition.
27	RW	1	MAST86EN	**Master 86 Enable.** See MAST80EN for definition.
26:24	RW	0	MAST86DEST	**Master 86 Destination.** See MAST80DEST for definition.
23	RW	1	MAST85EN	**Master 85 Enable.** See MAST80EN for definition.
22:20	RW	0	MAST85DEST	**Master 85 Destination.** See MAST80DEST for definition.
19	RW	1	MAST84EN	**Master 84 Enable.** See MAST80EN for definition.
18:16	RW	0	MAST84DEST	**Master 84 Destination.** See MAST80DEST for definition.
15	RW	1	MAST83EN	**Master 83 Enable.** See MAST80EN for definition.
14:12	RW	0	MAST83DEST	**Master 83 Destination.** See MAST80DEST for definition.

CSR Register Name: SWDEST_10: Switching Destination [10]					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 30	**Offset End:** 33
Bits	**Access**	**Default**	**Label**	**Bit Description**	
11	RW	1	MAST82EN	**Master 82 Enable.** See MAST80EN for definition.	
10:8	RW	0	MAST82DEST	**Master 82 Destination.** See MAST80DEST for definition.	
7	RW	1	MAST81EN	**Master 81 Enable.** See MAST80EN for definition.	
6:4	RW	0	MAST81DEST	**Master 81 Destination.** See MAST80DEST for definition.	
3	RW	1	MAST80EN	**Master 80 Enable.** Enables tracing for Master 80. 0: tracing for Master 80 is disabled. All data received from Master 80 is received at the Input Buffer and immediately dropped. 1: tracing for Master 80 is enabled (Default)	
2:0	RW	0	MAST80DEST	**Master 80 Destination.** Specifies the destination port number (refer to GTHOPT register for details) for Master 80 trace data. 0x0: Master 80 is routed to Output Port 0 0x1: Master 80 is routed to Output Port 1 ... 0x7: Master 80 is routed to Output Port 0x7 The default value of this bit field is controlled by the corresponding MDEST[N] parameter.	

13.3.15 SWDEST_11: Switching Destination [11]

CSR Register Name: SWDEST_11: Switching Destination [11]					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 34	**Offset End:** 37
Bits	**Access**	**Default**	**Label**	**Bit Description**	
31	RW	1	MAST95EN	**Master 95 Enable.** See MAST88EN for definition.	
30:28	RW	0	MAST95DEST	**Master 95 Destination.** See MAST88DEST for definition.	
27	RW	1	MAST94EN	**Master 94 Enable.** See MAST88EN for definition.	

CSR Register Name: SWDEST_11: Switching Destination [11]

Bar: CSR_MTB_BAR | **Reset:** npk_rst_b | **Offset Start:** 34 | **Offset End:** 37

Bits	Access	Default	Label	Bit Description
26:24	RW	0	MAST94DEST	**Master 94 Destination.** See MAST88DEST for definition.
23	RW	1	MAST93EN	**Master 93 Enable.** See MAST88EN for definition.
22:20	RW	0	MAST93DEST	**Master 93 Destination.** See MAST88DEST for definition.
19	RW	1	MAST92EN	**Master 92 Enable.** See MAST88EN for definition.
18:16	RW	0	MAST92DEST	**Master 92 Destination.** See MAST88DEST for definition.
15	RW	1	MAST91EN	**Master 91 Enable.** See MAST88EN for definition.
14:12	RW	0	MAST91DEST	**Master 91 Destination.** See MAST88DEST for definition.
11	RW	1	MAST90EN	**Master 90 Enable.** See MAST88EN for definition.
10:8	RW	0	MAST90DEST	**Master 90 Destination.** See MAST88DEST for definition.
7	RW	1	MAST89EN	**Master 89 Enable.** See MAST88EN for definition.
6:4	RW	0	MAST89DEST	**Master 89 Destination.** See MAST88DEST for definition.
3	RW	1	MAST88EN	**Master 88 Enable.** Enables tracing for Master 88. 0: tracing for Master 88 is disabled. All data received from Master 88 is received at the Input Buffer and immediately dropped. 1: tracing for Master 88 is enabled (Default)
2:0	RW	0	MAST88DEST	**Master 88 Destination.** Specifies the destination port number (refer to GTHOPT register for details) for Master 88 trace data. 0x0: Master 88 is routed to Output Port 0 0x1: Master 88 is routed to Output Port 1 ... 0x7: Master 88 is routed to Output Port 0x7 The default value of this bit field is controlled by the corresponding MDEST[N] parameter.

13.3.16 SWDEST_12: Switching Destination [12]

<table>
<tr><td colspan="6"><b>CSR Register Name:</b> <b>SWDEST_12</b>: Switching Destination [12]</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 38</td><td><b>Offset End:</b> 3B</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31</td><td>RW</td><td>1</td><td>MAST103EN</td><td colspan="2"><b>Master 103 Enable.</b> See MAST96EN for definition.</td></tr>
<tr><td>30:28</td><td>RW</td><td>0</td><td>MAST103DEST</td><td colspan="2"><b>Master 103 Destination.</b> See MAST96DEST for definition.</td></tr>
<tr><td>27</td><td>RW</td><td>1</td><td>MAST102EN</td><td colspan="2"><b>Master 102 Enable.</b> See MAST96EN for definition.</td></tr>
<tr><td>26:24</td><td>RW</td><td>0</td><td>MAST102DEST</td><td colspan="2"><b>Master 102 Destination.</b> See MAST96DEST for definition.</td></tr>
<tr><td>23</td><td>RW</td><td>1</td><td>MAST101EN</td><td colspan="2"><b>Master 101 Enable.</b> See MAST96EN for definition.</td></tr>
<tr><td>22:20</td><td>RW</td><td>0</td><td>MAST101DEST</td><td colspan="2"><b>Master 101 Destination.</b> See MAST96DEST for definition.</td></tr>
<tr><td>19</td><td>RW</td><td>1</td><td>MAST100EN</td><td colspan="2"><b>Master 100 Enable.</b> See MAST96EN for definition.</td></tr>
<tr><td>18:16</td><td>RW</td><td>0</td><td>MAST100DEST</td><td colspan="2"><b>Master 100 Destination.</b> See MAST96DEST for definition.</td></tr>
<tr><td>15</td><td>RW</td><td>1</td><td>MAST99EN</td><td colspan="2"><b>Master 99 Enable.</b> See MAST96EN for definition.</td></tr>
<tr><td>14:12</td><td>RW</td><td>0</td><td>MAST99DEST</td><td colspan="2"><b>Master 99 Destination.</b> See MAST96DEST for definition.</td></tr>
<tr><td>11</td><td>RW</td><td>1</td><td>MAST98EN</td><td colspan="2"><b>Master 98 Enable.</b> See MAST96EN for definition.</td></tr>
<tr><td>10:8</td><td>RW</td><td>0</td><td>MAST98DEST</td><td colspan="2"><b>Master 98 Destination.</b> See MAST96DEST for definition.</td></tr>
<tr><td>7</td><td>RW</td><td>1</td><td>MAST97EN</td><td colspan="2"><b>Master 97 Enable.</b> See MAST96EN for definition.</td></tr>
<tr><td>6:4</td><td>RW</td><td>0</td><td>MAST97DEST</td><td colspan="2"><b>Master 97 Destination.</b> See MAST96DEST for definition.</td></tr>
<tr><td>3</td><td>RW</td><td>1</td><td>MAST96EN</td><td colspan="2"><b>Master 96 Enable.</b> Enables tracing for Master 96.
0: tracing for Master 96 is disabled. All data received from Master 96 is received at the Input Buffer and immediately dropped.
1: tracing for Master 96 is enabled (Default)</td></tr>
</table>

<table>
<tr><td colspan="6"><b>CSR Register Name: SWDEST_12</b>: Switching Destination [12]</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 38</td><td><b>Offset End:</b> 3B</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>2:0</td><td>RW</td><td>0</td><td>MAST96DEST</td><td colspan="2"><b>Master 96 Destination.</b> Specifies the destination port number (refer to GTHOPT register for details) for Master 96 trace data.
0x0: Master 96 is routed to Output Port 0
0x1: Master 96 is routed to Output Port 1
...
0x7: Master 96 is routed to Output Port 0x7
The default value of this bit field is controlled by the corresponding MDEST[N] parameter.</td></tr>
</table>

13.3.17 SWDEST_13: Switching Destination [13]

<table>
<tr><td colspan="6"><b>CSR Register Name: SWDEST_13</b>: Switching Destination [13]</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 3C</td><td><b>Offset End:</b> 3F</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31</td><td>RW</td><td>1</td><td>MAST111EN</td><td colspan="2"><b>Master 111 Enable.</b> See MAST104EN for definition.</td></tr>
<tr><td>30:28</td><td>RW</td><td>0</td><td>MAST111DEST</td><td colspan="2"><b>Master 111 Destination.</b> See MAST104DEST for definition.</td></tr>
<tr><td>27</td><td>RW</td><td>1</td><td>MAST110EN</td><td colspan="2"><b>Master 110 Enable.</b> See MAST104EN for definition.</td></tr>
<tr><td>26:24</td><td>RW</td><td>0</td><td>MAST110DEST</td><td colspan="2"><b>Master 110 Destination.</b> See MAST104DEST for definition.</td></tr>
<tr><td>23</td><td>RW</td><td>1</td><td>MAST109EN</td><td colspan="2"><b>Master 109 Enable.</b> See MAST104EN for definition.</td></tr>
<tr><td>22:20</td><td>RW</td><td>0</td><td>MAST109DEST</td><td colspan="2"><b>Master 109 Destination.</b> See MAST104DEST for definition.</td></tr>
<tr><td>19</td><td>RW</td><td>1</td><td>MAST108EN</td><td colspan="2"><b>Master 108 Enable.</b> See MAST104EN for definition.</td></tr>
<tr><td>18:16</td><td>RW</td><td>0</td><td>MAST108DEST</td><td colspan="2"><b>Master 108 Destination.</b> See MAST104DEST for definition.</td></tr>
<tr><td>15</td><td>RW</td><td>1</td><td>MAST107EN</td><td colspan="2"><b>Master 107 Enable.</b> See MAST104EN for definition.</td></tr>
<tr><td>14:12</td><td>RW</td><td>0</td><td>MAST107DEST</td><td colspan="2"><b>Master 107 Destination.</b> See MAST104DEST for definition.</td></tr>
</table>

CSR Register Name: SWDEST_13: Switching Destination [13]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 3C	**Offset End:** 3F
Bits	**Access**	**Default**	**Label**	**Bit Description**
11	RW	1	MAST106EN	**Master 106 Enable.** See MAST104EN for definition.
10:8	RW	0	MAST106DEST	**Master 106 Destination.** See MAST104DEST for definition.
7	RW	1	MAST105EN	**Master 105 Enable.** See MAST104EN for definition.
6:4	RW	0	MAST105DEST	**Master 105 Destination.** See MAST104DEST for definition.
3	RW	1	MAST104EN	**Master 104 Enable.** Enables tracing for Master 104. 0: tracing for Master 104 is disabled. All data received from Master 104 is received at the Input Buffer and immediately dropped. 1: tracing for Master 104 is enabled (Default)
2:0	RW	0	MAST104DEST	**Master 104 Destination.** Specifies the destination port number (refer to GTHOPT register for details) for Master 104 trace data. 0x0: Master 104 is routed to Output Port 0 0x1: Master 104 is routed to Output Port 1 ... 0x7: Master 104 is routed to Output Port 0x7 The default value of this bit field is controlled by the corresponding MDEST[N] parameter.

13.3.18 SWDEST_14: Switching Destination [14]

CSR Register Name: SWDEST_14: Switching Destination [14]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 40	**Offset End:** 43
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	1	MAST119EN	**Master 119 Enable.** See MAST112EN for definition.
30:28	RW	0	MAST119DEST	**Master 119 Destination.** See MAST112DEST for definition.
27	RW	1	MAST118EN	**Master 118 Enable.** See MAST112EN for definition.

<table>
<tr><td colspan="6"><b>CSR Register Name: SWDEST_14</b>: Switching Destination [14]</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 40</td><td><b>Offset End:</b> 43</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>26:24</td><td>RW</td><td>0</td><td>MAST118DEST</td><td colspan="2"><b>Master 118 Destination.</b> See MAST112DEST for definition.</td></tr>
<tr><td>23</td><td>RW</td><td>1</td><td>MAST117EN</td><td colspan="2"><b>Master 117 Enable.</b> See MAST112EN for definition.</td></tr>
<tr><td>22:20</td><td>RW</td><td>0</td><td>MAST117DEST</td><td colspan="2"><b>Master 117 Destination.</b> See MAST112DEST for definition.</td></tr>
<tr><td>19</td><td>RW</td><td>1</td><td>MAST116EN</td><td colspan="2"><b>Master 116 Enable.</b> See MAST112EN for definition.</td></tr>
<tr><td>18:16</td><td>RW</td><td>0</td><td>MAST116DEST</td><td colspan="2"><b>Master 116 Destination.</b> See MAST112DEST for definition.</td></tr>
<tr><td>15</td><td>RW</td><td>1</td><td>MAST115EN</td><td colspan="2"><b>Master 115 Enable.</b> See MAST112EN for definition.</td></tr>
<tr><td>14:12</td><td>RW</td><td>0</td><td>MAST115DEST</td><td colspan="2"><b>Master 115 Destination.</b> See MAST112DEST for definition.</td></tr>
<tr><td>11</td><td>RW</td><td>1</td><td>MAST114EN</td><td colspan="2"><b>Master 114 Enable.</b> See MAST112EN for definition.</td></tr>
<tr><td>10:8</td><td>RW</td><td>0</td><td>MAST114DEST</td><td colspan="2"><b>Master 114 Destination.</b> See MAST112DEST for definition.</td></tr>
<tr><td>7</td><td>RW</td><td>1</td><td>MAST113EN</td><td colspan="2"><b>Master 113 Enable.</b> See MAST112EN for definition.</td></tr>
<tr><td>6:4</td><td>RW</td><td>0</td><td>MAST113DEST</td><td colspan="2"><b>Master 113 Destination.</b> See MAST112DEST for definition.</td></tr>
<tr><td>3</td><td>RW</td><td>1</td><td>MAST112EN</td><td colspan="2"><b>Master 112 Enable.</b> Enables tracing for Master 112.
0: tracing for Master 112 is disabled. All data received from Master 112 is received at the Input Buffer and immediately dropped.
1: tracing for Master 112 is enabled (Default)</td></tr>
<tr><td>2:0</td><td>RW</td><td>0</td><td>MAST112DEST</td><td colspan="2"><b>Master 112 Destination.</b> Specifies the destination port number (refer to GTHOPT register for details) for Master 112 trace data.
0x0: Master 112 is routed to Output Port 0
0x1: Master 112 is routed to Output Port 1
...
0x7: Master 112 is routed to Output Port 0x7
The default value of this bit field is controlled by the corresponding MDEST[N] parameter.</td></tr>
</table>

13.3.19 SWDEST_15: Switching Destination [15]

CSR Register Name: SWDEST_15: Switching Destination [15]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 44	**Offset End:** 47
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	1	MAST127EN	**Master 127 Enable.** See MAST120EN for definition.
30:28	RW	0	MAST127DEST	**Master 127 Destination.** See MAST120DEST for definition.
27	RW	1	MAST126EN	**Master 126 Enable.** See MAST120EN for definition.
26:24	RW	0	MAST126DEST	**Master 126 Destination.** See MAST120DEST for definition.
23	RW	1	MAST125EN	**Master 125 Enable.** See MAST120EN for definition.
22:20	RW	0	MAST125DEST	**Master 125 Destination.** See MAST120DEST for definition.
19	RW	1	MAST124EN	**Master 124 Enable.** See MAST120EN for definition.
18:16	RW	0	MAST124DEST	**Master 124 Destination.** See MAST120DEST for definition.
15	RW	1	MAST123EN	**Master 123 Enable.** See MAST120EN for definition.
14:12	RW	0	MAST123DEST	**Master 123 Destination.** See MAST120DEST for definition.
11	RW	1	MAST122EN	**Master 122 Enable.** See MAST120EN for definition.
10:8	RW	0	MAST122DEST	**Master 122 Destination.** See MAST120DEST for definition.
7	RW	1	MAST121EN	**Master 121 Enable.** See MAST120EN for definition.
6:4	RW	0	MAST121DEST	**Master 121 Destination.** See MAST120DEST for definition.
3	RW	1	MAST120EN	**Master 120 Enable.** Enables tracing for Master 120. 0: tracing for Master 120 is disabled. All data received from Master 120 is received at the Input Buffer and immediately dropped. 1: tracing for Master 120 is enabled (Default)

CSR Register Name: SWDEST_15: Switching Destination [15]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 44	**Offset End:** 47
Bits	**Access**	**Default**	**Label**	**Bit Description**
2:0	RW	0	MAST120DEST	**Master 120 Destination.** Specifies the destination port number (refer to GTHOPT register for details) for Master 120 trace data. 0x0: Master 120 is routed to Output Port 0 0x1: Master 120 is routed to Output Port 1 ... 0x7: Master 120 is routed to Output Port 0x7 The default value of this bit field is controlled by the corresponding MDEST[N] parameter.

13.3.20 SWDEST_16: Switching Destination [16]

CSR Register Name: SWDEST_16: Switching Destination [16]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 48	**Offset End:** 4B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	1	MAST135EN	**Master 135 Enable.** See MAST128EN for definition.
30:28	RW	0	MAST135DEST	**Master 135 Destination.** See MAST128DEST for definition.
27	RW	1	MAST134EN	**Master 134 Enable.** See MAST128EN for definition.
26:24	RW	0	MAST134DEST	**Master 134 Destination.** See MAST128DEST for definition.
23	RW	1	MAST133EN	**Master 133 Enable.** See MAST128EN for definition.
22:20	RW	0	MAST133DEST	**Master 133 Destination.** See MAST128DEST for definition.
19	RW	1	MAST132EN	**Master 132 Enable.** See MAST128EN for definition.
18:16	RW	0	MAST132DEST	**Master 132 Destination.** See MAST128DEST for definition.
15	RW	1	MAST131EN	**Master 131 Enable.** See MAST128EN for definition.
14:12	RW	0	MAST131DEST	**Master 131 Destination.** See MAST128DEST for definition.

CSR Register Name: SWDEST_16: Switching Destination [16]				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 48	**Offset End:** 4B
Bits	**Access**	**Default**	**Label**	**Bit Description**
11	RW	1	MAST130EN	**Master 130 Enable.** See MAST128EN for definition.
10:8	RW	0	MAST130DEST	**Master 130 Destination.** See MAST128DEST for definition.
7	RW	1	MAST129EN	**Master 129 Enable.** See MAST128EN for definition.
6:4	RW	0	MAST129DEST	**Master 129 Destination.** See MAST128DEST for definition.
3	RW	1	MAST128EN	**Master 128 Enable.** Enables tracing for Master 128. 0: tracing for Master 128 is disabled. All data received from Master 128 is received at the Input Buffer and immediately dropped. 1: tracing for Master 128 is enabled (Default)
2:0	RW	0	MAST128DEST	**Master 128 Destination.** Specifies the destination port number (refer to GTHOPT register for details) for Master 128 trace data. 0x0: Master 128 is routed to Output Port 0 0x1: Master 128 is routed to Output Port 1 ... 0x7: Master 128 is routed to Output Port 0x7 The default value of this bit field is controlled by the corresponding MDEST[N] parameter.

13.3.21 SWDEST_17: Switching Destination [17]

CSR Register Name: SWDEST_17: Switching Destination [17]				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 4C	**Offset End:** 4F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	1	MAST143EN	**Master 143 Enable.** See MAST136EN for definition.
30:28	RW	0	MAST143DEST	**Master 143 Destination.** See MAST136DEST for definition.
27	RW	1	MAST142EN	**Master 142 Enable.** See MAST136EN for definition.

CSR Register Name: SWDEST_17: Switching Destination [17]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 4C **Offset End:** 4F
Bits	**Access**	**Default**	**Label**	**Bit Description**
26:24	RW	0	MAST142DEST	**Master 142 Destination.** See MAST136DEST for definition.
23	RW	1	MAST141EN	**Master 141 Enable.** See MAST136EN for definition.
22:20	RW	0	MAST141DEST	**Master 141 Destination.** See MAST136DEST for definition.
19	RW	1	MAST140EN	**Master 140 Enable.** See MAST136EN for definition.
18:16	RW	0	MAST140DEST	**Master 140 Destination.** See MAST136DEST for definition.
15	RW	1	MAST139EN	**Master 139 Enable.** See MAST136EN for definition.
14:12	RW	0	MAST139DEST	**Master 139 Destination.** See MAST136DEST for definition.
11	RW	1	MAST138EN	**Master 138 Enable.** See MAST136EN for definition.
10:8	RW	0	MAST138DEST	**Master 138 Destination.** See MAST136DEST for definition.
7	RW	1	MAST137EN	**Master 137 Enable.** See MAST136EN for definition.
6:4	RW	0	MAST137DEST	**Master 137 Destination.** See MAST136DEST for definition.
3	RW	1	MAST136EN	**Master 136 Enable.** Enables tracing for Master 136. 0: tracing for Master 136 is disabled. All data received from Master 136 is received at the Input Buffer and immediately dropped. 1: tracing for Master 136 is enabled (Default)
2:0	RW	0	MAST136DEST	**Master 136 Destination.** Specifies the destination port number (refer to GTHOPT register for details) for Master 136 trace data. 0x0: Master 136 is routed to Output Port 0 0x1: Master 136 is routed to Output Port 1 ... 0x7: Master 136 is routed to Output Port 0x7 The default value of this bit field is controlled by the corresponding MDEST[N] parameter.

13.3.22 SWDEST_18: Switching Destination [18]

CSR Register Name: **SWDEST_18**: Switching Destination [18]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 50	**Offset End:** 53
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	1	MAST151EN	**Master 151 Enable.** See MAST144EN for definition.
30:28	RW	0	MAST151DEST	**Master 151 Destination.** See MAST144DEST for definition.
27	RW	1	MAST150EN	**Master 150 Enable.** See MAST144EN for definition.
26:24	RW	0	MAST150DEST	**Master 150 Destination.** See MAST144DEST for definition.
23	RW	1	MAST149EN	**Master 149 Enable.** See MAST144EN for definition.
22:20	RW	0	MAST149DEST	**Master 149 Destination.** See MAST144DEST for definition.
19	RW	1	MAST148EN	**Master 148 Enable.** See MAST144EN for definition.
18:16	RW	0	MAST148DEST	**Master 148 Destination.** See MAST144DEST for definition.
15	RW	1	MAST147EN	**Master 147 Enable.** See MAST144EN for definition.
14:12	RW	0	MAST147DEST	**Master 147 Destination.** See MAST144DEST for definition.
11	RW	1	MAST146EN	**Master 146 Enable.** See MAST144EN for definition.
10:8	RW	0	MAST146DEST	**Master 146 Destination.** See MAST144DEST for definition.
7	RW	1	MAST145EN	**Master 145 Enable.** See MAST144EN for definition.
6:4	RW	0	MAST145DEST	**Master 145 Destination.** See MAST144DEST for definition.
3	RW	1	MAST144EN	**Master 144 Enable.** Enables tracing for Master 144. 0: tracing for Master 144 is disabled. All data received from Master 144 is received at the Input Buffer and immediately dropped. 1: tracing for Master 144 is enabled (Default)

CSR Register Name: SWDEST_18: Switching Destination [18]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 50 **Offset End:** 53
Bits	**Access**	**Default**	**Label**	**Bit Description**
2:0	RW	0	MAST144DEST	**Master 144 Destination.** Specifies the destination port number (refer to GTHOPT register for details) for Master 144 trace data. 0x0: Master 144 is routed to Output Port 0 0x1: Master 144 is routed to Output Port 1 … 0x7: Master 144 is routed to Output Port 0x7 The default value of this bit field is controlled by the corresponding MDEST[N] parameter.

13.3.23 SWDEST_19: Switching Destination [19]

CSR Register Name: SWDEST_19: Switching Destination [19]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 54 **Offset End:** 57
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	1	MAST159EN	**Master 159 Enable.** See MAST152EN for definition.
30:28	RW	0	MAST159DEST	**Master 159 Destination.** See MAST152DEST for definition.
27	RW	1	MAST158EN	**Master 158 Enable.** See MAST152EN for definition.
26:24	RW	0	MAST158DEST	**Master 158 Destination.** See MAST152DEST for definition.
23	RW	1	MAST157EN	**Master 157 Enable.** See MAST152EN for definition.
22:20	RW	0	MAST157DEST	**Master 157 Destination.** See MAST152DEST for definition.
19	RW	1	MAST156EN	**Master 156 Enable.** See MAST152EN for definition.
18:16	RW	0	MAST156DEST	**Master 156 Destination.** See MAST152DEST for definition.
15	RW	1	MAST155EN	**Master 155 Enable.** See MAST152EN for definition.
14:12	RW	0	MAST155DEST	**Master 155 Destination.** See MAST152DEST for definition.

<table>
<tr><td colspan="6"><b>CSR Register Name: SWDEST_19</b>: Switching Destination [19]</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 54</td><td><b>Offset End:</b> 57</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>11</td><td>RW</td><td>1</td><td>MAST154EN</td><td colspan="2"><b>Master 154 Enable.</b> See MAST152EN for definition.</td></tr>
<tr><td>10:8</td><td>RW</td><td>0</td><td>MAST154DEST</td><td colspan="2"><b>Master 154 Destination.</b> See MAST152DEST for definition.</td></tr>
<tr><td>7</td><td>RW</td><td>1</td><td>MAST153EN</td><td colspan="2"><b>Master 153 Enable.</b> See MAST152EN for definition.</td></tr>
<tr><td>6:4</td><td>RW</td><td>0</td><td>MAST153DEST</td><td colspan="2"><b>Master 153 Destination.</b> See MAST152DEST for definition.</td></tr>
<tr><td>3</td><td>RW</td><td>1</td><td>MAST152EN</td><td colspan="2"><b>Master 152 Enable.</b> Enables tracing for Master 152.
0: tracing for Master 152 is disabled. All data received from Master 152 is received at the Input Buffer and immediately dropped.
1: tracing for Master 152 is enabled (Default)</td></tr>
<tr><td>2:0</td><td>RW</td><td>0</td><td>MAST152DEST</td><td colspan="2"><b>Master 152 Destination.</b> Specifies the destination port number (refer to GTHOPT register for details) for Master 152 trace data.
0x0: Master 152 is routed to Output Port 0
0x1: Master 152 is routed to Output Port 1
...
0x7: Master 152 is routed to Output Port 0x7
The default value of this bit field is controlled by the corresponding MDEST[N] parameter.</td></tr>
</table>

13.3.24 SWDEST_20: Switching Destination [20]

<table>
<tr><td colspan="6"><b>CSR Register Name: SWDEST_20</b>: Switching Destination [20]</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 58</td><td><b>Offset End:</b> 5B</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31</td><td>RW</td><td>1</td><td>MAST167EN</td><td colspan="2"><b>Master 167 Enable.</b> See MAST160EN for definition.</td></tr>
<tr><td>30:28</td><td>RW</td><td>0</td><td>MAST167DEST</td><td colspan="2"><b>Master 167 Destination.</b> See MAST160DEST for definition.</td></tr>
<tr><td>27</td><td>RW</td><td>1</td><td>MAST166EN</td><td colspan="2"><b>Master 166 Enable.</b> See MAST160EN for definition.</td></tr>
</table>

<table>
<tr><td colspan="6"><b>CSR Register Name: SWDEST_20</b>: Switching Destination [20]</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 58</td><td><b>Offset End:</b> 5B</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>26:24</td><td>RW</td><td>0</td><td>MAST166DEST</td><td colspan="2"><b>Master 166 Destination.</b> See MAST160DEST for definition.</td></tr>
<tr><td>23</td><td>RW</td><td>1</td><td>MAST165EN</td><td colspan="2"><b>Master 165 Enable.</b> See MAST160EN for definition.</td></tr>
<tr><td>22:20</td><td>RW</td><td>0</td><td>MAST165DEST</td><td colspan="2"><b>Master 165 Destination.</b> See MAST160DEST for definition.</td></tr>
<tr><td>19</td><td>RW</td><td>1</td><td>MAST164EN</td><td colspan="2"><b>Master 164 Enable.</b> See MAST160EN for definition.</td></tr>
<tr><td>18:16</td><td>RW</td><td>0</td><td>MAST164DEST</td><td colspan="2"><b>Master 164 Destination.</b> See MAST160DEST for definition.</td></tr>
<tr><td>15</td><td>RW</td><td>1</td><td>MAST163EN</td><td colspan="2"><b>Master 163 Enable.</b> See MAST160EN for definition.</td></tr>
<tr><td>14:12</td><td>RW</td><td>0</td><td>MAST163DEST</td><td colspan="2"><b>Master 163 Destination.</b> See MAST160DEST for definition.</td></tr>
<tr><td>11</td><td>RW</td><td>1</td><td>MAST162EN</td><td colspan="2"><b>Master 162 Enable.</b> See MAST160EN for definition.</td></tr>
<tr><td>10:8</td><td>RW</td><td>0</td><td>MAST162DEST</td><td colspan="2"><b>Master 162 Destination.</b> See MAST160DEST for definition.</td></tr>
<tr><td>7</td><td>RW</td><td>1</td><td>MAST161EN</td><td colspan="2"><b>Master 161 Enable.</b> See MAST160EN for definition.</td></tr>
<tr><td>6:4</td><td>RW</td><td>0</td><td>MAST161DEST</td><td colspan="2"><b>Master 161 Destination.</b> See MAST160DEST for definition.</td></tr>
<tr><td>3</td><td>RW</td><td>1</td><td>MAST160EN</td><td colspan="2"><b>Master 160 Enable.</b> Enables tracing for Master 160.
0: tracing for Master 160 is disabled. All data received from Master 160 is received at the Input Buffer and immediately dropped.
1: tracing for Master 160 is enabled (Default)</td></tr>
<tr><td>2:0</td><td>RW</td><td>0</td><td>MAST160DEST</td><td colspan="2"><b>Master 160 Destination.</b> Specifies the destination port number (refer to GTHOPT register for details) for Master 160 trace data.
0x0: Master 160 is routed to Output Port 0
0x1: Master 160 is routed to Output Port 1
...
0x7: Master 160 is routed to Output Port 0x7
The default value of this bit field is controlled by the corresponding MDEST[N] parameter.</td></tr>
</table>

13.3.25 SWDEST_21: Switching Destination [21]

CSR Register Name: SWDEST_21: Switching Destination [21]				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 5C	**Offset End:** 5F

Bits	Access	Default	Label	Bit Description
31	RW	1	MAST175EN	**Master 175 Enable.** See MAST168EN for definition.
30:28	RW	0	MAST175DEST	**Master 175 Destination.** See MAST168DEST for definition.
27	RW	1	MAST174EN	**Master 174 Enable.** See MAST168EN for definition.
26:24	RW	0	MAST174DEST	**Master 174 Destination.** See MAST168DEST for definition.
23	RW	1	MAST173EN	**Master 173 Enable.** See MAST168EN for definition.
22:20	RW	0	MAST173DEST	**Master 173 Destination.** See MAST168DEST for definition.
19	RW	1	MAST172EN	**Master 172 Enable.** See MAST168EN for definition.
18:16	RW	0	MAST172DEST	**Master 172 Destination.** See MAST168DEST for definition.
15	RW	1	MAST171EN	**Master 171 Enable.** See MAST168EN for definition.
14:12	RW	0	MAST171DEST	**Master 171 Destination.** See MAST168DEST for definition.
11	RW	1	MAST170EN	**Master 170 Enable.** See MAST168EN for definition.
10:8	RW	0	MAST170DEST	**Master 170 Destination.** See MAST168DEST for definition.
7	RW	1	MAST169EN	**Master 169 Enable.** See MAST168EN for definition.
6:4	RW	0	MAST169DEST	**Master 169 Destination.** See MAST168DEST for definition.
3	RW	1	MAST168EN	**Master 168 Enable.** Enables tracing for Master 168. 0: tracing for Master 168 is disabled. All data received from Master 168 is received at the Input Buffer and immediately dropped. 1: tracing for Master 168 is enabled (Default)

CSR Register Name: SWDEST_21: Switching Destination [21]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 5C	**Offset End:** 5F
Bits	**Access**	**Default**	**Label**	**Bit Description**
2:0	RW	0	MAST168DEST	**Master 168 Destination.** Specifies the destination port number (refer to GTHOPT register for details) for Master 168 trace data. 0x0: Master 168 is routed to Output Port 0 0x1: Master 168 is routed to Output Port 1 ... 0x7: Master 168 is routed to Output Port 0x7 The default value of this bit field is controlled by the corresponding MDEST[N] parameter.

13.3.26 SWDEST_22: Switching Destination [22]

CSR Register Name: SWDEST_22: Switching Destination [22]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 60	**Offset End:** 63
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	1	MAST183EN	**Master 183 Enable.** See MAST176EN for definition.
30:28	RW	0	MAST183DEST	**Master 183 Destination.** See MAST176DEST for definition.
27	RW	1	MAST182EN	**Master 182 Enable.** See MAST176EN for definition.
26:24	RW	0	MAST182DEST	**Master 182 Destination.** See MAST176DEST for definition.
23	RW	1	MAST181EN	**Master 181 Enable.** See MAST176EN for definition.
22:20	RW	0	MAST181DEST	**Master 181 Destination.** See MAST176DEST for definition.
19	RW	1	MAST180EN	**Master 180 Enable.** See MAST176EN for definition.
18:16	RW	0	MAST180DEST	**Master 180 Destination.** See MAST176DEST for definition.
15	RW	1	MAST179EN	**Master 179 Enable.** See MAST176EN for definition.
14:12	RW	0	MAST179DEST	**Master 179 Destination.** See MAST176DEST for definition.

<table>
<tr><td colspan="6"><b>CSR Register Name: SWDEST_22</b>: Switching Destination [22]</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 60</td><td><b>Offset End:</b> 63</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>11</td><td>RW</td><td>1</td><td>MAST178EN</td><td colspan="2"><b>Master 178 Enable.</b> See MAST176EN for definition.</td></tr>
<tr><td>10:8</td><td>RW</td><td>0</td><td>MAST178DEST</td><td colspan="2"><b>Master 178 Destination.</b> See MAST176DEST for definition.</td></tr>
<tr><td>7</td><td>RW</td><td>1</td><td>MAST177EN</td><td colspan="2"><b>Master 177 Enable.</b> See MAST176EN for definition.</td></tr>
<tr><td>6:4</td><td>RW</td><td>0</td><td>MAST177DEST</td><td colspan="2"><b>Master 177 Destination.</b> See MAST176DEST for definition.</td></tr>
<tr><td>3</td><td>RW</td><td>1</td><td>MAST176EN</td><td colspan="2"><b>Master 176 Enable.</b> Enables tracing for Master 176.
0: tracing for Master 176 is disabled. All data received from Master 176 is received at the Input Buffer and immediately dropped.
1: tracing for Master 176 is enabled (Default)</td></tr>
<tr><td>2:0</td><td>RW</td><td>0</td><td>MAST176DEST</td><td colspan="2"><b>Master 176 Destination.</b> Specifies the destination port number (refer to GTHOPT register for details) for Master 176 trace data.
0x0: Master 176 is routed to Output Port 0
0x1: Master 176 is routed to Output Port 1
...
0x7: Master 176 is routed to Output Port 0x7
The default value of this bit field is controlled by the corresponding MDEST[N] parameter.</td></tr>
</table>

13.3.27 SWDEST_23: Switching Destination [23]

<table>
<tr><td colspan="6"><b>CSR Register Name: SWDEST_23</b>: Switching Destination [23]</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 64</td><td><b>Offset End:</b> 67</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31</td><td>RW</td><td>1</td><td>MAST191EN</td><td colspan="2"><b>Master 191 Enable.</b> See MAST184EN for definition.</td></tr>
<tr><td>30:28</td><td>RW</td><td>0</td><td>MAST191DEST</td><td colspan="2"><b>Master 191 Destination.</b> See MAST184DEST for definition.</td></tr>
<tr><td>27</td><td>RW</td><td>1</td><td>MAST190EN</td><td colspan="2"><b>Master 190 Enable.</b> See MAST184EN for definition.</td></tr>
</table>

CSR Register Name: **SWDEST_23**: Switching Destination [23]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 64	**Offset End:** 67
Bits	**Access**	**Default**	**Label**	**Bit Description**
26:24	RW	0	MAST190DEST	**Master 190 Destination.** See MAST184DEST for definition.
23	RW	1	MAST189EN	**Master 189 Enable.** See MAST184EN for definition.
22:20	RW	0	MAST189DEST	**Master 189 Destination.** See MAST184DEST for definition.
19	RW	1	MAST188EN	**Master 188 Enable.** See MAST184EN for definition.
18:16	RW	0	MAST188DEST	**Master 188 Destination.** See MAST184DEST for definition.
15	RW	1	MAST187EN	**Master 187 Enable.** See MAST184EN for definition.
14:12	RW	0	MAST187DEST	**Master 187 Destination.** See MAST184DEST for definition.
11	RW	1	MAST186EN	**Master 186 Enable.** See MAST184EN for definition.
10:8	RW	0	MAST186DEST	**Master 186 Destination.** See MAST184DEST for definition.
7	RW	1	MAST185EN	**Master 185 Enable.** See MAST184EN for definition.
6:4	RW	0	MAST185DEST	**Master 185 Destination.** See MAST184DEST for definition.
3	RW	1	MAST184EN	**Master 184 Enable.** Enables tracing for Master 184. 0: tracing for Master 184 is disabled. All data received from Master 184 is received at the Input Buffer and immediately dropped. 1: tracing for Master 184 is enabled (Default)
2:0	RW	0	MAST184DEST	**Master 184 Destination.** Specifies the destination port number (refer to GTHOPT register for details) for Master 184 trace data. 0x0: Master 184 is routed to Output Port 0 0x1: Master 184 is routed to Output Port 1 ... 0x7: Master 184 is routed to Output Port 0x7 The default value of this bit field is controlled by the corresponding MDEST[N] parameter.

13.3.28 SWDEST_24: Switching Destination [24]

CSR Register Name: SWDEST_24: Switching Destination [24]				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 68	**Offset End:** 6B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	1	MAST199EN	**Master 199 Enable.** See MAST192EN for definition.
30:28	RW	0	MAST199DEST	**Master 199 Destination.** See MAST192DEST for definition.
27	RW	1	MAST198EN	**Master 198 Enable.** See MAST192EN for definition.
26:24	RW	0	MAST198DEST	**Master 198 Destination.** See MAST192DEST for definition.
23	RW	1	MAST197EN	**Master 197 Enable.** See MAST192EN for definition.
22:20	RW	0	MAST197DEST	**Master 197 Destination.** See MAST192DEST for definition.
19	RW	1	MAST196EN	**Master 196 Enable.** See MAST192EN for definition.
18:16	RW	0	MAST196DEST	**Master 196 Destination.** See MAST192DEST for definition.
15	RW	1	MAST195EN	**Master 195 Enable.** See MAST192EN for definition.
14:12	RW	0	MAST195DEST	**Master 195 Destination.** See MAST192DEST for definition.
11	RW	1	MAST194EN	**Master 194 Enable.** See MAST192EN for definition.
10:8	RW	0	MAST194DEST	**Master 194 Destination.** See MAST192DEST for definition.
7	RW	1	MAST193EN	**Master 193 Enable.** See MAST192EN for definition.
6:4	RW	0	MAST193DEST	**Master 193 Destination.** See MAST192DEST for definition.
3	RW	1	MAST192EN	**Master 192 Enable.** Enables tracing for Master 192. 0: tracing for Master 192 is disabled. All data received from Master 192 is received at the Input Buffer and immediately dropped. 1: tracing for Master 192 is enabled (Default)

<table>
<tr><td colspan="6"><b>CSR Register Name: SWDEST_24</b>: Switching Destination [24]</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 68</td><td><b>Offset End:</b> 6B</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>2:0</td><td>RW</td><td>0</td><td>MAST192DEST</td><td colspan="2"><b>Master 192 Destination.</b> Specifies the destination port number (refer to GTHOPT register for details) for Master 192 trace data.
0x0: Master 192 is routed to Output Port 0
0x1: Master 192 is routed to Output Port 1
...
0x7: Master 192 is routed to Output Port 0x7
The default value of this bit field is controlled by the corresponding MDEST[N] parameter.</td></tr>
</table>

13.3.29 SWDEST_25: Switching Destination [25]

<table>
<tr><td colspan="6"><b>CSR Register Name: SWDEST_25</b>: Switching Destination [25]</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 6C</td><td><b>Offset End:</b> 6F</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31</td><td>RW</td><td>1</td><td>MAST207EN</td><td colspan="2"><b>Master 207 Enable.</b> See MAST200EN for definition.</td></tr>
<tr><td>30:28</td><td>RW</td><td>0</td><td>MAST207DEST</td><td colspan="2"><b>Master 207 Destination.</b> See MAST200DEST for definition.</td></tr>
<tr><td>27</td><td>RW</td><td>1</td><td>MAST206EN</td><td colspan="2"><b>Master 206 Enable.</b> See MAST200EN for definition.</td></tr>
<tr><td>26:24</td><td>RW</td><td>0</td><td>MAST206DEST</td><td colspan="2"><b>Master 206 Destination.</b> See MAST200DEST for definition.</td></tr>
<tr><td>23</td><td>RW</td><td>1</td><td>MAST205EN</td><td colspan="2"><b>Master 205 Enable.</b> See MAST200EN for definition.</td></tr>
<tr><td>22:20</td><td>RW</td><td>0</td><td>MAST205DEST</td><td colspan="2"><b>Master 205 Destination.</b> See MAST200DEST for definition.</td></tr>
<tr><td>19</td><td>RW</td><td>1</td><td>MAST204EN</td><td colspan="2"><b>Master 204 Enable.</b> See MAST200EN for definition.</td></tr>
<tr><td>18:16</td><td>RW</td><td>0</td><td>MAST204DEST</td><td colspan="2"><b>Master 204 Destination.</b> See MAST200DEST for definition.</td></tr>
<tr><td>15</td><td>RW</td><td>1</td><td>MAST203EN</td><td colspan="2"><b>Master 203 Enable.</b> See MAST200EN for definition.</td></tr>
<tr><td>14:12</td><td>RW</td><td>0</td><td>MAST203DEST</td><td colspan="2"><b>Master 203 Destination.</b> See MAST200DEST for definition.</td></tr>
</table>

<table>
<tr><td colspan="5"><b>CSR Register Name: SWDEST_25</b>: Switching Destination [25]</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 6C   <b>Offset End:</b> 6F</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th>Bit Description</th></tr>
<tr><td>11</td><td>RW</td><td>1</td><td>MAST202EN</td><td><b>Master 202 Enable.</b> See MAST200EN for definition.</td></tr>
<tr><td>10:8</td><td>RW</td><td>0</td><td>MAST202DEST</td><td><b>Master 202 Destination.</b> See MAST200DEST for definition.</td></tr>
<tr><td>7</td><td>RW</td><td>1</td><td>MAST201EN</td><td><b>Master 201 Enable.</b> See MAST200EN for definition.</td></tr>
<tr><td>6:4</td><td>RW</td><td>0</td><td>MAST201DEST</td><td><b>Master 201 Destination.</b> See MAST200DEST for definition.</td></tr>
<tr><td>3</td><td>RW</td><td>1</td><td>MAST200EN</td><td><b>Master 200 Enable.</b> Enables tracing for Master 200.
0: tracing for Master 200 is disabled. All data received from Master 200 is received at the Input Buffer and immediately dropped.
1: tracing for Master 200 is enabled (Default)</td></tr>
<tr><td>2:0</td><td>RW</td><td>0</td><td>MAST200DEST</td><td><b>Master 200 Destination.</b> Specifies the destination port number (refer to GTHOPT register for details) for Master 200 trace data.
0x0: Master 200 is routed to Output Port 0
0x1: Master 200 is routed to Output Port 1
...
0x7: Master 200 is routed to Output Port 0x7
The default value of this bit field is controlled by the corresponding MDEST[N] parameter.</td></tr>
</table>

13.3.30 SWDEST_26: Switching Destination [26]

<table>
<tr><td colspan="5"><b>CSR Register Name: SWDEST_26</b>: Switching Destination [26]</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 70   <b>Offset End:</b> 73</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th>Bit Description</th></tr>
<tr><td>31</td><td>RW</td><td>1</td><td>MAST215EN</td><td><b>Master 215 Enable.</b> See MAST208EN for definition.</td></tr>
<tr><td>30:28</td><td>RW</td><td>0</td><td>MAST215DEST</td><td><b>Master 215 Destination.</b> See MAST208DEST for definition.</td></tr>
<tr><td>27</td><td>RW</td><td>1</td><td>MAST214EN</td><td><b>Master 214 Enable.</b> See MAST208EN for definition.</td></tr>
</table>

CSR Register Name: SWDEST_26: Switching Destination [26]				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 70	**Offset End:** 73
Bits	**Access**	**Default**	**Label**	**Bit Description**
26:24	RW	0	MAST214DEST	**Master 214 Destination.** See MAST208DEST for definition.
23	RW	1	MAST213EN	**Master 213 Enable.** See MAST208EN for definition.
22:20	RW	0	MAST213DEST	**Master 213 Destination.** See MAST208DEST for definition.
19	RW	1	MAST212EN	**Master 212 Enable.** See MAST208EN for definition.
18:16	RW	0	MAST212DEST	**Master 212 Destination.** See MAST208DEST for definition.
15	RW	1	MAST211EN	**Master 211 Enable.** See MAST208EN for definition.
14:12	RW	0	MAST211DEST	**Master 211 Destination.** See MAST208DEST for definition.
11	RW	1	MAST210EN	**Master 210 Enable.** See MAST208EN for definition.
10:8	RW	0	MAST210DEST	**Master 210 Destination.** See MAST208DEST for definition.
7	RW	1	MAST209EN	**Master 209 Enable.** See MAST208EN for definition.
6:4	RW	0	MAST209DEST	**Master 209 Destination.** See MAST208DEST for definition.
3	RW	1	MAST208EN	**Master 208 Enable.** Enables tracing for Master 208. 0: tracing for Master 208 is disabled. All data received from Master 208 is received at the Input Buffer and immediately dropped. 1: tracing for Master 208 is enabled (Default)
2:0	RW	0	MAST208DEST	**Master 208 Destination.** Specifies the destination port number (refer to GTHOPT register for details) for Master 208 trace data. 0x0: Master 208 is routed to Output Port 0 0x1: Master 208 is routed to Output Port 1 ... 0x7: Master 208 is routed to Output Port 0x7 The default value of this bit field is controlled by the corresponding MDEST[N] parameter.

13.3.31 SWDEST_27: Switching Destination [27]

CSR Register Name: **SWDEST_27**: Switching Destination [27]				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 74	**Offset End:** 77

Bits	Access	Default	Label	Bit Description
31	RW	1	MAST223EN	**Master 223 Enable.** See MAST216EN for definition.
30:28	RW	0	MAST223DEST	**Master 223 Destination.** See MAST216DEST for definition.
27	RW	1	MAST222EN	**Master 222 Enable.** See MAST216EN for definition.
26:24	RW	0	MAST222DEST	**Master 222 Destination.** See MAST216DEST for definition.
23	RW	1	MAST221EN	**Master 221 Enable.** See MAST216EN for definition.
22:20	RW	0	MAST221DEST	**Master 221 Destination.** See MAST216DEST for definition.
19	RW	1	MAST220EN	**Master 220 Enable.** See MAST216EN for definition.
18:16	RW	0	MAST220DEST	**Master 220 Destination.** See MAST216DEST for definition.
15	RW	1	MAST219EN	**Master 219 Enable.** See MAST216EN for definition.
14:12	RW	0	MAST219DEST	**Master 219 Destination.** See MAST216DEST for definition.
11	RW	1	MAST218EN	**Master 218 Enable.** See MAST216EN for definition.
10:8	RW	0	MAST218DEST	**Master 218 Destination.** See MAST216DEST for definition.
7	RW	1	MAST217EN	**Master 217 Enable.** See MAST216EN for definition.
6:4	RW	0	MAST217DEST	**Master 217 Destination.** See MAST216DEST for definition.
3	RW	1	MAST216EN	**Master 216 Enable.** Enables tracing for Master 216. 0: tracing for Master 216 is disabled. All data received from Master 216 is received at the Input Buffer and immediately dropped. 1: tracing for Master 216 is enabled (Default)

CSR Register Name: **SWDEST_27**: Switching Destination [27]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 74	**Offset End:** 77
Bits	**Access**	**Default**	**Label**	**Bit Description**
2:0	RW	0	MAST216DEST	**Master 216 Destination.** Specifies the destination port number (refer to GTHOPT register for details) for Master 216 trace data. 0x0: Master 216 is routed to Output Port 0 0x1: Master 216 is routed to Output Port 1 ... 0x7: Master 216 is routed to Output Port 0x7 The default value of this bit field is controlled by the corresponding MDEST[N] parameter.

13.3.32 SWDEST_28: Switching Destination [28]

CSR Register Name: **SWDEST_28**: Switching Destination [28]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 78	**Offset End:** 7B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	1	MAST231EN	**Master 231 Enable.** See MAST224EN for definition.
30:28	RW	0	MAST231DEST	**Master 231 Destination.** See MAST224DEST for definition.
27	RW	1	MAST230EN	**Master 230 Enable.** See MAST224EN for definition.
26:24	RW	0	MAST230DEST	**Master 230 Destination.** See MAST224DEST for definition.
23	RW	1	MAST229EN	**Master 229 Enable.** See MAST224EN for definition.
22:20	RW	0	MAST229DEST	**Master 229 Destination.** See MAST224DEST for definition.
19	RW	1	MAST228EN	**Master 228 Enable.** See MAST224EN for definition.
18:16	RW	0	MAST228DEST	**Master 228 Destination.** See MAST224DEST for definition.
15	RW	1	MAST227EN	**Master 227 Enable.** See MAST224EN for definition.
14:12	RW	0	MAST227DEST	**Master 227 Destination.** See MAST224DEST for definition.

CSR Register Name: SWDEST_28: Switching Destination [28]					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 78	**Offset End:** 7B
Bits	**Access**	**Default**	**Label**	**Bit Description**	
11	RW	1	MAST226EN	**Master 226 Enable.** See MAST224EN for definition.	
10:8	RW	0	MAST226DEST	**Master 226 Destination.** See MAST224DEST for definition.	
7	RW	1	MAST225EN	**Master 225 Enable.** See MAST224EN for definition.	
6:4	RW	0	MAST225DEST	**Master 225 Destination.** See MAST224DEST for definition.	
3	RW	1	MAST224EN	**Master 224 Enable.** Enables tracing for Master 224. 0: tracing for Master 224 is disabled. All data received from Master 224 is received at the Input Buffer and immediately dropped. 1: tracing for Master 224 is enabled (Default)	
2:0	RW	0	MAST224DEST	**Master 224 Destination.** Specifies the destination port number (refer to GTHOPT register for details) for Master 224 trace data. 0x0: Master 224 is routed to Output Port 0 0x1: Master 224 is routed to Output Port 1 ... 0x7: Master 224 is routed to Output Port 0x7 The default value of this bit field is controlled by the corresponding MDEST[N] parameter.	

13.3.33 SWDEST_29: Switching Destination [29]

CSR Register Name: SWDEST_29: Switching Destination [29]					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 7C	**Offset End:** 7F
Bits	**Access**	**Default**	**Label**	**Bit Description**	
31	RW	1	MAST239EN	**Master 239 Enable.** See MAST232EN for definition.	
30:28	RW	0	MAST239DEST	**Master 239 Destination.** See MAST232DEST for definition.	
27	RW	1	MAST238EN	**Master 238 Enable.** See MAST232EN for definition.	

<table>
<tr><td colspan="6"><b>CSR Register Name: SWDEST_29</b>: Switching Destination [29]</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 7C</td><td><b>Offset End:</b> 7F</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>26:24</td><td>RW</td><td>0</td><td>MAST238DEST</td><td colspan="2"><b>Master 238 Destination.</b> See MAST232DEST for definition.</td></tr>
<tr><td>23</td><td>RW</td><td>1</td><td>MAST237EN</td><td colspan="2"><b>Master 237 Enable.</b> See MAST232EN for definition.</td></tr>
<tr><td>22:20</td><td>RW</td><td>0</td><td>MAST237DEST</td><td colspan="2"><b>Master 237 Destination.</b> See MAST232DEST for definition.</td></tr>
<tr><td>19</td><td>RW</td><td>1</td><td>MAST236EN</td><td colspan="2"><b>Master 236 Enable.</b> See MAST232EN for definition.</td></tr>
<tr><td>18:16</td><td>RW</td><td>0</td><td>MAST236DEST</td><td colspan="2"><b>Master 236 Destination.</b> See MAST232DEST for definition.</td></tr>
<tr><td>15</td><td>RW</td><td>1</td><td>MAST235EN</td><td colspan="2"><b>Master 235 Enable.</b> See MAST232EN for definition.</td></tr>
<tr><td>14:12</td><td>RW</td><td>0</td><td>MAST235DEST</td><td colspan="2"><b>Master 235 Destination.</b> See MAST232DEST for definition.</td></tr>
<tr><td>11</td><td>RW</td><td>1</td><td>MAST234EN</td><td colspan="2"><b>Master 234 Enable.</b> See MAST232EN for definition.</td></tr>
<tr><td>10:8</td><td>RW</td><td>0</td><td>MAST234DEST</td><td colspan="2"><b>Master 234 Destination.</b> See MAST232DEST for definition.</td></tr>
<tr><td>7</td><td>RW</td><td>1</td><td>MAST233EN</td><td colspan="2"><b>Master 233 Enable.</b> See MAST232EN for definition.</td></tr>
<tr><td>6:4</td><td>RW</td><td>0</td><td>MAST233DEST</td><td colspan="2"><b>Master 233 Destination.</b> See MAST232DEST for definition.</td></tr>
<tr><td>3</td><td>RW</td><td>1</td><td>MAST232EN</td><td colspan="2"><b>Master 232 Enable.</b> Enables tracing for Master 232.
0: tracing for Master 232 is disabled. All data received from Master 232 is received at the Input Buffer and immediately dropped.
1: tracing for Master 232 is enabled (Default)</td></tr>
<tr><td>2:0</td><td>RW</td><td>0</td><td>MAST232DEST</td><td colspan="2"><b>Master 232 Destination.</b> Specifies the destination port number (refer to GTHOPT register for details) for Master 232 trace data.
0x0: Master 232 is routed to Output Port 0
0x1: Master 232 is routed to Output Port 1
…
0x7: Master 232 is routed to Output Port 0x7
The default value of this bit field is controlled by the corresponding MDEST[N] parameter.</td></tr>
</table>

13.3.34 SWDEST_30: Switching Destination [30]

CSR Register Name: SWDEST_30: Switching Destination [30]				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 80	**Offset End:** 83
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	1	MAST247EN	**Master 247 Enable.** See MAST240EN for definition.
30:28	RW	0	MAST247DEST	**Master 247 Destination.** See MAST240DEST for definition.
27	RW	1	MAST246EN	**Master 246 Enable.** See MAST240EN for definition.
26:24	RW	0	MAST246DEST	**Master 246 Destination.** See MAST240DEST for definition.
23	RW	1	MAST245EN	**Master 245 Enable.** See MAST240EN for definition.
22:20	RW	0	MAST245DEST	**Master 245 Destination.** See MAST240DEST for definition.
19	RW	1	MAST244EN	**Master 244 Enable.** See MAST240EN for definition.
18:16	RW	0	MAST244DEST	**Master 244 Destination.** See MAST240DEST for definition.
15	RW	1	MAST243EN	**Master 243 Enable.** See MAST240EN for definition.
14:12	RW	0	MAST243DEST	**Master 243 Destination.** See MAST240DEST for definition.
11	RW	1	MAST242EN	**Master 242 Enable.** See MAST240EN for definition.
10:8	RW	0	MAST242DEST	**Master 242 Destination.** See MAST240DEST for definition.
7	RW	1	MAST241EN	**Master 241 Enable.** See MAST240EN for definition.
6:4	RW	0	MAST241DEST	**Master 241 Destination.** See MAST240DEST for definition.
3	RW	1	MAST240EN	**Master 240 Enable.** Enables tracing for Master 240. 0: tracing for Master 240 is disabled. All data received from Master 240 is received at the Input Buffer and immediately dropped. 1: tracing for Master 240 is enabled (Default)

CSR Register Name: SWDEST_30: Switching Destination [30]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 80 / **Offset End:** 83
Bits	**Access**	**Default**	**Label**	**Bit Description**
2:0	RW	0	MAST240DEST	**Master 240 Destination.** Specifies the destination port number (refer to GTHOPT register for details) for Master 240 trace data. 0x0: Master 240 is routed to Output Port 0 0x1: Master 240 is routed to Output Port 1 … 0x7: Master 240 is routed to Output Port 0x7 The default value of this bit field is controlled by the corresponding MDEST[N] parameter.

13.3.35 SWDEST_31: Switching Destination [31]

CSR Register Name: SWDEST_31: Switching Destination [31]				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 84 / **Offset End:** 87
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	1	MAST255EN	**Master 255 Enable.** See MAST248EN for definition.
30:28	RW	0	MAST255DEST	**Master 255 Destination.** See MAST248DEST for definition.
27	RW	1	MAST254EN	**Master 254 Enable.** See MAST248EN for definition.
26:24	RW	0	MAST254DEST	**Master 254 Destination.** See MAST248DEST for definition.
23	RW	1	MAST253EN	**Master 253 Enable.** See MAST248EN for definition.
22:20	RW	0	MAST253DEST	**Master 253 Destination.** See MAST248DEST for definition.
19	RW	1	MAST252EN	**Master 252 Enable.** See MAST248EN for definition.
18:16	RW	0	MAST252DEST	**Master 252 Destination.** See MAST248DEST for definition.
15	RW	1	MAST251EN	**Master 251 Enable.** See MAST248EN for definition.
14:12	RW	0	MAST251DEST	**Master 251 Destination.** See MAST248DEST for definition.

CSR Register Name: SWDEST_31: Switching Destination [31]				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 84	**Offset End:** 87
Bits	**Access**	**Default**	**Label**	**Bit Description**
11	RW	1	MAST250EN	**Master 250 Enable.** See MAST248EN for definition.
10:8	RW	0	MAST250DEST	**Master 250 Destination.** See MAST248DEST for definition.
7	RW	1	MAST249EN	**Master 249 Enable.** See MAST248EN for definition.
6:4	RW	0	MAST249DEST	**Master 249 Destination.** See MAST248DEST for definition.
3	RW	1	MAST248EN	**Master 248 Enable.** Enables tracing for Master 248. 0: tracing for Master 248 is disabled. All data received from Master 248 is received at the Input Buffer and immediately dropped. 1: tracing for Master 248 is enabled (Default)
2:0	RW	0	MAST248DEST	**Master 248 Destination.** Specifies the destination port number (refer to GTHOPT register for details) for Master 248 trace data. 0x0: Master 248 is routed to Output Port 0 0x1: Master 248 is routed to Output Port 1 ... 0x7: Master 248 is routed to Output Port 0x7 The default value of this bit field is controlled by the corresponding MDEST[N] parameter.

13.3.36 GSWDEST: General Software Trace Destination

CSR Register Name: GSWDEST: General Software Trace Destination				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 88	**Offset End:** 8B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:4	RO	0	Reserved	GSWDEST Reserved

<table>
<tr><td colspan="6"><b>CSR Register Name: GSWDEST</b>: General Software Trace Destination</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 88</td><td><b>Offset End:</b> 8B</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>3</td><td>RW</td><td>1</td><td>MAST256EN</td><td colspan="2"><b>Master 256 Enable.</b> Enables tracing for Master 256.
0: tracing for Master 256 is disabled. All data received from Master 256 is received at the Input Buffer and immediately dropped.
1: tracing for Master 256 is enabled (Default)</td></tr>
<tr><td>2:0</td><td>RW</td><td>0</td><td>MAST256DEST</td><td colspan="2"><b>Master 256 Destination.</b> Specifies the destination port number (refer to GTHOPT register for details) for Master 256 trace data.
0x0: Master 256 is routed to Output Port 0
0x1: Master 256 is routed to Output Port 1
...
0x7: Master 256 is routed to Output Port 0x7
The default value of this bit field is controlled by the corresponding MDEST[N] parameter.</td></tr>
</table>

13.3.37 LWM0: Low WaterMark for Sources 0-7

<table>
<tr><td colspan="6"><b>CSR Register Name: LWM0</b>: Low WaterMark for Sources 0-7</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 8C</td><td><b>Offset End:</b> 8F</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:28</td><td>RW</td><td>0</td><td>LWM7</td><td colspan="2">Low Water Mark for Trace Source #7.</td></tr>
<tr><td>27:24</td><td>RW</td><td>0</td><td>LWM6</td><td colspan="2">Low Water Mark for Trace Source #6.</td></tr>
<tr><td>23:20</td><td>RW</td><td>0</td><td>LWM5</td><td colspan="2">Low Water Mark for Trace Source #5.</td></tr>
<tr><td>19:16</td><td>RW</td><td>0</td><td>LWM4</td><td colspan="2">Low Water Mark for Trace Source #4.</td></tr>
<tr><td>15:12</td><td>RW</td><td>0</td><td>LWM3</td><td colspan="2">Low Water Mark for Trace Source #3.</td></tr>
<tr><td>11:8</td><td>RW</td><td>0</td><td>LWM2</td><td colspan="2">Low Water Mark for Trace Source #2.</td></tr>
<tr><td>7:4</td><td>RW</td><td>0</td><td>LWM1</td><td colspan="2">Low Water Mark for Trace Source #1.</td></tr>
</table>

CSR Register Name: LWM0: Low WaterMark for Sources 0-7					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 8C	**Offset End:** 8F
Bits	**Access**	**Default**	**Label**	**Bit Description**	
3:0	RW	0	LWM0	**Low Water Mark for Trace Source #0.** Indicates the maximum level to which the input buffer for trace source #0 can be filled before the lowWaterMark signal will be de-asserted. That is, the lowWaterMark signal is asserted when the input buffer has LWM0 entries or less. Conversely, when greater than LWM0 entries are filled, lowWaterMark is deasserted. With a default value of 0, the lowWaterMark signal serves as an empty signal. In practice the LWM value for a trace source should be set to the grant duration minus 1 (GrantDur-1).	

13.3.38 LWM1: Low WaterMark for Sources 7-15

CSR Register Name: LWM1: Low WaterMark for Sources 7-15					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 90	**Offset End:** 93
Bits	**Access**	**Default**	**Label**	**Bit Description**	
31:28	RW	0	LWM15	Low Water Mark for Trace Source #15.	
27:24	RW	0	LWM14	Low Water Mark for Trace Source #14.	
23:20	RW	0	LWM13	Low Water Mark for Trace Source #13.	
19:16	RW	0	LWM12	Low Water Mark for Trace Source #12.	
15:12	RW	0	LWM11	Low Water Mark for Trace Source #11.	
11:8	RW	0	LWM10	Low Water Mark for Trace Source #10.	
7:4	RW	0	LWM9	Low Water Mark for Trace Source #9.	
3:0	RW	0	LWM8	Low Water Mark for Trace Source #8.	

13.3.39 GTH_INFO_1: GTH Parameter Info 1

CSR Register Name: GTH_INFO_1: GTH Parameter Info 1					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 94	**Offset End:** 97

Bits	Access	Default	Label	Bit Description
3:0	RO	4	CTS_SIGNAL_ WIDTH	**CTS External Signal Width.** Indicates the number of external (external to NPK, not the SoC) signals available to the SOC for its own use (implementation specific use).

13.3.40 GTH_MISC: GTH Miscellaneous Register

CSR Register Name: GTH_MISC: GTH Miscellaneous Register				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 98	**Offset End:** 9B
Bits	**Access**	**Default**	**Label**	**Bit Description**
0	RW	0	RSVD	reserved

13.3.41 SMCR0: STP Maintenance Control Register 0

CSR Register Name: SMCR0: STP Maintenance Control Register 0				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 9C	**Offset End:** 9F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:16	RW	8000	SYNCF1	**Sync Packet Frequency for Port 1.** Specifies the number data sets between Maintenance packets for port 1. A value of 0 (zero) turns off Maintenance packet generation for the port. However, an initial maintenance packet will be sent at the start of the trace so that reconstruction software can successfully reconstruct the trace data.
15:0	RW	8000	SYNCF0	**Sync Packet Frequency for Port 0.** Specifies the number of data sets between Maintenance packets for port 0. A value of 0 (zero) turns off Maintenance packet generation for the port. However, an initial maintenance packet will be sent at the start of the trace so that reconstruction software can successfully reconstruct the trace data.

13.3.42 SMCR1: STP Maintenance Control Register 1

CSR Register Name: SMCR1: STP Maintenance Control Register 1				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** A0 **Offset End:** A3
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:16	RW	8000	SYNCF3	**Sync Packet Frequency for Port 3.** Specifies the number of data sets between Maintenance packets for port 3. A value of 0 (zero) turns off Maintenance packet generation for the port. However, an initial maintenance packet will be sent at the start of the trace so that reconstruction software can successfully reconstruct the trace data.
15:0	RW	8000	SYNCF2	**Sync Packet Frequency for Port 2.** Specifies the number of data sets between Maintenance packets for port 2. A value of 0 (zero) turns off Maintenance packet generation for the port. However, an initial maintenance packet will be sent at the start of the trace so that reconstruction software can successfully reconstruct the trace data.

13.3.43 SMCR2: STP Maintenance Control Register 2

CSR Register Name: SMCR2: STP Maintenance Control Register 2				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** A4 **Offset End:** A7
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:16	RW	8000	SYNCF5	**Sync Packet Frequency for Port 5.** Specifies the number of data sets between Maintenence packets for port 5. A value of 0 (zero) turns off Maintenance packet generation for the port. However, an initial maintenance packet will be sent at the start of the trace so that reconstruction software can successfully reconstruct the trace data.
15:0	RW	8000	SYNCF4	**Sync Packet Frequency for Port 4.** Specifies the number of data sets between Maintenance packets for port 4. A value of 0 (zero) turns off Maintenance packet generation for the port. However, an initial maintenance packet will be sent at the start of the trace so that reconstruction software can successfully reconstruct the trace data.

13.3.44 SMCR3: STP Maintenance Control Register 3

CSR Register Name: SMCR3: STP Maintenance Control Register 3				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** A8 **Offset End:** AB
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:16	RW	8000	SYNCF7	**Sync Packet Frequency for Port 7.** Specifies the number of data sets between Maintenance packets for port 5. A value of 0 (zero) turns off Maintenance packet generation for the port. However, an initial maintenance packet will be sent at the start of the trace so that reconstruction software can successfully reconstruct the trace data.
15:0	RW	8000	SYNCF6	**Sync Packet Frequency for Port 6.** Specifies the number of data sets between Maintenance packets for port 4. A value of 0 (zero) turns off Maintenance packet generation for the port. However, an initial maintenance packet will be sent at the start of the trace so that reconstruction software can successfully reconstruct the trace data.

13.3.45 PGD0: Programmable Grant Duration Register 0

CSR Register Name: PGD0: Programmable Grant Duration Register 0				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** AC **Offset End:** AF
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:28	RW	1	GNTDUR_7	**Programmable Grant Duration for Input Port 7.** See definition for GNTDUR[0] for details. These bits are read-only as 0h if NUMTSOURCE is less than 8
27:24	RW	1	GNTDUR_6	**Programmable Grant Duration for Input Port 6.** See definition for GNTDUR[0] for details. These bits are read-only as 0h if NUMTSOURCE is less than 7

CSR Register Name: PGD0: Programmable Grant Duration Register 0				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** AC **Offset End:** AF
Bits	**Access**	**Default**	**Label**	**Bit Description**
23:20	RW	1	GNTDUR_5	**Programmable Grant Duration for Input Port 5.** See definition for GNTDUR[0] for details. These bits are read-only as 0h if NUMTSOURCE is less than 6
19:16	RW	1	GNTDUR_4	**Programmable Grant Duration for Input Port 4.** See definition for GNTDUR[0] for details. These bits are read-only as 0h if NUMTSOURCE is less than 5
15:12	RW	1	GNTDUR_3	**Programmable Grant Duration for Input Port 3.** See definition for GNTDUR[0] for details. These bits are read-only as 0h if NUMTSOURCE is less than 4
11:8	RW	1	GNTDUR_2	**Programmable Grant Duration for Input Port 2.** See definition for GNTDUR[0] for details. These bits are read-only as 0h if NUMTSOURCE is less than 3
7:4	RW	1	GNTDUR_1	**Programmable Grant Duration for Input Port 1.** See definition for GNTDUR[0] for details. These bits are read-only as 0h if NUMTSOURCE is less than 2
3:0	RW	1	GNTDUR_0	**Programmable Grant Duration for Input Port 0.** See definition for GNTDUR[0] for details. These bits are read-only as 0h if NUMTSOURCE is less than 1

13.3.46 PGD1: Programmable Grant Duration Register 1

CSR Register Name: PGD1: Programmable Grant Duration Register 1				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** B0 **Offset End:** B3
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:28	RW	1	GNTDUR_15	**Programmable Grant Duration for Input Port 15.** See definition for GNTDUR[0] for details. These bits are read-only as 0h if NUMTSOURCE is less than 16.

CSR Register Name: **PGD1**: Programmable Grant Duration Register 1				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** B0 / **Offset End:** B3
Bits	**Access**	**Default**	**Label**	**Bit Description**
27:24	RW	1	GNTDUR_14	**Programmable Grant Duration for Input Port 14.** See definition for GNTDUR[0] for details. These bits are read-only as 0h if NUMTSOURCE is less than 15.
23:20	RW	1	GNTDUR_13	**Programmable Grant Duration for Input Port 13.** See definition for GNTDUR[0] for details. These bits are read-only as 0h if NUMTSOURCE is less than 14.
19:16	RW	1	GNTDUR_12	**Programmable Grant Duration for Input Port 12.** See definition for GNTDUR[0] for details. These bits are read-only as 0h if NUMTSOURCE is less than 13.
15:12	RW	1	GNTDUR_11	**Programmable Grant Duration for Input Port 11.** See definition for GNTDUR[0] for details. These bits are read-only as 0h if NUMTSOURCE is less than 12.
11:8	RW	1	GNTDUR_10	**Programmable Grant Duration for Input Port 10.** See definition for GNTDUR[0] for details. These bits are read-only as 0h if NUMTSOURCE is less than 11.
7:4	RW	1	GNTDUR_9	**Programmable Grant Duration for Input Port 9.** See definition for GNTDUR[0] for details. These bits are read-only as 0h if NUMTSOURCE is less than 10.
3:0	RW	1	GNTDUR_8	**Programmable Grant Duration for Input Port 8.** See definition for GNTDUR[0] for details. These bits are read-only as 0h if NUMTSOURCE is less than 9.

13.3.47 SCR: Source Control Register

CSR Register Name: **SCR**: Source Control Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** C8 / **Offset End:** CB
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:24	RO	0	Reserved	SCR Reserved

CSR Register Name: SCR: Source Control Register				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** C8	**Offset End:** CB
Bits	**Access**	**Default**	**Label**	**Bit Description**
23	RW	0	StoreEnOvrd7	Storage Enable Override 7.
22	RW	0	StoreEnOvrd6	Storage Enable Override 6.
21	RW	0	StoreEnOvrd5	Storage Enable Override 5.
20	RW	0	StoreEnOvrd4	Storage Enable Override 4.
19	RW	0	StoreEnOvrd3	Storage Enable Override 3.
18	RW	0	StoreEnOvrd2	**Storage Enable Override 2.** Under normal operation the CTS gasket controls the StoreEn[N:0] signals to the trace sources. This Storage Enable Override configuration bit is a method to force the StoreEn[N:0] signals without requiring programming of the CTS or the gasket. The StoreEnOvrd is logically OR d together with the StoreEn from the gasket before being sent to the trace sources.
17:8	RO	0	Reserved	SCR Reserved
7	RW	0	STM7	Storage Mode control bit for Source 7.
6	RW	0	STM6	Storage Mode control bit for Source 6.
5	RW	0	STM5	Storage Mode control bit for Source 5.
4	RW	0	STM4	Storage Mode control bit for Source 4.
3	RW	0	STM3	Storage Mode control bit for Source 3.
2	RW	0	STM2	Storage Mode control bit for Source 2.
1	RW	0	STM1	Storage Mode control bit for Source 1.
0	RW	0	STM0	**Storage Mode control bit for Source 0.** 0: Normal mode. Trace source is stored in single logical buffer. 1: Multi-buffer. Trace source is stored in separate logical buffers with wrapping around trigger assertion

13.3.48 GTHFRQ: GTH Frequency Register

CSR Register Name: GTHFRQ: GTH Frequency Register			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** CC	**Offset End:** CF

Bits	Access	Default	Label	Bit Description
31:0	RW	varies	GTH_FREQ	**GTH Operating Frequency.** Specifies the operating frequency of the GTH, in Hertz.

13.3.49 GTHRSVD: GTH Reserved

CSR Register Name: GTHRSVD: GTH Reserved				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** D0	**Offset End:** D3
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:28	RW	0	RSVD	Reserved for future use
27:24	RW	0	RSVD	Reserved for future use
23:20	RW	0	RSVD	Reserved for future use
19:16	RW	0	RSVD	Reserved for future use
15:12	RW	0	RSVD	Reserved for future use
11:8	RW	1	RSVD	Reserved for future use
7:4	RW	1	RSVD	Reserved for future use
3:0	RW	1	RSVD	Reserved for future use

13.3.50 GTHSTAT: GTH Status

CSR Register Name: GTHSTAT: GTH Status				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** D4	**Offset End:** D7
Bits	**Access**	**Default**	**Label**	**Bit Description**
7	RO	0	PLE7	Pipe Line Empty, Output port 7.
6	RO	0	PLE6	Pipe Line Empty, Output port 6.
5	RO	0	PLE5	Pipe Line Empty, Output port 5.
4	RO	0	PLE4	Pipe Line Empty, Output port 4.
3	RO	0	PLE3	Pipe Line Empty, Output port 3.
2	RO	0	PLE2	Pipe Line Empty, Output port 2.
1	RO	0	PLE1	Pipe Line Empty, Output port 1.

<table>
<tr><td colspan="6"><b>CSR Register Name:</b> <b>GTHSTAT</b>: GTH Status</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> D4</td><td><b>Offset End:</b> D7</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>0</td><td>RO</td><td>0</td><td>PLE0</td><td colspan="2"><b>Pipe Line Empty, Output port 0.</b> Indicates that the data pipeline through GTH, from Input Buffer to Byte Packing Buffer, is empty. Software can read this bit to determine if the pipeline is empty. Along with an equivalent bit from the output port controller (e.g., MSU, PTI), software can determine if all the data has been sent to the destination, and processing of the data can begin.</td></tr>
</table>

13.3.51 SCR2: Source Control Register 2

<table>
<tr><td colspan="6"><b>CSR Register Name:</b> <b>SCR2</b>: Source Control Register 2</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> D8</td><td><b>Offset End:</b> DB</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:8</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">SCR2 Reserved</td></tr>
<tr><td>7</td><td>RW</td><td>0</td><td>FSEOFF7</td><td colspan="2">Force StoreEn Off for Trace Source 7</td></tr>
<tr><td>6</td><td>RW</td><td>0</td><td>FSEOFF6</td><td colspan="2">Force StoreEn Off for Trace Source 6</td></tr>
<tr><td>5</td><td>RW</td><td>0</td><td>FSEOFF5</td><td colspan="2">Force StoreEn Off for Trace Source 5</td></tr>
<tr><td>4</td><td>RW</td><td>0</td><td>FSEOFF4</td><td colspan="2">Force StoreEn Off for Trace Source 4</td></tr>
<tr><td>3</td><td>RW</td><td>0</td><td>FSEOFF3</td><td colspan="2">Force StoreEn Off for Trace Source 3</td></tr>
<tr><td>2</td><td>RW</td><td>0</td><td>FSEOFF2</td><td colspan="2"><b>Force StoreEn Off for Trace Source 2.</b> Overrides StoreEn[N] signal to force it off. This will override the StoreEnOvrd setting, forcing the storeEn to off.</td></tr>
<tr><td>1</td><td>RO</td><td>0</td><td>Rsvd</td><td colspan="2">SCR2 Reserved</td></tr>
<tr><td>0</td><td>RW</td><td>0</td><td>FCD</td><td colspan="2"><b>Force Capture Done.</b> When set, this bit will force the captureDone signal going from CTS to The CTS Gasket to be asserted. Note: when an MSC is in multi-block mode, this will cause the MSU to switch to the next memory window.</td></tr>
</table>

13.3.52 DESTOVR: Destination Override Register

CSR Register Name: DESTOVR: Destination Override Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** DC	**Offset End:** DF
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:4	RO	0	Rsvd	DESTOVR Reserved
3	RW	0	DESTOVREN	**Destination Override Enable.** Setting this bit activates the Destination Override function. All trace data from all masters will be sent to the output port specified by bits 2:0.
2:0	RW	0	DESTOVR	**Destination Override.** Overrides all Mast[N]Dest bits in the SWDEST and GSWDEST registers with the value specified in this register field. This has the net effect of sending all trace data to the port specified by these bits. These bits use the same encoding as MAST[N]DEST bits in SWDEST register.

13.3.53 SCRPD0: ScratchPad 0 Register

CSR Register Name: SCRPD0: ScratchPad 0 Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** E0	**Offset End:** E3
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RW	0	scpd0	Scratch Pad bits, for use by software/firmware.

13.3.54 SCRPD1: ScratchPad 1 Register

CSR Register Name: SCRPD1: ScratchPad 1 Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** E4	**Offset End:** E7
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RW	0	scpd1	Scratch Pad bits, for use by software/firmware.

13.3.55 SCRPD2: ScratchPad 2 Register

CSR Register Name: SCRPD2: ScratchPad 2 Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** E8	**Offset End:** EB
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RW	0	scpd2	Scratch Pad bits, for use by software/firmware.

13.3.56 SCRPD3: ScratchPad 3 Register

CSR Register Name: SCRPD3: ScratchPad 3 Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** EC	**Offset End:** EF
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RW	0	scpd3	Scratch Pad bits, for use by software/firmware.

13.4 TSCU Registers

13.4.1 TSCU Registers Summary

TSCU Registers			
Offset Start	**Offset End**	**Symbol**	**Register Name/Function**
2000	2007	TSUCTRL	TSCU Control Register
2004	2007	TSCUSTAT	TSCU Control Register
2004	2007	TSCUSTAT	TSCU Status Register
2008	200F	CTCSSL	CTC Snapshot Lower Register
2010	2013	NPKTSCSSL	NPKTSC Snapshot Lower Register
2014	2017	NPKTSCSSU	NPKTSC Snapshot Upper Register
2100	2103	NPKTSCINCVAL0	NPKTSC Increment Value 0 Register
2104	2107	NPKTSCINCVAL1	NPKTSC Increment Value 1 Register
2108	210B	NPKTSCINCVAL2	NPKTSC Increment

TSCU Registers			
Offset Start	**Offset End**	**Symbol**	**Register Name/Function**
			Value 2 Register
210C	210F	NPKTSCINCVAL3	NPKTSC Increment Value 3 Register
2110	2113	NPKTSCINCVAL4	NPKTSC Increment Value 4 Register

13.4.2 TSUCTRL: TSCU Control Register

CSR Register Name: TSUCTRL: TSCU Control Register

Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 2000	**Offset End:** 2007

Bits	Access	Default	Label	Bit Description
31:22	RO	0	Reserved	TSUCTRL Reserved
21:16	RW/P	0	LastIncVal	Last Increment Value. Points to the last IncValX field of the corresponding NPKTSCINCVALY register required in the NPKTSC/NPKOffset counter scaling logic. Logic iterates from NPKTSCINCVAL0.IncVal0 through the field specified by LastIncVal, one value per NPCLK, and then wraps back around to begin again at NPKTSCINCVAL0.IncVal0. The currently selected IncVal in a given NPCLK cycle is added to NPKTSC/NPKOffset counters when pllclksel is 0. Results in scaling the NPKTSC/NPKOffset counters to a consistent timebase in terms of the PLL clock frequency when the slower clock is selected. 0: NPKTSCINCVAL0.IncVal0 1: NPKTSCINCVAL0.IncVal1 … 7: NPKTSCINCVAL0.IncVal7 8: NPKTSCINCVAL1.IncVal0 … 39: NPKTSCINCVAL4.IncVal7 40-63: Reserved
15:14	RO	0	Reserved	TSUCTRL Reserved

CSR Register Name: TSUCTRL: TSCU Control Register				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 2000	**Offset End:** 2007
Bits	**Access**	**Default**	**Label**	**Bit Description**
13:8	RW	38	SampleRateSel	**Sample Rate Select.** Selects the periodic rate of correlation packet generation and the selected value corresponds to a single CTC bit. A correlation packet is created whenever the selected CTC bit changes value (both 0->1 and 1->0 transitions) 6b000000: CTC[0] 6b000001: CTC[1] 6b000010: CTC[2] 6b000011: CTC[3] ... 6b110111: CTC[55] 6b111XXX: Disable periodic sampling

7:4	RW	1	MaxAllowedDelay	MaxAllowedDelay. Specifies the maximum allowed round trip time of the non-posted LocalStamp message when the sync command is bounded. The agent can choose one of three bound values pre-defined by the corresponding ARU register. The agent must choose a value which is longer than the minimum round trip delay of a non-posted message from the ARU to the target agent. Encoding:
				0000 => Bound Range Low;
				0001 => Bound Range 2;
				0010 => Bound Range Max;
				0011-1111 => Reserved

3	RO	0	Reserved	TSUCTRL Reserved
2	RW	0	SnapShotTSC	**Snapshot Timestamp Counter.** Writing 0x1 causes the current value of both CTC and NPKTSC to be written to their respective snapshot registers so that they can be safely read by software. This bit is automatically cleared one North Peak clock after being written to remove the need for a separate write to clear it before getting subsequent snapshot updates. Writing zero has no effect. Reads will always return 0 for this field.
1	RW/P	0	SwFrcSample	**Software Force Sample.** Writing 0x1 forces a correlation packet with the current CTC, NPKOffset, and NPKTSC values to be sent to GTH. This bit is automatically cleared one North Peak clock after being written. Writing zero has no effect. Reads will always return 0 for this field.

<table>
<tr><td colspan="6"><b>CSR Register Name: TSUCTRL</b>: TSCU Control Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 2000</td><td><b>Offset End:</b> 2007</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>0</td><td>RW/P</td><td>0</td><td>CTCResync</td><td colspan="2"><b>CTC Resync.</b> When set, forces the CTC control state machine to go to the INIT state. When cleared, allows the CTC control state machine to begin the CTC initialization process if the ctc_enable signal is high and if the XCLK is running.</td></tr>
</table>

13.4.3 TSCUSTAT: TSCU Control Register

<table>
<tr><td colspan="6"><b>CSR Register Name: TSCUSTAT</b>: TSCU Control Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 2004</td><td><b>Offset End:</b> 2007</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>10:8</td><td>RO</td><td>0</td><td>CTCCSMstat</td><td colspan="2"><b>CTC Control State Machine Status.</b> The current state of the CTC control state machine.
000: INIT
001: IRDY
010: WAIT
011: ARMED
100: SYNC
101: ERROR
110: N/A
111: N/A</td></tr>
</table>

13.4.4 TSCUSTAT: TSCU Status Register

<table>
<tr><td colspan="6"><b>CSR Register Name: TSCUSTAT</b>: TSCU Status Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 2004</td><td><b>Offset End:</b> 2007</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>31:11</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">TSCUSTAT Reserved</td></tr>
</table>

13.4.5 CTCSSL: CTC Snapshot Lower Register

<table>
<tr><td colspan="6"><b>CSR Register Name: CTCSSL</b>: CTC Snapshot Lower Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 2008</td><td><b>Offset End:</b> 200F</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:0</td><td>RW/1C/P/V</td><td>0</td><td>CTCSnapShotL</td><td colspan="2"><b>CTC Snapshot Lower.</b> Lower DW of the CTC snapshot value</td></tr>
<tr><td>31:0</td><td>RW/1C/P/V</td><td>0</td><td>CTCSnapShotU</td><td colspan="2"><b>CTC Snapshot Upper.</b> Upper 24 bits of the CTC snapshot value</td></tr>
</table>

13.4.6 NPKTSCSSL: NPKTSC Snapshot Lower Register

<table>
<tr><td colspan="6"><b>CSR Register Name: NPKTSCSSL</b>: NPKTSC Snapshot Lower Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 2010</td><td><b>Offset End:</b> 2013</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:0</td><td>RW/1C/P/V</td><td>0</td><td>NPKTSCSnapShotL</td><td colspan="2"><b>NPKTSC Snapshot Lower.</b> Lower DW of the NPKTSC snapshot value</td></tr>
</table>

13.4.7 NPKTSCSSU: NPKTSC Snapshot Upper Register

<table>
<tr><td colspan="6"><b>CSR Register Name: NPKTSCSSU</b>: NPKTSC Snapshot Upper Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 2014</td><td><b>Offset End:</b> 2017</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:0</td><td>RW/1C/P/V</td><td>0</td><td>NPKTSCSnapShotU</td><td colspan="2"><b>NPKTSC Snapshot Upper.</b> Upper DW of the NPKTSC snapshot value</td></tr>
</table>

13.4.8 NPKTSCINCVAL0: NPKTSC Increment Value 0 Register

<table>
<tr><td colspan="6"><b>CSR Register Name: NPKTSCINCVAL0</b>: NPKTSC Increment Value 0 Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 2100</td><td><b>Offset End:</b> 2103</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:28</td><td>RW</td><td>1</td><td>IncVal7</td><td colspan="2">Increment Value 7.</td></tr>
</table>

CSR Register Name: **NPKTSCINCVAL0**: NPKTSC Increment Value 0 Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 2100	**Offset End:** 2103
Bits	**Access**	**Default**	**Label**	**Bit Description**
27:24	RW	1	IncVal6	Increment Value 6.
23:20	RW	1	IncVal5	Increment Value 5.
19:16	RW	1	IncVal4	Increment Value 4.
15:12	RW	1	IncVal3	Increment Value 3.
11:8	RW	1	IncVal2	Increment Value 2.
7:4	RW	1	IncVal1	Increment Value 1.
3:0	RW	1	IncVal0	**Increment Value 0.** Specifies the increment value for the first NPCLK cycle within the series of increment values up through the value indicated by TSCUCTRL.LastIncVal. This value is added to the NPKTSC/NPKOffset counters in the first phase of the iteration through all required increment values specified in the other IncValN fields. Values should always be >= 1 so that the affected counters will always increment on each NPCLK cycle.

13.4.9 NPKTSCINCVAL1: NPKTSC Increment Value 1 Register

CSR Register Name: **NPKTSCINCVAL1**: NPKTSC Increment Value 1 Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 2104	**Offset End:** 2107
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:28	RW	1	IncVal15	Increment Value 15.
27:24	RW	1	IncVal14	Increment Value 14.
23:20	RW	1	IncVal13	Increment Value 13.
19:16	RW	1	IncVal12	Increment Value 12.
15:12	RW	1	IncVal11	Increment Value 11.
11:8	RW	1	IncVal10	Increment Value 10.
7:4	RW	1	IncVal9	Increment Value 9.
3:0	RW	1	IncVal8	Increment Value 8.

13.4.10 NPKTSCINCVAL2: NPKTSC Increment Value 2 Register

CSR Register Name: NPKTSCINCVAL2: NPKTSC Increment Value 2 Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 2108 **Offset End:** 210B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:28	RW	1	IncVal23	Increment Value 23.
27:24	RW	1	IncVal22	Increment Value 22.
23:20	RW	1	IncVal21	Increment Value 21.
19:16	RW	1	IncVal20	Increment Value 20.
15:12	RW	1	IncVal19	Increment Value 19.
11:8	RW	1	IncVal18	Increment Value 18.
7:4	RW	1	IncVal17	Increment Value 17.
3:0	RW	1	IncVal16	Increment Value 16.

13.4.11 NPKTSCINCVAL3: NPKTSC Increment Value 3 Register

CSR Register Name: NPKTSCINCVAL3: NPKTSC Increment Value 3 Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 210C **Offset End:** 210F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:28	RW	1	IncVal31	Increment Value 31.
27:24	RW	1	IncVal30	Increment Value 30.
23:20	RW	1	IncVal29	Increment Value 29.
19:16	RW	1	IncVal28	Increment Value 28.
15:12	RW	1	IncVal27	Increment Value 27.
11:8	RW	1	IncVal26	Increment Value 26.
7:4	RW	1	IncVal25	Increment Value 25.
3:0	RW	1	IncVal24	Increment Value 24.

13.4.12 NPKTSCINCVAL4: NPKTSC Increment Value 4 Register

CSR Register Name: NPKTSCINCVAL4: NPKTSC Increment Value 4 Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 2110 **Offset End:** 2113

Bits	Access	Default	Label	Bit Description
31:28	RW	1	IncVal39	Increment Value 39.
27:24	RW	1	IncVal38	Increment Value 38.
23:20	RW	1	IncVal37	Increment Value 37.
19:16	RW	1	IncVal36	Increment Value 36.
15:12	RW	1	IncVal35	Increment Value 35.
11:8	RW	1	IncVal34	Increment Value 34.
7:4	RW	1	IncVal33	Increment Value 33.
3:0	RW	1	IncVal32	Increment Value 32.

13.5 MSU Registers

13.5.1 MSU Registers Summary

MSU Registers			
Offset Start	Offset End	Symbol	Register Name/Function
A0000	A0003	MSUPARAMS	MSU Parameter Register
A0008	A000B	MSUSTATUS	MSU Status Register
A00FC	A00FF	MSUSPARE	MSU Spare Register
A0100	A0103	MSC0CTL	MSC0 Control Register
A0104	A0107	MSC0STS	MSC0 Status Register
A0108	A010B	MSC0BAR	MSC0 Base Address Register
A010C	A010F	MSC0DESTSZ	MSC0 Memory Buffer Size
A0110	A0113	MSC0MWP	MSC0 Memory Write Pointer
A0114	A0117	MSC0TBRP	MSC0 Trace Buffer Read Pointer
A0118	A011B	MSC0TBWP	MSC0 Trace Buffer Write Pointer
A011C	A011F	MSC0NWSA	MSC0 Next Window Starting Address
A0200	A0203	MSC1CTL	MSC1 Control Register
A0204	A0207	MSC1STS	MSC1 Status Register

MSU Registers			
Offset Start	**Offset End**	**Symbol**	**Register Name/Function**
A0208	A020B	MSC1BAR	MSC1 Base Address Register
A020C	A020F	MSC1DESTSZ	MSC1 Memory Buffer Size
A0210	A0213	MSC1MWP	MSC1 Memory Write Pointer
A0214	A0217	MSC1TBRP	MSC1 Trace Buffer Read Pointer
A0218	A021B	MSC1TBWP	MSC1 Trace Buffer Write Pointer
A021C	A021F	MSC1NWSA	MSC1 Next Window Starting Address
A02FC	A02FF	MSC1SPARE	MSC1 Spare Register

13.5.2 MSUPARAMS: MSU Parameter Register

CSR Register Name: **MSUPARAMS**: MSU Parameter Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** A0000	**Offset End:** A0003
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:16	RO	0	RSVD	MSUPARAMS Reserved
15:0	RO	100	MTBDEP	**MSC Trace Buffer Depth.** MTBDEP is the MSC Trace Buffer Depth Parameter for each of the two MTBs (both are of identical size and depth)

13.5.3 MSUSTATUS: MSU Status Register

CSR Register Name: **MSUSTATUS**: MSU Status Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** A0008	**Offset End:** A000B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:1	RO	0	RSVD	MSUSTATUS Reserved

<table>
<tr><td colspan="6"><b>CSR Register Name: MSUSTATUS</b>: MSU Status Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> A0008</td><td><b>Offset End:</b> A000B</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>0</td><td>RO</td><td>0</td><td>MSU_INT</td><td colspan="2"><b>msu_interrupt Capture Done output status.</b> Provides the user with readable status indicating that the capture has completed. Once set this status will be held high true until reset or both MSCnEN configuration bits are low.</td></tr>
</table>

13.5.4 MSUSPARE: MSU Spare Register

<table>
<tr><td colspan="6"><b>CSR Register Name: MSUSPARE</b>: MSU Spare Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> A00FC</td><td><b>Offset End:</b> A00FF</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:8</td><td>RO</td><td>0</td><td>RSVD</td><td colspan="2">MSUSPARE Reserved</td></tr>
<tr><td>7:0</td><td>RW</td><td>0</td><td>RSVD</td><td colspan="2">Reserved for future use.</td></tr>
</table>

13.5.5 MSC0CTL: MSC0 Control Register

<table>
<tr><td colspan="6"><b>CSR Register Name: MSC0CTL</b>: MSC0 Control Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> A0100</td><td><b>Offset End:</b> A0103</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:11</td><td>RO</td><td>0</td><td>RSVD0</td><td colspan="2">MSC0CTL Reserved</td></tr>
</table>

<table>
<tr><td colspan="6"><b>CSR Register Name:</b> <b>MSC0CTL</b>: MSC0 Control Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b>
A0100</td><td><b>Offset End:</b>
A0103</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>10:8</td><td>RW</td><td>3</td><td>MSC_LEN</td><td colspan="2"><b>MSCn outbound write burst length.</b>
The MSC by default will send only a full cache line (64 bytes) when writing to memory. MSCn_LEN is represented as the number of data cycles that is transferred. MSCn_LEN is used for both single buffer and multi block mode. If the primary bus width is 64 bits, the reset value for MSCn_LEN = 3b011, for a transfer length of 8 * 8 bytes per transfer = 64 bytes total. If the primary bus width is 128 bits, the reset value for MSCn_LEN = 3b010, for a transfer length of 4 * 16 bytes per transfer = 64 bytes total. This outbound posted write burst length can be programmed to any supported value by the software for flexibility. Here is the encoding for MSCn_LEN:
3b000 : Length = 1
3b001 : Length = 2
3b010 : Length = 4 default if width = 128
3b011 : Length = 8 default if width = 64
3b100 : Length = 16</td></tr>
<tr><td>7:6</td><td>RO</td><td>0</td><td>RSVD1</td><td colspan="2">MSC0CTL Reserved</td></tr>
<tr><td>5:4</td><td>RW</td><td>0</td><td>MSC_MODE</td><td colspan="2"><b>MSCn Mode Control</b>
<b>0x0: single block or CSR-driven mode.</b>
0x1: multi-block or linked-list mode
0x2: DCI Trace Handler Mode
0x3: Internal Buffer mode
See also the note in section 2.2.4 regarding reading of data when switching from a normal mode to Debug mode.</td></tr>
<tr><td>3</td><td>RO</td><td>0</td><td>RSVD2</td><td colspan="2">MSC0CTL Reserved</td></tr>
</table>

<table>
<tr><td colspan="6"><b>CSR Register Name: MSC0CTL</b>: MSC0 Control Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b>
A0100</td><td><b>Offset End:</b>
A0103</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>2</td><td>RW</td><td>0</td><td>MSC_RD_HDR_OVRD</td><td colspan="2"><b>MSCn Read Header Override.</b> In multi window mode, before the MSU begins writing data to the current memory block it performs a SW header read. The read response for this can take some time due to latency in the fabric. For performance optimization, the MSU presumptively starts to write trace data to this block without knowing the Block Size which is contained in the SW header. Since all blocks are a minimum of 4 Kbytes, this is a fairly safe thing to do as long as it stops at this 4 Kbyte boundary if it hasn t yet received the SW header read response. The MSU must receive the SW header read response that contains the Block Size set to a value above 4Kbytes before it can proceed past this 4 Kbyte boundary. When RD_HDR_OVRD = 0, each MSC will operate as described above, which is considered the normal operating mode. When RD_HRD_OVRD = 1, each MSC will wait for the SW header read response containing this Block Size before it writes any data to this block. This will reduce the performance of the MSC operation.</td></tr>
<tr><td>1</td><td>RW</td><td>0</td><td>MSC_WRAPEN</td><td colspan="2"><b>Memory Wrap Enabled, MSCn.</b>
0: no wrap
1: wrap enabled
Note: The usage of the memory wrap enable is dependent upon the operational mode of the MSC. In single block mode wrapping within the single memory block is enabled and/or disabled with this CSR In multi-block mode wrapping is always enabled, so WRAPENn is not used. In DCI Trace Handler mode this CSR is unused, as the MSU will wrap within its trace buffer, but will not be writing the trace data to memory. In Internal Buffer mode wrapping within the trace buffer is enabled and/or disabled with this CSR</td></tr>
<tr><td>0</td><td>RW</td><td>0</td><td>MSC_EN</td><td colspan="2"><b>MSCn Enable.</b> Enables MSCn operation when set.</td></tr>
</table>

13.5.6 MSC0STS: MSC0 Status Register

CSR Register Name: MSC0STS: MSC0 Status Register			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** A0104	**Offset End:** A0107

Bits	Access	Default	Label	Bit Description
31:8	RO	0	RSVD	MSC0STS Reserved
7:4	RO	0	RSVD	Reserved for future use.
3	RW/1C	0	ERROR_STATUS	**MSU Outbound Error Status.** If the MSU receives an error type response on an outbound read, then it will set this error status bit and halt its operation.
2	RO	0	MSC_PLE	**MSCn PipeLine Empty.** When set, indicates that there is no data in the pipeline intended for MSCn or in the MSC Trace Buffer for this MSCn. When software is moving to the retrieval stage, it can check this bit to ensure no data is pending before it begins processing the data.
1	RO	0	WRAPSTAT	**Memory Wrap Status, MSCn.** 0: memory wrap did not occur. That is, the write pointer never wrapped around to zero. Trace data should be reconstructed starting from the start of the memory buffer or trace buffer. 1: memory wrap occurred. Note: This status is dependent upon the operational mode of the MSC.
0	RO	0	RSVD0	MSC0STS Reserved

13.5.7 MSC0BAR: MSC0 Base Address Register

CSR Register Name: MSC0BAR: MSC0 Base Address Register			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** A0108	**Offset End:** A010B

Bits	Access	Default	Label	Bit Description
31:0	RW	0	MSCBASE	**MSCn Base Address Specifies the starting address of the trace buffer to which the MSCn will store its data.** The value programmed in MSCnBASE is shifted left by 12 bits to create the physical address [43:12]. Due to this shift, the starting address is restricted to start on a 4-kbyte boundary. Note: MSCnBASE is used in both single block mode and in multi block mode to point to the starting address of memory. In multi block mode, this MSCn Base Address will point to the first block in memory and the MSCn will interpret the remaining block and window addresses from the header information in memory.

13.5.8 MSC0DESTSZ: MSC0 Memory Buffer Size

CSR Register Name: MSC0DESTSZ: MSC0 Memory Buffer Size				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** A010C	**Offset End:** A010F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RW	0	MSCSIZE	**Size of memory buffer for MSCn.** Specifies the size of the system memory buffer for MSCn in terms of 4kB pages bits 43:12. By shifting the value in MSCnSIZE[31:0] left 12 bits, the result is MSCnSIZE multiplied by 4 kBytes. A value of 0h is undefined. Only applicable when operating in CSR-driven mode to indicate the memory buffer size of the single memory block.

13.5.9 MSC0MWP: MSC0 Memory Write Pointer

CSR Register Name: MSC0MWP: MSC0 Memory Write Pointer			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** A0110	**Offset End:** A0113

Bits	Access	Default	Label	Bit Description
31:0	RO	0	MSC_MWP	**MSCn Memory Write Pointer.** The current value of the memory write pointer for MSCn. In single block mode, software will use this information along with the wrap status to interpret the trace data stored in memory. At the end of the capture, MSCnMWP points to the next location in memory that data would be written to. This means that MSCnMWP-1 is the last byte of trace data that was successfully written. If wrapping is enabled, and WRAPSTAT0 is 1, then the trace buffer wrapped. In this case, the beginning of the stored trace is at MSCnMWP and end of the stored trace is at MSCnMWP-1. If wrapping is not enabled, or WRAPSTAT0 is low, then the start of the trace buffer is at MSCnBAR and the end is at MSCnMWP-1. MSCnMWP is not useful for multi block mode. The information required to reconstruct the trace is stored into the memory headers. MSCnMWP is loaded with the contents of MSCnBAR when MSCnEN = 0. For this reason, at the end of the trace capture the contents of MSCnMWP should be read prior to clearing the MSCnEN if this information is required. Note: The actual implementation of the memory write pointer is larger than 32 bits, and MSCnMWP is the readable status of the lower 32 bits. For this reason, MSCnMWP is lower 32 bits of the physical memory address. MSCnMWP represents the lower 32-bit offset that begins at the MSCnBAR shifted left by 12 bits. This will be problematic only if the trace buffer crosses a 4 GByte boundary.

13.5.10 MSC0TBRP: MSC0 Trace Buffer Read Pointer

CSR Register Name: **MSC0TBRP**: MSC0 Trace Buffer Read Pointer			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** A0114	**Offset End:** A0117

Bits	Access	Default	Label	Bit Description
31:0	RO	0	MSC_TBRP	**MSCn Trace Buffer Read Pointer.** Gives the current value of the Trace Buffer 0 read pointer for MSCn. This is mainly used for debug visibility. MSCnTBRP is reset to all 0 s when MSCnEN = 0. For this reason, at the end of the trace capture the contents of MSCnTBRP should be read prior to clearing the MSCnEN if this information is required. Note: This is applicable for all operational modes. Note also that the trace buffer depth is set by a parameter called MTBDEP. The pointers to this trace buffer are of size [LOG2_MTBDEP-1:0] and is lower justified within this 32-bit register and the upper bits are read as zero. Note also that in all modes except debug mode the MSI implements a state machine that pre-fetches data into the output stage of the trace buffer, then increments the read pointer. If the user reads this MSCnTBRP value while the MSI is operational, the value of this pointer will be one greater than the head of the trace buffer.

13.5.11 MSC0TBWP: MSC0 Trace Buffer Write Pointer

CSR Register Name: MSC0TBWP: MSC0 Trace Buffer Write Pointer			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** A0118	**Offset End:** A011B

Bits	Access	Default	Label	Bit Description
31:0	RO	0	MSC_TBWP	**MSCn Trace Buffer Write Pointer.** Gives the current value of the Trace Buffer 0 write pointer for MSCn. This is mainly used for debug visibility. It is useful in debug mode to point to the tail of the trace, and if wrapping is enabled will indicate the head of the trace if wrap occurred. MSCnTBWP is reset to all 0 s when MSCnEN = 0. For this reason, at the end of the trace capture the contents of MSCnTBWP should be read prior to clearing the MSCnEN if this information is required. Note: This is applicable for all operational modes. Note also that the trace buffer depth is set by a parameter called MTBDEP. The pointers to this trace buffer are of size [LOG2_MTBDEP-1:0] and is lower justified within this 32-bit register and the upper bits are read as zero.

13.5.12 MSC0NWSA: MSC0 Next Window Starting Address

CSR Register Name: **MSC0NWSA**: MSC0 Next Window Starting Address			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** A011C	**Offset End:** A011F

Bits	Access	Default	Label	Bit Description
31:0	RO	0	MSC_NWSA	MSCn Next Window Starting Address. Gives the current value of the Next Window Starting Address as programmed by the software into the current block s software header. This is valuable information when in multi-window mode when North Peak will go through a power down event. Before powering down, the software can read the starting address of the next memory window from this status register and save it. Upon resuming power, the MSC can be configured to start at the next window by programming the value that was saved from this CSR into the MSCnBAR. Note: This is applicable for multi-window mode only. Note: The contents of this register are shifted left 12 bits before being used by the MSCn to create a physical memory address. Note: The value of this register is preserved only while MSCnEN = 1. If MSCnEN is cleared to zero, the value of NWSA will be erased. Therefore, it is important to read the value of this register before disabling the associated MSC. Note: when reading this register for save/restore purposes low power mode support, software must ensure that the associated MSC is in a stable state, such as the idle state. This can be determined by reading the MSCnSTATE bits in the MSCnSTS register.

13.5.13 MSC1CTL: MSC1 Control Register

CSR Register Name: MSC1CTL: MSC1 Control Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** A0200	**Offset End:** A0203
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:11	RO	0	RSVD	MSC0CTL Reserved

<table>
<tr><td colspan="5"><b>CSR Register Name:</b> <b>MSC1CTL</b>: MSC1 Control Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b>
A0200
<b>Offset End:</b>
A0203</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th>Bit Description</th></tr>
<tr><td>10:8</td><td>RW</td><td>3</td><td>MSC_LEN</td><td><b>MSCn outbound write burst length.</b>
The MSC by default will send only a full cache line (64 bytes) when writing to memory. MSCn_LEN is represented as the number of data cycles that is transferred. MSCn_LEN is used for both single buffer and multi block mode. If the primary bus width is 64 bits, the reset value for MSCn_LEN = 3b011, for a transfer length of 8 * 8 bytes per transfer = 64 bytes total. If the primary bus width is 128 bits, the reset value for MSCn_LEN = 3b010, for a transfer length of 4 * 16 bytes per transfer = 64 bytes total. This outbound posted write burst length can be programmed to any supported value by the software for flexibility. Here is the encoding for MSCn_LEN:
3b000 : Length = 1
3b001 : Length = 2
3b010 : Length = 4 default if width = 128
3b011 : Length = 8 default if width = 64
3b100 : Length = 16</td></tr>
<tr><td>7:6</td><td>RO</td><td>0</td><td>RSVD0</td><td>MSC0CTL Reserved</td></tr>
<tr><td>5:4</td><td>RW</td><td>0</td><td>MSC_MODE</td><td><b>MSCn Mode Control</b>
<b>0x0: single block or CSR-driven mode.</b>
0x1: multi-block or linked-list mode
0x2: DCI Trace Handler Mode
0x3: Internal Buffer mode
See also the note in section 2.2.4 regarding reading of data when switching from a normal mode to Debug mode.</td></tr>
<tr><td>3</td><td>RO</td><td>0</td><td>RSVD1</td><td>MSC0CTL Reserved</td></tr>
</table>

<table>
<tr><td colspan="6"><b>CSR Register Name: MSC1CTL</b>: MSC1 Control Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b>
A0200</td><td><b>Offset End:</b>
A0203</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>2</td><td>RW</td><td>0</td><td>MSC_RD_HDR_OVRD</td><td colspan="2"><b>MSCn Read Header Override.</b> In multi window mode, before the MSU begins writing data to the current memory block it performs a SW header read. The read response for this can take some time due to latency in the fabric. For performance optimization, the MSU presumptively starts to write trace data to this block without knowing the Block Size which is contained in the SW header. Since all blocks are a minimum of 4 Kbytes, this is a fairly safe thing to do as long as it stops at this 4 Kbyte boundary if it hasn t yet received the SW header read response. The MSU must receive the SW header read response that contains the Block Size set to a value above 4Kbytes before it can proceed past this 4 Kbyte boundary. When RD_HDR_OVRD = 0, each MSC will operate as described above, which is considered the normal operating mode. When RD_HRD_OVRD = 1, each MSC will wait for the SW header read response containing this Block Size before it writes any data to this block. This will reduce the performance of the MSC operation.</td></tr>
<tr><td>1</td><td>RW</td><td>0</td><td>MSC_WRAPEN</td><td colspan="2"><b>Memory Wrap Enabled, MSCn.</b>
0: no wrap
1: wrap enabled
Note: The usage of the memory wrap enable is dependent upon the operational mode of the MSC. In single block mode wrapping within the single memory block is enabled and/or disabled with this CSR In multi-block mode wrapping is always enabled, so WRAPENn is not used. In DCI Trace Handler mode this CSR is unused, as the MSU will wrap within its trace buffer, but will not be writing the trace data to memory. In Internal Buffer mode wrapping within the trace buffer is enabled and/or disabled with this CSR</td></tr>
<tr><td>0</td><td>RW</td><td>0</td><td>MSC_EN</td><td colspan="2"><b>MSCn Enable.</b> Enables MSCn operation when set.</td></tr>
</table>

13.5.14 MSC1STS: MSC1 Status Register

CSR Register Name: MSC1STS: MSC1 Status Register			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** A0204	**Offset End:** A0207

Bits	Access	Default	Label	Bit Description
31:8	RO	0	RSVD	MSC0STS Reserved
7:4	RO	0	RSVD	Reserved for future use.
3	RW/1C	0	ERROR_STATUS	**MSU Outbound Error Status.** If the MSU receives an error type response on an outbound read, then it will set this error status bit and halt its operation.
2	RO	0	MSC_PLE	**MSCn PipeLine Empty.** When set, indicates that there is no data in the pipeline intended for MSCn or in the MSC Trace Buffer for this MSCn. When software is moving to the retrieval stage, it can check this bit to ensure no data is pending before it begins processing the data.
1	RO	0	WRAPSTAT	**Memory Wrap Status, MSCn.** 0: memory wrap did not occur. That is, the write pointer never wrapped around to zero. Trace data should be reconstructed starting from the start of the memory buffer or trace buffer. 1: memory wrap occurred. Note: This status is dependent upon the operational mode of the MSC.
0	RO	0	RSVD0	MSC0STS Reserved

13.5.15 MSC1BAR: MSC1 Base Address Register

CSR Register Name: MSC1BAR: MSC1 Base Address Register			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** A0208	**Offset End:** A020B

Bits	Access	Default	Label	Bit Description
31:0	RW	0	MSCBASE	**MSCn Base Address Specifies the starting address of the trace buffer to which the MSCn will store its data.** The value programmed in MSCnBASE is shifted left by 12 bits to create the physical address [43:12]. Due to this shift, the starting address is restricted to start on a 4-kbyte boundary. Note: MSCnBASE is used in both single block mode and in multi block mode to point to the starting address of memory. In multi block mode, this MSCn Base Address will point to the first block in memory and the MSCn will interpret the remaining block and window addresses from the header information in memory.

13.5.16 MSC1DESTSZ: MSC1 Memory Buffer Size

CSR Register Name: **MSC1DESTSZ**: MSC1 Memory Buffer Size				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** A020C	**Offset End:** A020F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RW	0	MSCSIZE	**Size of memory buffer for MSCn.** Specifies the size of the system memory buffer for MSCn in terms of 4kB pages bits 43:12. By shifting the value in MSCnSIZE[31:0] left 12 bits, the result is MSCnSIZE multiplied by 4 kBytes. A value of 0h is undefined. Only applicable when operating in CSR-driven mode to indicate the memory buffer size of the single memory block.

13.5.17 MSC1MWP: MSC1 Memory Write Pointer

CSR Register Name: **MSC1MWP**: MSC1 Memory Write Pointer			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** A0210	**Offset End:** A0213

Bits	Access	Default	Label	Bit Description
31:0	RO	0	MSC_MWP	**MSCn Memory Write Pointer.** The current value of the memory write pointer for MSCn. In single block mode, software will use this information along with the wrap status to interpret the trace data stored in memory. At the end of the capture, MSCnMWP points to the next location in memory that data would be written to. This means that MSCnMWP-1 is the last byte of trace data that was successfully written. If wrapping is enabled, and WRAPSTAT0 is high 1, then the trace buffer wrapped. In this case, the beginning of the stored trace is at MSCnMWP and end of the stored trace is at MSCnMWP-1. If wrapping is not enabled, or WRAPSTAT0 is low, then the start of the trace buffer is at MSCnBAR and the end is at MSCnMWP-1. MSCnMWP is not useful for multi block mode. The information required to reconstruct the trace is stored into the memory headers. MSCnMWP is loaded with the contents of MSCnBAR when MSCnEN = 0. For this reason, at the end of the trace capture the contents of MSCnMWP should be read prior to clearing the MSCnEN if this information is required. Note: The actual implementation of the memory write pointer is larger than 32 bits, and MSCnMWP is the readable status of the lower 32 bits. For this reason, MSCnMWP is lower 32 bits of the physical memory address. MSCnMWP represents the lower 32-bit offset that begins at the MSCnBAR shifted left by 12 bits. This will be problematic only if the trace buffer crosses a 4 GByte boundary.

13.5.18 MSC1TBRP: MSC1 Trace Buffer Read Pointer

CSR Register Name: **MSC1TBRP**: MSC1 Trace Buffer Read Pointer			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** A0214	**Offset End:** A0217

Bits	Access	Default	Label	Bit Description
31:0	RO	0	MSC_TBRP	**MSCn Trace Buffer Read Pointer.** Gives the current value of the Trace Buffer 0 read pointer for MSCn. This is mainly used for debug visibility. MSCnTBRP is reset to all 0 s when MSCnEN = 0. For this reason, at the end of the trace capture the contents of MSCnTBRP should be read prior to clearing the MSCnEN if this information is required. Note: This is applicable for all operational modes. Note also that the trace buffer depth is set by a parameter called MTBDEP. The pointers to this trace buffer are of size [LOG2_MTBDEP-1:0] and is lower justified within this 32-bit register and the upper bits are read as zero. Note also that in all modes except debug mode the MSI implements a state machine that pre-fetches data into the output stage of the trace buffer, then increments the read pointer. If the user reads this MSCnTBRP value while the MSI is operational, the value of this pointer will be one greater than the head of the trace buffer.

13.5.19 MSC1TBWP: MSC1 Trace Buffer Write Pointer

CSR Register Name: **MSC1TBWP**: MSC1 Trace Buffer Write Pointer			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** A0218	**Offset End:** A021B

Bits	Access	Default	Label	Bit Description
31:0	RO	0	MSC_TBWP	**MSCn Trace Buffer Write Pointer.** Gives the current value of the Trace Buffer 0 write pointer for MSCn. This is mainly used for debug visibility. It is useful in debug mode to point to the tail of the trace, and if wrapping is enabled will indicate the head of the trace if wrap occurred. MSCnTBWP is reset to all 0 s when MSCnEN = 0. For this reason, at the end of the trace capture the contents of MSCnTBWP should be read prior to clearing the MSCnEN if this information is required. Note: This is applicable for all operational modes. Note also that the trace buffer depth is set by a parameter called MTBDEP. The pointers to this trace buffer are of size [LOG2_MTBDEP-1:0] and is lower justified within this 32-bit register and the upper bits are read as zero.

13.5.20 MSC1NWSA: MSC1 Next Window Starting Address

CSR Register Name: **MSC1NWSA**: MSC1 Next Window Starting Address			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** A021C	**Offset End:** A021F

Bits	Access	Default	Label	Bit Description
31:0	RO	0	MSC_NWSA	MSCn Next Window Starting Address. Gives the current value of the Next Window Starting Address as programmed by the software into the current block s software header. This is valuable information when in multi-window mode when North Peak will go through a power down event. Before powering down, the software can read the starting address of the next memory window from this status register and save it. Upon resuming power, the MSC can be configured to start at the next window by programming the value that was saved from this CSR into the MSCnBAR. Note: This is applicable for multi-window mode only. Note: The contents of this register are shifted left 12 bits before being used by the MSCn to create a physical memory address. Note: The value of this register is preserved only while MSCnEN = 1. If MSCnEN is cleared to zero, the value of NWSA will be erased. Therefore, it is important to read the value of this register before disabling the associated MSC. Note: when reading this register for save/restore purposes low power mode support, software must ensure that the associated MSC is in a stable state, such as the idle state. This can be determined by reading the MSCnSTATE bits in the MSCnSTS register.

13.5.21 MSC1SPARE: MSC1 Spare Register

CSR Register Name: **MSC1SPARE**: MSC1 Spare Register				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** A02FC	**Offset End:** A02FF
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:8	RO	0	RSVD	MSC0SPARE Reserved
7:0	RW	0	MSC_SPARE	Spare - These reserved bits will have actual flip flops instantiated in the RTL that can be read and written. They will be saved for debug purposes.

13.6 DCI Trace Handler Registers

13.6.1 DCI Trace Handler Registers Summary

DCI Trace Handler Registers			
Offset Start	Offset End	Symbol	Register Name/Function
A1000	A1003	STREAMCFG1	DCI-TH Streaming Configuration
A1004	A1007	STREAMCFG2	DCI-TH Streaming Configuration 2
A1008	A100B	DBCSTSCMD	DBC.Trace Status Command
A100C	A100F	DBCSTSDATA	DBC Status Data
A1010	A1013	EXISTSCMD	EXI Bridge Status Command
A1014	A1017	EXISTSDATA	Exi Status Data
A1018	A101B	TIMEOUT	DCI-TH Time out register
A101C	A101F	BRIDGE_BASE_ADDR_HI	EXI Bridge Base Address Hi register
A1020	A1023	BRIDGE_BASE_ADDR_LO	EXI Bridge Base Address Lo register
A1024	A1027	DBC_BASE_ADDR_HI	DBC Bridge Base Address Hi register
A1028	A102B	DBC_BASE_ADDR_LO	DBC Base Address Lo register
A102C	A102F	NPKHSTS	DCI-TH Status register
A1030	A1033	MAXTIMEOUTS	DCI-TH Retry Register
A1034	A1037	RSVD2	DCI-TH Reserved Register 2

13.6.2 STREAMCFG1: DCI-TH Streaming Configuration

CSR Register Name: **STREAMCFG1**: DCI-TH Streaming Configuration			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** A1000	**Offset End:** A1003

Bits	Access	Default	Label	Bit Description
31	RW	0	RESET	**DCI Trace handler Soft Reset.** Writing 1 to this will reset DCI Trace Handler state machine and credits. This bit is self-clearing, and will always read as zero.
30	RO	0	RSVD1	Reserved for future use.
29	RO	0	RSVD0	Reserved for future use.
28	RW	0	ENABLE	**DCI Trace Handler Enable.** 0: DCI TH is disabled 1: DCI TH is enabled, and will stream data to the destination.
27	RO	0	FLUSHDONE	**Flush Done.** This bit is set when the flush is completed.
26	RW	0	FLUSH	**Flush DCI TH data.** Writing 1 to this bit will cause the DCI TH to flush trace data to current streaming destination. This bit is not self-clearing.
25:14	RW	0	EXIOFFSET	**DCI Bridge Offset.** Specifies the offset from the DCI Bridge BAR EXIBASEHI + EXIBASELO that the DCI TH will use when writing data to the DCI Bridge
13:2	RW	0	DBCOFFSET	**DbC.**Trace DMA Request Offset. Specifies the offset from the DbC BAR DBCBASEHI + DBCBASELO that the the DCI Trace Handler will use when sending DMA requests to the DbC.Trace.
1	RO	0	STRMMODE	**Current streaming mode.** Reflects the current streaming mode. 0: BSSB Mode 1: DBC.Trace Mode.
0	RW	0	SETSTRMMODE	**Streaming mode:** **0: DBC.**Trace mode. 1: BSSB Mode Changing this bit will cause the DCI TH to change streaming mode after all pending transactions in current streaming mode are completed.

13.6.3 STREAMCFG2: DCI-TH Streaming Configuration 2

CSR Register Name: **STREAMCFG2**: DCI-TH Streaming Configuration 2			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** A1004	**Offset End:** A1007

Bits	Access	Default	Label	Bit Description
31:12	RO	0	RSVD	STREAMCFG2 Reserved
11:0	RW	0	EXISTSOFFSET	**DCI Status Register Offset.** Specifies the offset of the DCI streaming Status register that the DCI Trace handler uses for sending status updates. This register and feature is currently not used.

13.6.4 DBCSTSCMD: DBC.Trace Status Command

CSR Register Name: DBCSTSCMD: DBC.Trace Status Command				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** A1008	**Offset End:** A100B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:16	RSV	0	DATA	DBCSTSCMD Reserved
3:0	RW	0	CMD	**Status return command :** 0000 : none 0001 : Credit return 1111 : Error report

13.6.5 DBCSTSDATA: DBC Status Data

CSR Register Name: DBCSTSDATA: DBC Status Data				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** A100C	**Offset End:** A100F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RSV	0	DATA	Data value associated with status cmd.

13.6.6 EXISTSCMD: EXI Bridge Status Command

CSR Register Name: EXISTSCMD: EXI Bridge Status Command				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** A1010	**Offset End:** A1013
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:16	WO	0	DATA	EXISTSCMD Reserved

CSR Register Name: EXISTSCMD: EXI Bridge Status Command					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** A1010	**Offset End:** A1013
Bits	**Access**	**Default**	**Label**	**Bit Description**	
3:0	RW	0	CMD	Status return command : 0000 : None 0001 : Credit Return 1111 : Error report others: reserved	

13.6.7 EXISTSDATA: Exi Status Data

CSR Register Name: EXISTSDATA: Exi Status Data					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** A1014	**Offset End:** A1017
Bits	**Access**	**Default**	**Label**	**Bit Description**	
31:0	RW	0	DATA	Data value associated with status cmd.	

13.6.8 TIMEOUT: DCI-TH Time out register

CSR Register Name: TIMEOUT: DCI-TH Time out register					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** A1018	**Offset End:** A101B
Bits	**Access**	**Default**	**Label**	**Bit Description**	
31:0	RW	0	TIMEOUT	**DCI Trace Handler Time out register.** Time out value for DCI Trace handler 00000000h : Time out disabled all other values : Time out value.	

13.6.9 BRIDGE_BASE_ADDR_HI: EXI Bridge Base Address Hi register

CSR Register Name: BRIDGE_BASE_ADDR_HI: EXI Bridge Base Address Hi register			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** A101C	**Offset End:** A101F

Bits	Access	Default	Label	Bit Description
31:0	RW	0	ADDR_HI	**DCI Bridge Base Address.** Specifies the high-order DWord of the DCI Bridge base address bits 63:32

13.6.10 BRIDGE_BASE_ADDR_LO: EXI Bridge Base Address Lo register

CSR Register Name: BRIDGE_BASE_ADDR_LO: EXI Bridge Base Address Lo register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** A1020 **Offset End:** A1023
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RW	0	ADDR_LO	**DCI Bridge Base Address.** Specifies the low-order DWord of the DCI Bridge base address bits 31:0

13.6.11 DBC_BASE_ADDR_HI: DBC Bridge Base Address Hi register

CSR Register Name: DBC_BASE_ADDR_HI: DBC Bridge Base Address Hi register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** A1024 **Offset End:** A1027
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RW	0	ADDR_HI	**DbC.**Trace Base Address. Specifies the high-order DWord of the DbC.Trace base address bits 63:32

13.6.12 DBC_BASE_ADDR_LO: DBC Base Address Lo register

CSR Register Name: DBC_BASE_ADDR_LO: DBC Base Address Lo register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** A1028 **Offset End:** A102B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RW	0	ADDR_LO	**DbC.**Trace Base Address. Specifies the low-order DWord of the DbC.Trace base address bits 31:0

13.6.13 NPKHSTS: DCI-TH Status register

CSR Register Name: NPKHSTS: DCI-TH Status register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** A102C **Offset End:** A102F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RO	0	TBD	Reserved for future use.

13.6.14 MAXTIMEOUTS: DCI-TH Retry Register

CSR Register Name: MAXTIMEOUTS: DCI-TH Retry Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** A1030 **Offset End:** A1033
Bits	**Access**	**Default**	**Label**	**Bit Description**
5:0	RW	0	MAXTIMEOUTS	**DCI Trace Handler Retry Register.** Maximum number of retries. When non-zero, specifies the maximum number of times the DCI Trace Handler will reset itself and retry its data

13.6.15 RSVD2: DCI-TH Reserved Register 2

CSR Register Name: RSVD2: DCI-TH Reserved Register 2				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** A1034 **Offset End:** A1037
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RO	0	TBD	Reserved for future use.

13.7 CTS Registers

13.7.1 CTS Registers Summary

CTS Registers

Offset Start	Offset End	Symbol	Register Name/Function
03000	03003	CLAUSE_EVENT_MODE_C0S0	Clause Event Mode Register C0S0
03004	03007	CLAUSE_EVENT_MODE_C1S0	Clause Event Mode Register C1S0
03008	0300B	CLAUSE_EVENT_MODE_C2S0	Clause Event Mode Register C2S0
0300C	0300F	CLAUSE_EVENT_MODE_C3S0	Clause Event Mode Register C3S0
03018	0301B	CLAUSE_EVENT_MODE_C0S1	Clause Event Mode Register C0S1
0301C	0301F	CLAUSE_EVENT_MODE_C1S1	Clause Event Mode Register C1S1
03020	03023	CLAUSE_EVENT_MODE_C2S1	Clause Event Mode Register C2S1
03024	03027	CLAUSE_EVENT_MODE_C3S1	Clause Event Mode Register C3S1
03030	03033	CLAUSE_EVENT_MODE_C0S2	Clause Event Mode Register C0S2
03034	03037	CLAUSE_EVENT_MODE_C1S2	Clause Event Mode Register C1S2
03038	0303B	CLAUSE_EVENT_MODE_C2S2	Clause Event Mode Register C2S2
0303C	0303F	CLAUSE_EVENT_MODE_C3S2	Clause Event Mode Register C3S2
03048	0304B	CLAUSE_EVENT_MODE_C0S3	Clause Event Mode Register C0S3
0304C	0304F	CLAUSE_EVENT_MODE_C1S3	Clause Event Mode Register C1S3
03050	03053	CLAUSE_EVENT_MODE_C2S3	Clause Event Mode Register C2S3
03054	03057	CLAUSE_EVENT_MODE_C3S3	Clause Event Mode Register C3S3
030C0	030C3	CLAUSE_EVENT_ENABLE_C0S0	Clause Event Enable Register C0S0
030C4	030C7	CLAUSE_EVENT_ENABLE_C1S0	Clause Event Enable Register C1S0
030C8	030CB	CLAUSE_EVENT_ENABL	Clause Event Enable

CTS Registers			
Offset Start	Offset End	Symbol	Register Name / Function
		E_C2S0	Register C2S0
030CC	030CF	CLAUSE_EVENT_ENABLE_C3S0	Clause Event Enable Register C3S0
030D8	030DB	CLAUSE_EVENT_ENABLE_C0S1	Clause Event Enable Register C0S1
030DC	030DF	CLAUSE_EVENT_ENABLE_C1S1	Clause Event Enable Register C1S1
030E0	030E3	CLAUSE_EVENT_ENABLE_C2S1	Clause Event Enable Register C2S1
030E4	030E7	CLAUSE_EVENT_ENABLE_C3S1	Clause Event Enable Register C3S1
030F0	030F3	CLAUSE_EVENT_ENABLE_C0S2	Clause Event Enable Register C0S2
030F4	030F7	CLAUSE_EVENT_ENABLE_C1S2	Clause Event Enable Register C1S2
030F8	030FB	CLAUSE_EVENT_ENABLE_C2S2	Clause Event Enable Register C2S2
030FC	030FF	CLAUSE_EVENT_ENABLE_C3S2	Clause Event Enable Register C3S2
03108	0310B	CLAUSE_EVENT_ENABLE_C0S3	Clause Event Enable Register C0S3
0310C	0310F	CLAUSE_EVENT_ENABLE_C1S3	Clause Event Enable Register C1S3
03110	03113	CLAUSE_EVENT_ENABLE_C2S3	Clause Event Enable Register C2S3
03114	03117	CLAUSE_EVENT_ENABLE_C3S3	Clause Event Enable Register C3S3
03180	03183	CLAUSE_ACTION_CONTROL_C0S0	Clause Action Control Registers C0S0
03184	03187	CLAUSE_ACTION_CONTROL_C1S0	Clause Action Control Registers C1S0
03188	0318B	CLAUSE_ACTION_CONTROL_C2S0	Clause Action Control Registers C2S0
0318C	0318F	CLAUSE_ACTION_CONTROL_C3S0	Clause Action Control Registers C3S0
03198	0319B	CLAUSE_ACTION_CONTROL_C0S1	Clause Action Control Registers C0S1
0319C	0319F	CLAUSE_ACTION_CONTROL_C1S1	Clause Action Control Registers C1S1
031A0	031A3	CLAUSE_ACTION_CONTROL_C2S1	Clause Action Control Registers C2S1

CTS Registers			
Offset Start	Offset End	Symbol	Register Name/Function
031A4	031A7	CLAUSE_ACTION_CONTROL_C3S1	Clause Action Control Registers C3S1
031B0	031B3	CLAUSE_ACTION_CONTROL_C0S2	Clause Action Control Registers C0S2
031B4	031B7	CLAUSE_ACTION_CONTROL_C1S2	Clause Action Control Registers C1S2
031B8	031BB	CLAUSE_ACTION_CONTROL_C2S2	Clause Action Control Registers C2S2
031BC	031BF	CLAUSE_ACTION_CONTROL_C3S2	Clause Action Control Registers C3S2
031C8	031CB	CLAUSE_ACTION_CONTROL_C0S3	Clause Action Control Registers C0S3
031CC	031CF	CLAUSE_ACTION_CONTROL_C1S3	Clause Action Control Registers C1S3
031D0	031D3	CLAUSE_ACTION_CONTROL_C2S3	Clause Action Control Registers C2S3
031D4	031D7	CLAUSE_ACTION_CONTROL_C3S3	Clause Action Control Registers C3S3
03240	03243	SCT0_CONTROL	SCT0 Control Register
03244	03247	SCT1_CONTROL	SCT1 Control Register
03250	03253	LCT0_LOWER_CONTROL	LCT0 Lower Control Register
03254	03257	LCT0_UPPER_CONTROL	LCT0 Upper Control Register
03270	03273	SCT0_CURRENT_STATUS	SCT0 Current Status Register
03274	03277	SCT1_CURRENT_STATUS	SCT1 Current Status Register
03280	03283	LCT0_LOWER_CURRENT_STATUS	LCT0 Lower Current Status Register
03284	03287	LCT0_UPPER_CURRENT_STATUS	LCT0 Upper Current Status Register
032A0	032A3	CTS_STATUS	CTS Status Register
032A4	032A7	CTS_CONTROL	CTS Control Register
032A8	032AB	LCT0_PEAK_TIMER_LOWER_STATUS	LCT0 Peak Timer Lower Status Register
032AC	032AF	LCT0_PEAK_TIMER_UPPER_STATUS	LCT0 Peak Timer Upper Status Register
032C8	032CB	PARAMETER_STATUS	Parameter Status Register

CTS Registers			
Offset Start	**Offset End**	**Symbol**	**Register Name/Function**
03300	03303	EXTENDED_CLAUSE_ACTION_CONTROL_C0S0	Extended Clause Action Control Registers C0S0
03304	03307	EXTENDED_CLAUSE_ACTION_CONTROL_C1S0	Extended Clause Action Control Registers C1S0
03308	0330B	EXTENDED_CLAUSE_ACTION_CONTROL_C2S0	Extended Clause Action Control Registers C2S0
0330C	0330F	EXTENDED_CLAUSE_ACTION_CONTROL_C3S0	Extended Clause Action Control Registers C3S0
03318	0331B	EXTENDED_CLAUSE_ACTION_CONTROL_C0S1	Extended Clause Action Control Registers C0S1
0331C	0331F	EXTENDED_CLAUSE_ACTION_CONTROL_C1S1	Extended Clause Action Control Registers C1S1
03320	03323	EXTENDED_CLAUSE_ACTION_CONTROL_C2S1	Extended Clause Action Control Registers C2S1
03324	03327	EXTENDED_CLAUSE_ACTION_CONTROL_C3S1	Extended Clause Action Control Registers C3S1
03330	03333	EXTENDED_CLAUSE_ACTION_CONTROL_C0S2	Extended Clause Action Control Registers C0S2
03334	03337	EXTENDED_CLAUSE_ACTION_CONTROL_C1S2	Extended Clause Action Control Registers C1S2
03338	0333B	EXTENDED_CLAUSE_ACTION_CONTROL_C2S2	Extended Clause Action Control Registers C2S2
0333C	0333F	EXTENDED_CLAUSE_ACTION_CONTROL_C3S2	Extended Clause Action Control Registers C3S2
03348	0334B	EXTENDED_CLAUSE_ACTION_CONTROL_C0S3	Extended Clause Action Control Registers C0S3
0334C	0334F	EXTENDED_CLAUSE_ACTION_CONTROL_C1S3	Extended Clause Action Control Registers C1S3
03350	03353	EXTENDED_CLAUSE_ACTION_CONTROL_C2S	Extended Clause Action Control Registers C2S3

CTS Registers			
Offset Start	Offset End	Symbol	Register Name/Function
		3	
03354	03357	EXTENDED_CLAUSE_ACTION_CONTROL_C3S3	Extended Clause Action Control Registers C3S3

13.7.2 CLAUSE_EVENT_MODE_C0S0: Clause Event Mode Register C0S0

CSR Register Name: CLAUSE_EVENT_MODE_C0S0: Clause Event Mode Register C0S0				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 03000	**Offset End:** 03003
Bits	**Access**	**Default**	**Label**	**Bit Description**
28	RW	0	OP_TYPE	Boolean Operation Type control for this clause 1 = OR together all enabled match sources 0 = AND together all enabled match sources
24	RW	0	LCT0_MATCH_MODE	Match Mode control for the Large 45-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
21	RW	0	SCT1_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 1 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
20	RW	0	SCT0_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
17:4	RW	0	EVENT_IN_MODE	**Mode control for each Event_In[15:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0

CSR Register Name: CLAUSE_EVENT_MODE_C0S0: Clause Event Mode Register C0S0					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 03000	**Offset End:** 03003
Bits	**Access**	**Default**	**Label**	**Bit Description**	
3:0	RW	0	SIGNAL_IN_MODE	**Mode control for each Signal_In[3:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0	

13.7.3 CLAUSE_EVENT_MODE_C1S0: Clause Event Mode Register C1S0

CSR Register Name: CLAUSE_EVENT_MODE_C1S0: Clause Event Mode Register C1S0					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 03004	**Offset End:** 03007
Bits	**Access**	**Default**	**Label**	**Bit Description**	
28	RW	0	OP_TYPE	Boolean Operation Type control for this clause 1 = OR together all enabled match sources 0 = AND together all enabled match sources	
24	RW	0	LCT0_MATCH_MODE	Match Mode control for the Large 45-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched	
21	RW	0	SCT1_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 1 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched	
20	RW	0	SCT0_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched	

CSR Register Name: CLAUSE_EVENT_MODE_C1S0: Clause Event Mode Register C1S0				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 03004	**Offset End:** 03007

Bits	Access	Default	Label	Bit Description
17:4	RW	0	EVENT_IN_MODE	**Mode control for each Event_In[15:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0
3:0	RW	0	SIGNAL_IN_MODE	**Mode control for each Signal_In[3:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0

13.7.4 CLAUSE_EVENT_MODE_C2S0: Clause Event Mode Register C2S0

CSR Register Name: CLAUSE_EVENT_MODE_C2S0: Clause Event Mode Register C2S0			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 03008	**Offset End:** 0300B

Bits	Access	Default	Label	Bit Description
28	RW	0	OP_TYPE	Boolean Operation Type control for this clause 1 = OR together all enabled match sources 0 = AND together all enabled match sources
24	RW	0	LCT0_MATCH_MODE	Match Mode control for the Large 45-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
21	RW	0	SCT1_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 1 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched

CSR Register Name: CLAUSE_EVENT_MODE_C2S0: Clause Event Mode Register C2S0					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 03008	**Offset End:** 0300B
Bits	**Access**	**Default**	**Label**	**Bit Description**	
20	RW	0	SCT0_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched	
17:4	RW	0	EVENT_IN_MODE	**Mode control for each Event_In[15:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0	
3:0	RW	0	SIGNAL_IN_MODE	**Mode control for each Signal_In[3:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0	

13.7.5 CLAUSE_EVENT_MODE_C3S0: Clause Event Mode Register C3S0

CSR Register Name: CLAUSE_EVENT_MODE_C3S0: Clause Event Mode Register C3S0					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 0300C	**Offset End:** 0300F
Bits	**Access**	**Default**	**Label**	**Bit Description**	
28	RW	0	OP_TYPE	Boolean Operation Type control for this clause 1 = OR together all enabled match sources 0 = AND together all enabled match sources	
24	RW	0	LCT0_MATCH_MODE	Match Mode control for the Large 45-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched	

CSR Register Name: CLAUSE_EVENT_MODE_C3S0: Clause Event Mode Register C3S0				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 0300C **Offset End:** 0300F
Bits	**Access**	**Default**	**Label**	**Bit Description**
21	RW	0	SCT1_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 1 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
20	RW	0	SCT0_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
17:4	RW	0	EVENT_IN_MODE	**Mode control for each Event_In[15:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0
3:0	RW	0	SIGNAL_IN_MODE	**Mode control for each Signal_In[3:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0

13.7.6 CLAUSE_EVENT_MODE_C0S1: Clause Event Mode Register C0S1

CSR Register Name: CLAUSE_EVENT_MODE_C0S1: Clause Event Mode Register C0S1				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 03018 **Offset End:** 0301B
Bits	**Access**	**Default**	**Label**	**Bit Description**
28	RW	0	OP_TYPE	Boolean Operation Type control for this clause 1 = OR together all enabled match sources 0 = AND together all enabled match sources

CSR Register Name: CLAUSE_EVENT_MODE_C0S1: Clause Event Mode Register C0S1				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 03018	**Offset End:** 0301B
Bits	**Access**	**Default**	**Label**	**Bit Description**
24	RW	0	LCT0_MATCH_MODE	Match Mode control for the Large 45-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
21	RW	0	SCT1_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 1 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
20	RW	0	SCT0_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
17:4	RW	0	EVENT_IN_MODE	**Mode control for each Event_In[15:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0
3:0	RW	0	SIGNAL_IN_MODE	**Mode control for each Signal_In[3:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0

13.7.7 CLAUSE_EVENT_MODE_C1S1: Clause Event Mode Register C1S1

CSR Register Name: CLAUSE_EVENT_MODE_C1S1: Clause Event Mode Register C1S1			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 0301C	**Offset End:** 0301F

Bits	Access	Default	Label	Bit Description
28	RW	0	OP_TYPE	Boolean Operation Type control for this clause 1 = OR together all enabled match sources 0 = AND together all enabled match sources
24	RW	0	LCT0_MATCH_MODE	Match Mode control for the Large 45-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
21	RW	0	SCT1_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 1 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
20	RW	0	SCT0_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
17:4	RW	0	EVENT_IN_MODE	**Mode control for each Event_In[15:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0
3:0	RW	0	SIGNAL_IN_MODE	**Mode control for each Signal_In[3:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0

13.7.8 CLAUSE_EVENT_MODE_C2S1: Clause Event Mode Register C2S1

CSR Register Name: **CLAUSE_EVENT_MODE_C2S1**: Clause Event Mode Register C2S1			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 03020	**Offset End:** 03023

Bits	Access	Default	Label	Bit Description
28	RW	0	OP_TYPE	Boolean Operation Type control for this clause 1 = OR together all enabled match sources 0 = AND together all enabled match sources
24	RW	0	LCT0_MATCH_MODE	Match Mode control for the Large 45-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
21	RW	0	SCT1_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 1 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
20	RW	0	SCT0_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
17:4	RW	0	EVENT_IN_MODE	**Mode control for each Event_In[15:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0
3:0	RW	0	SIGNAL_IN_MODE	**Mode control for each Signal_In[3:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0

13.7.9 CLAUSE_EVENT_MODE_C3S1: Clause Event Mode Register C3S1

CSR Register Name: CLAUSE_EVENT_MODE_C3S1: Clause Event Mode Register C3S1			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 03024	**Offset End:** 03027

Bits	Access	Default	Label	Bit Description
28	RW	0	OP_TYPE	Boolean Operation Type control for this clause 1 = OR together all enabled match sources 0 = AND together all enabled match sources
24	RW	0	LCT0_MATCH_MODE	Match Mode control for the Large 45-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
21	RW	0	SCT1_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 1 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
20	RW	0	SCT0_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
17:4	RW	0	EVENT_IN_MODE	**Mode control for each Event_In[15:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0
3:0	RW	0	SIGNAL_IN_MODE	**Mode control for each Signal_In[3:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0

13.7.10 CLAUSE_EVENT_MODE_C0S2: Clause Event Mode Register C0S2

CSR Register Name: **CLAUSE_EVENT_MODE_C0S2**: Clause Event Mode Register C0S2			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 03030	**Offset End:** 03033

Bits	Access	Default	Label	Bit Description
28	RW	0	OP_TYPE	Boolean Operation Type control for this clause 1 = OR together all enabled match sources 0 = AND together all enabled match sources
24	RW	0	LCT0_MATCH_MODE	Match Mode control for the Large 45-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
21	RW	0	SCT1_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 1 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
20	RW	0	SCT0_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
17:4	RW	0	EVENT_IN_MODE	**Mode control for each Event_In[15:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0
3:0	RW	0	SIGNAL_IN_MODE	**Mode control for each Signal_In[3:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0

13.7.11 CLAUSE_EVENT_MODE_C1S2: Clause Event Mode Register C1S2

CSR Register Name: CLAUSE_EVENT_MODE_C1S2: Clause Event Mode Register C1S2			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 03034	**Offset End:** 03037

Bits	Access	Default	Label	Bit Description
28	RW	0	OP_TYPE	Boolean Operation Type control for this clause 1 = OR together all enabled match sources 0 = AND together all enabled match sources
24	RW	0	LCT0_MATCH_MODE	Match Mode control for the Large 45-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
21	RW	0	SCT1_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 1 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
20	RW	0	SCT0_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
17:4	RW	0	EVENT_IN_MODE	**Mode control for each Event_In[15:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0
3:0	RW	0	SIGNAL_IN_MODE	**Mode control for each Signal_In[3:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0

13.7.12 CLAUSE_EVENT_MODE_C2S2: Clause Event Mode Register C2S2

CSR Register Name: CLAUSE_EVENT_MODE_C2S2: Clause Event Mode Register C2S2			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 03038	**Offset End:** 0303B

Bits	Access	Default	Label	Bit Description
28	RW	0	OP_TYPE	Boolean Operation Type control for this clause 1 = OR together all enabled match sources 0 = AND together all enabled match sources
24	RW	0	LCT0_MATCH_MODE	Match Mode control for the Large 45-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
21	RW	0	SCT1_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 1 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
20	RW	0	SCT0_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
17:4	RW	0	EVENT_IN_MODE	**Mode control for each Event_In[15:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0
3:0	RW	0	SIGNAL_IN_MODE	**Mode control for each Signal_In[3:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0

13.7.13 CLAUSE_EVENT_MODE_C3S2: Clause Event Mode Register C3S2

CSR Register Name: **CLAUSE_EVENT_MODE_C3S2**: Clause Event Mode Register C3S2			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 0303C	**Offset End:** 0303F

Bits	Access	Default	Label	Bit Description
28	RW	0	OP_TYPE	Boolean Operation Type control for this clause 1 = OR together all enabled match sources 0 = AND together all enabled match sources
24	RW	0	LCT0_MATCH_MODE	Match Mode control for the Large 45-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
21	RW	0	SCT1_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 1 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
20	RW	0	SCT0_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
17:4	RW	0	EVENT_IN_MODE	**Mode control for each Event_In[15:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0
3:0	RW	0	SIGNAL_IN_MODE	**Mode control for each Signal_In[3:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0

13.7.14 CLAUSE_EVENT_MODE_C0S3: Clause Event Mode Register C0S3

CSR Register Name: CLAUSE_EVENT_MODE_C0S3: Clause Event Mode Register C0S3			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 03048	**Offset End:** 0304B

Bits	Access	Default	Label	Bit Description
28	RW	0	OP_TYPE	Boolean Operation Type control for this clause 1 = OR together all enabled match sources 0 = AND together all enabled match sources
24	RW	0	LCT0_MATCH_MODE	Match Mode control for the Large 45-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
21	RW	0	SCT1_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 1 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
20	RW	0	SCT0_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
17:4	RW	0	EVENT_IN_MODE	**Mode control for each Event_In[15:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0
3:0	RW	0	SIGNAL_IN_MODE	**Mode control for each Signal_In[3:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0

13.7.15 CLAUSE_EVENT_MODE_C1S3: Clause Event Mode Register C1S3

CSR Register Name: **CLAUSE_EVENT_MODE_C1S3**: Clause Event Mode Register C1S3			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 0304C	**Offset End:** 0304F

Bits	Access	Default	Label	Bit Description
28	RW	0	OP_TYPE	Boolean Operation Type control for this clause 1 = OR together all enabled match sources 0 = AND together all enabled match sources
24	RW	0	LCT0_MATCH_MODE	Match Mode control for the Large 45-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
21	RW	0	SCT1_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 1 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
20	RW	0	SCT0_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
17:4	RW	0	EVENT_IN_MODE	**Mode control for each Event_In[15:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0
3:0	RW	0	SIGNAL_IN_MODE	**Mode control for each Signal_In[3:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0

13.7.16 CLAUSE_EVENT_MODE_C2S3: Clause Event Mode Register C2S3

CSR Register Name: **CLAUSE_EVENT_MODE_C2S3**: Clause Event Mode Register C2S3			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 03050	**Offset End:** 03053

Bits	Access	Default	Label	Bit Description
28	RW	0	OP_TYPE	Boolean Operation Type control for this clause 1 = OR together all enabled match sources 0 = AND together all enabled match sources
24	RW	0	LCT0_MATCH_MODE	Match Mode control for the Large 45-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
21	RW	0	SCT1_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 1 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
20	RW	0	SCT0_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
17:4	RW	0	EVENT_IN_MODE	**Mode control for each Event_In[15:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0
3:0	RW	0	SIGNAL_IN_MODE	**Mode control for each Signal_In[3:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0

13.7.17 CLAUSE_EVENT_MODE_C3S3: Clause Event Mode Register C3S3

CSR Register Name: CLAUSE_EVENT_MODE_C3S3: Clause Event Mode Register C3S3			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 03054	**Offset End:** 03057

Bits	Access	Default	Label	Bit Description
28	RW	0	OP_TYPE	Boolean Operation Type control for this clause 1 = OR together all enabled match sources 0 = AND together all enabled match sources
24	RW	0	LCT0_MATCH_MODE	Match Mode control for the Large 45-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
21	RW	0	SCT1_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 1 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
20	RW	0	SCT0_MATCH_MODE	Match Mode control for the Small 20-bit Counter/Timer 0 for this clause 1 = match on counter/timer match 0 = match on counter/timer not matched
17:4	RW	0	EVENT_IN_MODE	**Mode control for each Event_In[15:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0
3:0	RW	0	SIGNAL_IN_MODE	**Mode control for each Signal_In[3:**0] input for this clause 1 = match on a logical 1 0 = match on a logical 0

13.7.18 CLAUSE_EVENT_ENABLE_C0S0: Clause Event Enable Register C0S0

CSR Register Name: CLAUSE_EVENT_ENABLE_C0S0: Clause Event Enable Register C0S0			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 030C0	**Offset End:** 030C3

Bits	Access	Default	Label	Bit Description
31	RW	0	IF_ANYTHING	**When the If Anything clause is enabled for a clause it allows the user to set any actions independent of the event inputs.** Setting the If Anything Clause Event Enable bit will force a hit on this event enable. This guarantees an active event to a clause. The if anything clause is independent of the OP_Type for the clause. The Clause Event Mode for the If Anything event is not required and not implemented.
24	RW	0	LCT0_SELECT	Selects the Large 45-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
21	RW	0	SCT1_SELECT	Selects the Small 20-bit Counter/Timer 1 as match event for this clause 1 = enable; 0 = disable
20	RW	0	SCT0_SELECT	Selects the Small 20-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
17:4	RW	0	EVENT_IN_SELECT	**Selects each Event_In[15:**0] as match events for this clause 1 = enable; 0 = disable
3:0	RW	0	SIGNAL_IN_SELECT	**Selects each Signal_In[3:**0] as match events for this clause 1 = enable; 0 = disable

13.7.19 CLAUSE_EVENT_ENABLE_C1S0: Clause Event Enable Register C1S0

CSR Register Name: CLAUSE_EVENT_ENABLE_C1S0: Clause Event Enable Register C1S0			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 030C4	**Offset End:** 030C7

Bits	Access	Default	Label	Bit Description
31	RW	0	IF_ANYTHING	**When the If Anything clause is enabled for a clause it allows the user to set any actions independent of the event inputs.** Setting the If Anything Clause Event Enable bit will force a hit on this event enable. This guarantees an active event to a clause. The if anything clause is independent of the OP_Type for the clause. The Clause Event Mode for the If Anything event is not required and not implemented.
24	RW	0	LCT0_SELECT	Selects the Large 45-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
21	RW	0	SCT1_SELECT	Selects the Small 20-bit Counter/Timer 1 as match event for this clause 1 = enable; 0 = disable
20	RW	0	SCT0_SELECT	Selects the Small 20-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
17:4	RW	0	EVENT_IN_SELECT	**Selects each Event_In[15:**0] as match events for this clause 1 = enable; 0 = disable
3:0	RW	0	SIGNAL_IN_SELECT	**Selects each Signal_In[3:**0] as match events for this clause 1 = enable; 0 = disable

13.7.20 CLAUSE_EVENT_ENABLE_C2S0: Clause Event Enable Register C2S0

CSR Register Name: CLAUSE_EVENT_ENABLE_C2S0: Clause Event Enable Register C2S0				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 030C8	**Offset End:** 030CB
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	0	IF_ANYTHING	**When the If Anything clause is enabled for a clause it allows the user to set any actions independent of the event inputs.** Setting the If Anything Clause Event Enable bit will force a hit on this event enable. This guarantees an active event to a clause. The if anything clause is independent of the OP_Type for the clause. The Clause Event Mode for the If Anything event is not required and not implemented.
24	RW	0	LCT0_SELECT	Selects the Large 45-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
21	RW	0	SCT1_SELECT	Selects the Small 20-bit Counter/Timer 1 as match event for this clause 1 = enable; 0 = disable
20	RW	0	SCT0_SELECT	Selects the Small 20-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
17:4	RW	0	EVENT_IN_SELECT	**Selects each Event_In[15:**0] as match events for this clause 1 = enable; 0 = disable
3:0	RW	0	SIGNAL_IN_SELECT	**Selects each Signal_In[3:**0] as match events for this clause 1 = enable; 0 = disable

13.7.21 CLAUSE_EVENT_ENABLE_C3S0: Clause Event Enable Register C3S0

CSR Register Name: CLAUSE_EVENT_ENABLE_C3S0: Clause Event Enable Register C3S0

Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 030CC	**Offset End:** 030CF

Bits	Access	Default	Label	Bit Description
31	RW	0	IF_ANYTHING	**When the If Anything clause is enabled for a clause it allows the user to set any actions independent of the event inputs.** Setting the If Anything Clause Event Enable bit will force a hit on this event enable. This guarantees an active event to a clause. The if anything clause is independent of the OP_Type for the clause. The Clause Event Mode for the If Anything event is not required and not implemented.
24	RW	0	LCT0_SELECT	Selects the Large 45-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
21	RW	0	SCT1_SELECT	Selects the Small 20-bit Counter/Timer 1 as match event for this clause 1 = enable; 0 = disable
20	RW	0	SCT0_SELECT	Selects the Small 20-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
17:4	RW	0	EVENT_IN_SELECT	**Selects each Event_In[15:**0] as match events for this clause 1 = enable; 0 = disable
3:0	RW	0	SIGNAL_IN_SELECT	**Selects each Signal_In[3:**0] as match events for this clause 1 = enable; 0 = disable

13.7.22 CLAUSE_EVENT_ENABLE_C0S1: Clause Event Enable Register C0S1

CSR Register Name: CLAUSE_EVENT_ENABLE_C0S1: Clause Event Enable Register C0S1				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 030D8	**Offset End:** 030DB
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	0	IF_ANYTHING	**When the If Anything clause is enabled for a clause it allows the user to set any actions independent of the event inputs.** Setting the If Anything Clause Event Enable bit will force a hit on this event enable. This guarantees an active event to a clause. The if anything clause is independent of the OP_Type for the clause. The Clause Event Mode for the If Anything event is not required and not implemented.
24	RW	0	LCT0_SELECT	Selects the Large 45-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
21	RW	0	SCT1_SELECT	Selects the Small 20-bit Counter/Timer 1 as match event for this clause 1 = enable; 0 = disable
20	RW	0	SCT0_SELECT	Selects the Small 20-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
17:4	RW	0	EVENT_IN_SELECT	**Selects each Event_In[15:**0] as match events for this clause 1 = enable; 0 = disable
3:0	RW	0	SIGNAL_IN_SELECT	**Selects each Signal_In[3:**0] as match events for this clause 1 = enable; 0 = disable

13.7.23 CLAUSE_EVENT_ENABLE_C1S1: Clause Event Enable Register C1S1

CSR Register Name: CLAUSE_EVENT_ENABLE_C1S1: Clause Event Enable Register C1S1			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 030DC	**Offset End:** 030DF

Bits	Access	Default	Label	Bit Description
31	RW	0	IF_ANYTHING	**When the If Anything clause is enabled for a clause it allows the user to set any actions independent of the event inputs.** Setting the If Anything Clause Event Enable bit will force a hit on this event enable. This guarantees an active event to a clause. The if anything clause is independent of the OP_Type for the clause. The Clause Event Mode for the If Anything event is not required and not implemented.
24	RW	0	LCT0_SELECT	Selects the Large 45-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
21	RW	0	SCT1_SELECT	Selects the Small 20-bit Counter/Timer 1 as match event for this clause 1 = enable; 0 = disable
20	RW	0	SCT0_SELECT	Selects the Small 20-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
17:4	RW	0	EVENT_IN_SELECT	**Selects each Event_In[15:**0] as match events for this clause 1 = enable; 0 = disable
3:0	RW	0	SIGNAL_IN_SELECT	**Selects each Signal_In[3:**0] as match events for this clause 1 = enable; 0 = disable

13.7.24 CLAUSE_EVENT_ENABLE_C2S1: Clause Event Enable Register C2S1

CSR Register Name: CLAUSE_EVENT_ENABLE_C2S1: Clause Event Enable Register C2S1				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 030E0	**Offset End:** 030E3

Bits	Access	Default	Label	Bit Description
31	RW	0	IF_ANYTHING	**When the If Anything clause is enabled for a clause it allows the user to set any actions independent of the event inputs.** Setting the If Anything Clause Event Enable bit will force a hit on this event enable. This guarantees an active event to a clause. The if anything clause is independent of the OP_Type for the clause. The Clause Event Mode for the If Anything event is not required and not implemented.
24	RW	0	LCT0_SELECT	Selects the Large 45-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
21	RW	0	SCT1_SELECT	Selects the Small 20-bit Counter/Timer 1 as match event for this clause 1 = enable; 0 = disable
20	RW	0	SCT0_SELECT	Selects the Small 20-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
17:4	RW	0	EVENT_IN_SELECT	**Selects each Event_In[15:**0] as match events for this clause 1 = enable; 0 = disable
3:0	RW	0	SIGNAL_IN_SELECT	**Selects each Signal_In[3:**0] as match events for this clause 1 = enable; 0 = disable

13.7.25 CLAUSE_EVENT_ENABLE_C3S1: Clause Event Enable Register C3S1

CSR Register Name: **CLAUSE_EVENT_ENABLE_C3S1**: Clause Event Enable Register C3S1			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 030E4	**Offset End:** 030E7

Bits	Access	Default	Label	Bit Description
31	RW	0	IF_ANYTHING	**When the If Anything clause is enabled for a clause it allows the user to set any actions independent of the event inputs.** Setting the If Anything Clause Event Enable bit will force a hit on this event enable. This guarantees an active event to a clause. The if anything clause is independent of the OP_Type for the clause. The Clause Event Mode for the If Anything event is not required and not implemented.
24	RW	0	LCT0_SELECT	Selects the Large 45-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
21	RW	0	SCT1_SELECT	Selects the Small 20-bit Counter/Timer 1 as match event for this clause 1 = enable; 0 = disable
20	RW	0	SCT0_SELECT	Selects the Small 20-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
17:4	RW	0	EVENT_IN_SELECT	**Selects each Event_In[15:**0] as match events for this clause 1 = enable; 0 = disable
3:0	RW	0	SIGNAL_IN_SELECT	**Selects each Signal_In[3:**0] as match events for this clause 1 = enable; 0 = disable

13.7.26 CLAUSE_EVENT_ENABLE_C0S2: Clause Event Enable Register C0S2

CSR Register Name: CLAUSE_EVENT_ENABLE_C0S2: Clause Event Enable Register C0S2				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 030F0	**Offset End:** 030F3

Bits	Access	Default	Label	Bit Description
31	RW	0	IF_ANYTHING	**When the If Anything clause is enabled for a clause it allows the user to set any actions independent of the event inputs.** Setting the If Anything Clause Event Enable bit will force a hit on this event enable. This guarantees an active event to a clause. The if anything clause is independent of the OP_Type for the clause. The Clause Event Mode for the If Anything event is not required and not implemented.
24	RW	0	LCT0_SELECT	Selects the Large 45-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
21	RW	0	SCT1_SELECT	Selects the Small 20-bit Counter/Timer 1 as match event for this clause 1 = enable; 0 = disable
20	RW	0	SCT0_SELECT	Selects the Small 20-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
17:4	RW	0	EVENT_IN_SELECT	**Selects each Event_In[15:**0] as match events for this clause 1 = enable; 0 = disable
3:0	RW	0	SIGNAL_IN_SELECT	**Selects each Signal_In[3:**0] as match events for this clause 1 = enable; 0 = disable

13.7.27 CLAUSE_EVENT_ENABLE_C1S2: Clause Event Enable Register C1S2

CSR Register Name: CLAUSE_EVENT_ENABLE_C1S2: Clause Event Enable Register C1S2				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 030F4 **Offset End:** 030F7
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	0	IF_ANYTHING	**When the If Anything clause is enabled for a clause it allows the user to set any actions independent of the event inputs.** Setting the If Anything Clause Event Enable bit will force a hit on this event enable. This guarantees an active event to a clause. The if anything clause is independent of the OP_Type for the clause. The Clause Event Mode for the If Anything event is not required and not implemented.
24	RW	0	LCT0_SELECT	Selects the Large 45-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
21	RW	0	SCT1_SELECT	Selects the Small 20-bit Counter/Timer 1 as match event for this clause 1 = enable; 0 = disable
20	RW	0	SCT0_SELECT	Selects the Small 20-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
17:4	RW	0	EVENT_IN_SELECT	**Selects each Event_In[15:**0] as match events for this clause 1 = enable; 0 = disable
3:0	RW	0	SIGNAL_IN_SELECT	**Selects each Signal_In[3:**0] as match events for this clause 1 = enable; 0 = disable

13.7.28 CLAUSE_EVENT_ENABLE_C2S2: Clause Event Enable Register C2S2

<table>
<tr><td colspan="5"><b>CSR Register Name:</b> <b>CLAUSE_EVENT_ENABLE_C2S2</b>: Clause Event Enable Register C2S2</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b>
030F8
<b>Offset End:</b>
030FB</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th>Bit Description</th></tr>
<tr><td>31</td><td>RW</td><td>0</td><td>IF_ANYTHING</td><td><b>When the If Anything clause is enabled for a clause it allows the user to set any actions independent of the event inputs.</b> Setting the If Anything Clause Event Enable bit will force a hit on this event enable. This guarantees an active event to a clause. The if anything clause is independent of the OP_Type for the clause. The Clause Event Mode for the If Anything event is not required and not implemented.</td></tr>
<tr><td>24</td><td>RW</td><td>0</td><td>LCT0_SELECT</td><td>Selects the Large 45-bit Counter/Timer 0 as match event for this clause
1 = enable;
0 = disable</td></tr>
<tr><td>21</td><td>RW</td><td>0</td><td>SCT1_SELECT</td><td>Selects the Small 20-bit Counter/Timer 1 as match event for this clause
1 = enable;
0 = disable</td></tr>
<tr><td>20</td><td>RW</td><td>0</td><td>SCT0_SELECT</td><td>Selects the Small 20-bit Counter/Timer 0 as match event for this clause
1 = enable;
0 = disable</td></tr>
<tr><td>17:4</td><td>RW</td><td>0</td><td>EVENT_IN_SELECT</td><td><b>Selects each Event_In[15:</b>0] as match events for this clause
1 = enable;
0 = disable</td></tr>
<tr><td>3:0</td><td>RW</td><td>0</td><td>SIGNAL_IN_SELECT</td><td><b>Selects each Signal_In[3:</b>0] as match events for this clause
1 = enable;
0 = disable</td></tr>
</table>

13.7.29 CLAUSE_EVENT_ENABLE_C3S2: Clause Event Enable Register C3S2

CSR Register Name: CLAUSE_EVENT_ENABLE_C3S2: Clause Event Enable Register C3S2				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 030FC	**Offset End:** 030FF
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	0	IF_ANYTHING	**When the If Anything clause is enabled for a clause it allows the user to set any actions independent of the event inputs.** Setting the If Anything Clause Event Enable bit will force a hit on this event enable. This guarantees an active event to a clause. The if anything clause is independent of the OP_Type for the clause. The Clause Event Mode for the If Anything event is not required and not implemented.
24	RW	0	LCT0_SELECT	Selects the Large 45-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
21	RW	0	SCT1_SELECT	Selects the Small 20-bit Counter/Timer 1 as match event for this clause 1 = enable; 0 = disable
20	RW	0	SCT0_SELECT	Selects the Small 20-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
17:4	RW	0	EVENT_IN_SELECT	**Selects each Event_In[15:**0] as match events for this clause 1 = enable; 0 = disable
3:0	RW	0	SIGNAL_IN_SELECT	**Selects each Signal_In[3:**0] as match events for this clause 1 = enable; 0 = disable

13.7.30 CLAUSE_EVENT_ENABLE_C0S3: Clause Event Enable Register C0S3

CSR Register Name: CLAUSE_EVENT_ENABLE_C0S3: Clause Event Enable Register C0S3				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 03108	**Offset End:** 0310B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	0	IF_ANYTHING	**When the If Anything clause is enabled for a clause it allows the user to set any actions independent of the event inputs.** Setting the If Anything Clause Event Enable bit will force a hit on this event enable. This guarantees an active event to a clause. The if anything clause is independent of the OP_Type for the clause. The Clause Event Mode for the If Anything event is not required and not implemented.
24	RW	0	LCT0_SELECT	Selects the Large 45-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
21	RW	0	SCT1_SELECT	Selects the Small 20-bit Counter/Timer 1 as match event for this clause 1 = enable; 0 = disable
20	RW	0	SCT0_SELECT	Selects the Small 20-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
17:4	RW	0	EVENT_IN_SELECT	**Selects each Event_In[15:**0] as match events for this clause 1 = enable; 0 = disable
3:0	RW	0	SIGNAL_IN_SELECT	**Selects each Signal_In[3:**0] as match events for this clause 1 = enable; 0 = disable

13.7.31 CLAUSE_EVENT_ENABLE_C1S3: Clause Event Enable Register C1S3

CSR Register Name: CLAUSE_EVENT_ENABLE_C1S3: Clause Event Enable Register C1S3				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 0310C	**Offset End:** 0310F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	0	IF_ANYTHING	**When the If Anything clause is enabled for a clause it allows the user to set any actions independent of the event inputs.** Setting the If Anything Clause Event Enable bit will force a hit on this event enable. This guarantees an active event to a clause. The if anything clause is independent of the OP_Type for the clause. The Clause Event Mode for the If Anything event is not required and not implemented.
24	RW	0	LCT0_SELECT	Selects the Large 45-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
21	RW	0	SCT1_SELECT	Selects the Small 20-bit Counter/Timer 1 as match event for this clause 1 = enable; 0 = disable
20	RW	0	SCT0_SELECT	Selects the Small 20-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
17:4	RW	0	EVENT_IN_SELECT	**Selects each Event_In[15:**0] as match events for this clause 1 = enable; 0 = disable
3:0	RW	0	SIGNAL_IN_SELECT	**Selects each Signal_In[3:**0] as match events for this clause 1 = enable; 0 = disable

13.7.32 CLAUSE_EVENT_ENABLE_C2S3: Clause Event Enable Register C2S3

CSR Register Name: CLAUSE_EVENT_ENABLE_C2S3: Clause Event Enable Register C2S3				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 03110	**Offset End:** 03113
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	0	IF_ANYTHING	**When the If Anything clause is enabled for a clause it allows the user to set any actions independent of the event inputs.** Setting the If Anything Clause Event Enable bit will force a hit on this event enable. This guarantees an active event to a clause. The if anything clause is independent of the OP_Type for the clause. The Clause Event Mode for the If Anything event is not required and not implemented.
24	RW	0	LCT0_SELECT	Selects the Large 45-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
21	RW	0	SCT1_SELECT	Selects the Small 20-bit Counter/Timer 1 as match event for this clause 1 = enable; 0 = disable
20	RW	0	SCT0_SELECT	Selects the Small 20-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
17:4	RW	0	EVENT_IN_SELECT	**Selects each Event_In[15:**0] as match events for this clause 1 = enable; 0 = disable
3:0	RW	0	SIGNAL_IN_SELECT	**Selects each Signal_In[3:**0] as match events for this clause 1 = enable; 0 = disable

13.7.33 CLAUSE_EVENT_ENABLE_C3S3: Clause Event Enable Register C3S3

CSR Register Name: CLAUSE_EVENT_ENABLE_C3S3: Clause Event Enable Register C3S3				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 03114	**Offset End:** 03117
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	0	IF_ANYTHING	**When the If Anything clause is enabled for a clause it allows the user to set any actions independent of the event inputs.** Setting the If Anything Clause Event Enable bit will force a hit on this event enable. This guarantees an active event to a clause. The if anything clause is independent of the OP_Type for the clause. The Clause Event Mode for the If Anything event is not required and not implemented.
24	RW	0	LCT0_SELECT	Selects the Large 45-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
21	RW	0	SCT1_SELECT	Selects the Small 20-bit Counter/Timer 1 as match event for this clause 1 = enable; 0 = disable
20	RW	0	SCT0_SELECT	Selects the Small 20-bit Counter/Timer 0 as match event for this clause 1 = enable; 0 = disable
17:4	RW	0	EVENT_IN_SELECT	**Selects each Event_In[15:**0] as match events for this clause 1 = enable; 0 = disable
3:0	RW	0	SIGNAL_IN_SELECT	**Selects each Signal_In[3:**0] as match events for this clause 1 = enable; 0 = disable

13.7.34 CLAUSE_ACTION_CONTROL_C0S0: Clause Action Control Registers C0S0

<table>
<tr><td colspan="6"><b>CSR Register Name:</b> <b>CLAUSE_ACTION_CONTROL_C0S0</b>: Clause Action Control Registers C0S0</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b>
03180</td><td><b>Offset End:</b>
03183</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>31</td><td>RW</td><td>0</td><td>SET_SIGNAL_VALUE</td><td colspan="2">Set_Signal_Value is an action qualifier for the Set_Signal_Out[3:0] actions for each clause of each state if Signal_Out_Mode[1:0] is set to 11. In this mode, the driven level of the Signal_Out output wire will be set or cleared to the value of the Set_Signal_Value action qualifier bit if the corresponding Set_Signal_Out[3:0] action is taken.</td></tr>
<tr><td>30:27</td><td>RW</td><td>0</td><td>SET_STATE</td><td colspan="2"><b>Go to state m as result of clause/state event match, where m is the contents of this register.</b> The default reset value for each clause will be the IDLE state. The user can set the next state to any state to build a desired sequencer flow.</td></tr>
<tr><td>23</td><td>RW</td><td>0</td><td>LCT0_CLEAR</td><td colspan="2"><b>Clear Large 45-bit Counter/Timer0 on clause/state event match.</b> When in Timer mode, a 1 on this action clears the timer and halts its operation. LCT0_start_inc is needed to restart the timer, but can be set simultaneously with LCT0_clear. When in Counter mode, a 1 on this action will clear the counter. When in Peak Timer mode, a 1 on this action clears the timer, and halts its operation while simultaneously updating the Peak Timer Status register if the LCT exceeds the current contents of the Peak Timer Status register.</td></tr>
<tr><td>20</td><td>RW</td><td>0</td><td>SCT1_CLEAR</td><td colspan="2"><b>Clear Small 20-bit Counter/Timer1 on clause/state event match.</b> When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT1_start_inc is needed to restart the timer, but can be set simultaneously with SCT1_clear. When in Counter mode, a 1 on this action will clear the counter.</td></tr>
</table>

CSR Register Name: CLAUSE_ACTION_CONTROL_C0S0: Clause Action Control Registers C0S0				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 03180	**Offset End:** 03183
Bits	**Access**	**Default**	**Label**	**Bit Description**
19	RW	0	SCT0_CLEAR	**Clear Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT0_start_inc is needed to restart the timer, but can be set simultaneously with SCT0_clear. When in Counter mode, a 1 on this action will clear the counter.
15	RW	0	LCT0_STOP	**Stop Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will stop the timer at its current value. This enables starting and stopping without losing timer value. When in Counter mode, this action is undefined.
11	RW	0	LCT0_START_INC	**Enable Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer or Peak Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
8	RW	0	SCT1_START_INC	**Enable Small 20-bit Counter/Timer1 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
7	RW	0	SCT0_START_INC	**Enable Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
6	RW	0	STOP_STORAGE	**Clear to logical 0 the Stor_Qual0 output, indicating to IOT storage to stop or suspend trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will clear CaptureDone.
5	RW	0	START_STORAGE	**Set to logical 1 the Stor_Qual0 output, indicating to IOT storage to begin or resume trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will set CaptureDone.

<table>
<tr><td colspan="6"><b>CSR Register Name:</b> <b>CLAUSE_ACTION_CONTROL_C0S0</b>: Clause Action Control Registers C0S0</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 03180</td><td><b>Offset End:</b> 03183</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>4</td><td>RW</td><td>0</td><td>SET_TRIG_OUT</td><td colspan="2"><b>Pulse the Trig_Out output.</b> All counter/ timers will continue to operate until a transition to the IDLE state, which may or may not be coincident with set_trig_out. This allows for continued operation past the Trig_Out event to control things like Stor_Qual or counting of events or time beyond a trigger for example.</td></tr>
<tr><td>3:0</td><td>RW</td><td>0</td><td>SET_SIGNAL_OUT</td><td colspan="2">If Signal_Out_Mode[1:0] = 00 pulsed mode, then this action will pulse the corresponding Signal_Out[3:0] output for one cycle as result of an event match. If Signal_Out_Mode[1:0] = 01 level mode, then this action will change invert the level of the corresponding Signal_Out[3:0] output as a result of an event match. If Signal_Out_Mode[1:0] = 11 signal out qualifier mode, then this action will set the corresponding Signal_Out[3:0] output to the value of Set_Signal_Out action qualifier. This is a new mode for CTS 2.0.</td></tr>
</table>

13.7.35 CLAUSE_ACTION_CONTROL_C1S0: Clause Action Control Registers C1S0

<table>
<tr><td colspan="4"><b>CSR Register Name:</b> <b>CLAUSE_ACTION_CONTROL_C1S0</b>: Clause Action Control Registers C1S0</td></tr>
<tr><td><b>Bar:</b> CSR_MTB_BAR</td><td><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 03184</td><td><b>Offset End:</b> 03187</td></tr>
</table>

Bits	Access	Default	Label	Bit Description
31	RW	0	SET_SIGNAL_VALUE	Set_Signal_Value is an action qualifier for the Set_Signal_Out[3:0] actions for each clause of each state if Signal_Out_Mode[1:0] is set to 11. In this mode, the driven level of the Signal_Out output wire will be set or cleared to the value of the Set_Signal_Value action qualifier bit if the corresponding Set_Signal_Out[3:0] action is taken. The Set_Signal_Value is only used in the signal out qualifier mode of operation.
30:27	RW	0	SET_STATE	**Go to state m as result of clause/state event match, where m is the contents of this register.** The default reset value for each clause will be the IDLE state. The user can set the next state to any state to build a desired sequencer flow.
23	RW	0	LCT0_CLEAR	**Clear Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. LCT0_start_inc is needed to restart the timer, but can be set simultaneously with LCT0_clear. When in Counter mode, a 1 on this action will clear the counter. When in Peak Timer mode, a 1 on this action clears the timer, and halts its operation while simultaneously updating the Peak Timer Status register if the LCT exceeds the current contents of the Peak Timer Status register.
20	RW	0	SCT1_CLEAR	**Clear Small 20-bit Counter/Timer1 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT1_start_inc is needed to restart the timer, but can be set simultaneously with SCT1_clear. When in Counter mode, a 1 on this action will clear the counter.
19	RW	0	SCT0_CLEAR	**Clear Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT0_start_inc is needed to restart the timer, but can be set simultaneously with SCT0_clear. When in Counter mode, a 1 on this action will clear the counter.

CSR Register Name: CLAUSE_ACTION_CONTROL_C1S0: Clause Action Control Registers C1S0				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 03184	**Offset End:** 03187
Bits	**Access**	**Default**	**Label**	**Bit Description**
15	RW	0	LCT0_STOP	**Stop Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will stop the timer at its current value. This enables starting and stopping without losing timer value. When in Counter mode, this action is undefined.
11	RW	0	LCT0_START_INC	**Enable Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer or Peak Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
8	RW	0	SCT1_START_INC	**Enable Small 20-bit Counter/Timer1 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
7	RW	0	SCT0_START_INC	**Enable Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
6	RW	0	STOP_STORAGE	**Clear to logical 0 the Stor_Qual0 output, indicating to IOT storage to stop or suspend trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will clear CaptureDone.
5	RW	0	START_STORAGE	**Set to logical 1 the Stor_Qual0 output, indicating to IOT storage to begin or resume trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will set CaptureDone.
4	RW	0	SET_TRIG_OUT	**Pulse the Trig_Out output.** All counter/ timers will continue to operate until a transition to the IDLE state, which may or may not be coincident with set_trig_out. This allows for continued operation past the Trig_Out event to control things like Stor_Qual or counting of events or time beyond a trigger for example.

CSR Register Name: CLAUSE_ACTION_CONTROL_C1S0: Clause Action Control Registers C1S0				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 03184	**Offset End:** 03187
Bits	**Access**	**Default**	**Label**	**Bit Description**
3:0	RW	0	SET_SIGNAL_OUT	If Signal_Out_Mode[1:0] = 00 pulsed mode, then this action will pulse the corresponding Signal_Out[3:0] output for one cycle as result of an event match. If Signal_Out_Mode[1:0] = 01 level mode, then this action will change invert the level of the corresponding Signal_Out[3:0] output as a result of an event match. If Signal_Out_Mode[1:0] = 11 signal out qualifier mode, then this action will set the corresponding Signal_Out[3:0] output to the value of Set_Signal_Out action qualifier. This is a new mode for CTS 2.0.

13.7.36 CLAUSE_ACTION_CONTROL_C2S0: Clause Action Control Registers C2S0

CSR Register Name: CLAUSE_ACTION_CONTROL_C2S0: Clause Action Control Registers C2S0				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 03188	**Offset End:** 0318B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	0	SET_SIGNAL_VALUE	Set_Signal_Value is an action qualifier for the Set_Signal_Out[3:0] actions for each clause of each state if Signal_Out_Mode[1:0] is set to 11. In this mode, the driven level of the Signal_Out output wire will be set or cleared to the value of the Set_Signal_Value action qualifier bit if the corresponding Set_Signal_Out[3:0] action is taken. The Set_Signal_Value is only used in the signal out qualifier mode of operation.
30:27	RW	0	SET_STATE	**Go to state m as result of clause/state event match, where m is the contents of this register.** The default reset value for each clause will be the IDLE state. The user can set the next state to any state to build a desired sequencer flow.

CSR Register Name: CLAUSE_ACTION_CONTROL_C2S0: Clause Action Control Registers C2S0				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 03188 **Offset End:** 0318B
Bits	**Access**	**Default**	**Label**	**Bit Description**
23	RW	0	LCT0_CLEAR	**Clear Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. LCT0_start_inc is needed to restart the timer, but can be set simultaneously with LCT0_clear. When in Counter mode, a 1 on this action will clear the counter. When in Peak Timer mode, a 1 on this action clears the timer, and halts its operation while simultaneously updating the Peak Timer Status register if the LCT exceeds the current contents of the Peak Timer Status register.
20	RW	0	SCT1_CLEAR	**Clear Small 20-bit Counter/Timer1 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT1_start_inc is needed to restart the timer, but can be set simultaneously with SCT1_clear. When in Counter mode, a 1 on this action will clear the counter.
19	RW	0	SCT0_CLEAR	**Clear Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT0_start_inc is needed to restart the timer, but can be set simultaneously with SCT0_clear. When in Counter mode, a 1 on this action will clear the counter.
15	RW	0	LCT0_STOP	**Stop Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will stop the timer at its current value. This enables starting and stopping without losing timer value. When in Counter mode, this action is undefined.
11	RW	0	LCT0_START_INC	**Enable Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer or Peak Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
8	RW	0	SCT1_START_INC	**Enable Small 20-bit Counter/Timer1 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter

CSR Register Name: CLAUSE_ACTION_CONTROL_C2S0: Clause Action Control Registers C2S0			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 03188	**Offset End:** 0318B

Bits	Access	Default	Label	Bit Description
7	RW	0	SCT0_START_INC	**Enable Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
6	RW	0	STOP_STORAGE	**Clear to logical 0 the Stor_Qual0 output, indicating to IOT storage to stop or suspend trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will clear CaptureDone.
5	RW	0	START_STORAGE	**Set to logical 1 the Stor_Qual0 output, indicating to IOT storage to begin or resume trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will set CaptureDone.
4	RW	0	SET_TRIG_OUT	**Pulse the Trig_Out output.** All counter/ timers will continue to operate until a transition to the IDLE state, which may or may not be coincident with set_trig_out. This allows for continued operation past the Trig_Out event to control things like Stor_Qual or counting of events or time beyond a trigger for example.
3:0	RW	0	SET_SIGNAL_OUT	If Signal_Out_Mode[1:0] = 00 pulsed mode, then this action will pulse the corresponding Signal_Out[3:0] output for one cycle as result of an event match. If Signal_Out_Mode[1:0] = 01 level mode, then this action will change invert the level of the corresponding Signal_Out[3:0] output as a result of an event match. If Signal_Out_Mode[1:0] = 11 signal out qualifier mode, then this action will set the corresponding Signal_Out[3:0] output to the value of Set_Signal_Out action qualifier. This is a new mode for CTS 2.0.

13.7.37 CLAUSE_ACTION_CONTROL_C3S0: Clause Action Control Registers C3S0

CSR Register Name: CLAUSE_ACTION_CONTROL_C3S0: Clause Action Control Registers C3S0				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 0318C	**Offset End:** 0318F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	0	SET_SIGNAL_VALUE	Set_Signal_Value is an action qualifier for the Set_Signal_Out[3:0] actions for each clause of each state if Signal_Out_Mode[1:0] is set to 11. In this mode, the driven level of the Signal_Out output wire will be set or cleared to the value of the Set_Signal_Value action qualifier bit if the corresponding Set_Signal_Out[3:0] action is taken. The Set_Signal_Value is only used in the signal out qualifier mode of operation.
30:27	RW	0	SET_STATE	**Go to state m as result of clause/state event match, where m is the contents of this register.** The default reset value for each clause will be the IDLE state. The user can set the next state to any state to build a desired sequencer flow.
23	RW	0	LCT0_CLEAR	**Clear Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. LCT0_start_inc is needed to restart the timer, but can be set simultaneously with LCT0_clear. When in Counter mode, a 1 on this action will clear the counter. When in Peak Timer mode, a 1 on this action clears the timer, and halts its operation while simultaneously updating the Peak Timer Status register if the LCT exceeds the current contents of the Peak Timer Status register.
20	RW	0	SCT1_CLEAR	**Clear Small 20-bit Counter/Timer1 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT1_start_inc is needed to restart the timer, but can be set simultaneously with SCT1_clear. When in Counter mode, a 1 on this action will clear the counter.

CSR Register Name: CLAUSE_ACTION_CONTROL_C3S0: Clause Action Control Registers C3S0				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 0318C	**Offset End:** 0318F
Bits	**Access**	**Default**	**Label**	**Bit Description**
19	RW	0	SCT0_CLEAR	**Clear Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT0_start_inc is needed to restart the timer, but can be set simultaneously with SCT0_clear. When in Counter mode, a 1 on this action will clear the counter.
15	RW	0	LCT0_STOP	**Stop Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will stop the timer at its current value. This enables starting and stopping without losing timer value. When in Counter mode, this action is undefined.
11	RW	0	LCT0_START_INC	**Enable Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer or Peak Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
8	RW	0	SCT1_START_INC	**Enable Small 20-bit Counter/Timer1 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
7	RW	0	SCT0_START_INC	**Enable Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
6	RW	0	STOP_STORAGE	**Clear to logical 0 the Stor_Qual0 output, indicating to IOT storage to stop or suspend trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will clear CaptureDone.
5	RW	0	START_STORAGE	**Set to logical 1 the Stor_Qual0 output, indicating to IOT storage to begin or resume trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will set CaptureDone.

CSR Register Name: CLAUSE_ACTION_CONTROL_C3S0: Clause Action Control Registers C3S0			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 0318C	**Offset End:** 0318F

Bits	Access	Default	Label	Bit Description
4	RW	0	SET_TRIG_OUT	**Pulse the Trig_Out output.** All counter/ timers will continue to operate until a transition to the IDLE state, which may or may not be coincident with set_trig_out. This allows for continued operation past the Trig_Out event to control things like Stor_Qual or counting of events or time beyond a trigger for example.
3:0	RW	0	SET_SIGNAL_OUT	If Signal_Out_Mode[1:0] = 00 pulsed mode, then this action will pulse the corresponding Signal_Out[3:0] output for one cycle as result of an event match. If Signal_Out_Mode[1:0] = 01 level mode, then this action will change invert the level of the corresponding Signal_Out[3:0] output as a result of an event match. If Signal_Out_Mode[1:0] = 11 signal out qualifier mode, then this action will set the corresponding Signal_Out[3:0] output to the value of Set_Signal_Out action qualifier. This is a new mode for CTS 2.0.

13.7.38 CLAUSE_ACTION_CONTROL_C0S1: Clause Action Control Registers C0S1

CSR Register Name: CLAUSE_ACTION_CONTROL_C0S1: Clause Action Control Registers C0S1			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 03198	**Offset End:** 0319B

Bits	Access	Default	Label	Bit Description
31	RW	0	SET_SIGNAL_VALUE	Set_Signal_Value is an action qualifier for the Set_Signal_Out[3:0] actions for each clause of each state if Signal_Out_Mode[1:0] is set to 11. In this mode, the driven level of the Signal_Out output wire will be set or cleared to the value of the Set_Signal_Value action qualifier bit if the corresponding Set_Signal_Out[3:0] action is taken. The Set_Signal_Value is only used in the signal out qualifier mode of operation.
30:27	RW	0	SET_STATE	**Go to state m as result of clause/state event match, where m is the contents of this register.** The default reset value for each clause will be the IDLE state. The user can set the next state to any state to build a desired sequencer flow.
23	RW	0	LCT0_CLEAR	**Clear Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. LCT0_start_inc is needed to restart the timer, but can be set simultaneously with LCT0_clear. When in Counter mode, a 1 on this action will clear the counter. When in Peak Timer mode, a 1 on this action clears the timer, and halts its operation while simultaneously updating the Peak Timer Status register if the LCT exceeds the current contents of the Peak Timer Status register.
20	RW	0	SCT1_CLEAR	**Clear Small 20-bit Counter/Timer1 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT1_start_inc is needed to restart the timer, but can be set simultaneously with SCT1_clear. When in Counter mode, a 1 on this action will clear the counter.
19	RW	0	SCT0_CLEAR	**Clear Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT0_start_inc is needed to restart the timer, but can be set simultaneously with SCT0_clear. When in Counter mode, a 1 on this action will clear the counter.

CSR Register Name: CLAUSE_ACTION_CONTROL_C0S1: Clause Action Control Registers C0S1				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 03198	**Offset End:** 0319B
Bits	**Access**	**Default**	**Label**	**Bit Description**
15	RW	0	LCT0_STOP	**Stop Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will stop the timer at its current value. This enables starting and stopping without losing timer value. When in Counter mode, this action is undefined.
11	RW	0	LCT0_START_INC	**Enable Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer or Peak Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
8	RW	0	SCT1_START_INC	**Enable Small 20-bit Counter/Timer1 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
7	RW	0	SCT0_START_INC	**Enable Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
6	RW	0	STOP_STORAGE	**Clear to logical 0 the Stor_Qual0 output, indicating to IOT storage to stop or suspend trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will clear CaptureDone.
5	RW	0	START_STORAGE	**Set to logical 1 the Stor_Qual0 output, indicating to IOT storage to begin or resume trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will set CaptureDone.
4	RW	0	SET_TRIG_OUT	**Pulse the Trig_Out output.** All counter/ timers will continue to operate until a transition to the IDLE state, which may or may not be coincident with set_trig_out. This allows for continued operation past the Trig_Out event to control things like Stor_Qual or counting of events or time beyond a trigger for example.

CSR Register Name: CLAUSE_ACTION_CONTROL_C0S1: Clause Action Control Registers C0S1				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 03198	**Offset End:** 0319B
Bits	**Access**	**Default**	**Label**	**Bit Description**
3:0	RW	0	SET_SIGNAL_OUT	If Signal_Out_Mode[1:0] = 00 pulsed mode, then this action will pulse the corresponding Signal_Out[3:0] output for one cycle as result of an event match. If Signal_Out_Mode[1:0] = 01 level mode, then this action will change invert the level of the corresponding Signal_Out[3:0] output as a result of an event match. If Signal_Out_Mode[1:0] = 11 signal out qualifier mode, then this action will set the corresponding Signal_Out[3:0] output to the value of Set_Signal_Out action qualifier. This is a new mode for CTS 2.0.

13.7.39 CLAUSE_ACTION_CONTROL_C1S1: Clause Action Control Registers C1S1

CSR Register Name: CLAUSE_ACTION_CONTROL_C1S1: Clause Action Control Registers C1S1				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 0319C	**Offset End:** 0319F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	0	SET_SIGNAL_VALUE	Set_Signal_Value is an action qualifier for the Set_Signal_Out[3:0] actions for each clause of each state if Signal_Out_Mode[1:0] is set to 11. In this mode, the driven level of the Signal_Out output wire will be set or cleared to the value of the Set_Signal_Value action qualifier bit if the corresponding Set_Signal_Out[3:0] action is taken. The Set_Signal_Value is only used in the signal out qualifier mode of operation.
30:27	RW	0	SET_STATE	**Go to state m as result of clause/state event match, where m is the contents of this register.** The default reset value for each clause will be the IDLE state. The user can set the next state to any state to build a desired sequencer flow.

<table>
<tr><td colspan="6"><b>CSR Register Name: CLAUSE_ACTION_CONTROL_C1S1</b>: Clause Action Control Registers C1S1</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 0319C</td><td><b>Offset End:</b> 0319F</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>23</td><td>RW</td><td>0</td><td>LCT0_CLEAR</td><td colspan="2"><b>Clear Large 45-bit Counter/Timer0 on clause/state event match.</b> When in Timer mode, a 1 on this action clears the timer and halts its operation. LCT0_start_inc is needed to restart the timer, but can be set simultaneously with LCT0_clear. When in Counter mode, a 1 on this action will clear the counter. When in Peak Timer mode, a 1 on this action clears the timer, and halts its operation while simultaneously updating the Peak Timer Status register if the LCT exceeds the current contents of the Peak Timer Status register.</td></tr>
<tr><td>20</td><td>RW</td><td>0</td><td>SCT1_CLEAR</td><td colspan="2"><b>Clear Small 20-bit Counter/Timer1 on clause/state event match.</b> When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT1_start_inc is needed to restart the timer, but can be set simultaneously with SCT1_clear. When in Counter mode, a 1 on this action will clear the counter.</td></tr>
<tr><td>19</td><td>RW</td><td>0</td><td>SCT0_CLEAR</td><td colspan="2"><b>Clear Small 20-bit Counter/Timer0 on clause/state event match.</b> When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT0_start_inc is needed to restart the timer, but can be set simultaneously with SCT0_clear. When in Counter mode, a 1 on this action will clear the counter.</td></tr>
<tr><td>15</td><td>RW</td><td>0</td><td>LCT0_STOP</td><td colspan="2"><b>Stop Large 45-bit Counter/Timer0 on clause/state event match.</b> When in Timer mode, a 1 on this action will stop the timer at its current value. This enables starting and stopping without losing timer value. When in Counter mode, this action is undefined.</td></tr>
<tr><td>11</td><td>RW</td><td>0</td><td>LCT0_START_INC</td><td colspan="2"><b>Enable Large 45-bit Counter/Timer0 on clause/state event match.</b> When in Timer or Peak Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter</td></tr>
<tr><td>8</td><td>RW</td><td>0</td><td>SCT1_START_INC</td><td colspan="2"><b>Enable Small 20-bit Counter/Timer1 on clause/state event match.</b> When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter</td></tr>
</table>

CSR Register Name: CLAUSE_ACTION_CONTROL_C1S1: Clause Action Control Registers C1S1				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 0319C	**Offset End:** 0319F
Bits	**Access**	**Default**	**Label**	**Bit Description**
7	RW	0	SCT0_START_INC	**Enable Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
6	RW	0	STOP_STORAGE	**Clear to logical 0 the Stor_Qual0 output, indicating to IOT storage to stop or suspend trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will clear CaptureDone.
5	RW	0	START_STORAGE	**Set to logical 1 the Stor_Qual0 output, indicating to IOT storage to begin or resume trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will set CaptureDone.
4	RW	0	SET_TRIG_OUT	**Pulse the Trig_Out output.** All counter/ timers will continue to operate until a transition to the IDLE state, which may or may not be coincident with set_trig_out. This allows for continued operation past the Trig_Out event to control things like Stor_Qual or counting of events or time beyond a trigger for example.
3:0	RW	0	SET_SIGNAL_OUT	If Signal_Out_Mode[1:0] = 00 pulsed mode, then this action will pulse the corresponding Signal_Out[3:0] output for one cycle as result of an event match. If Signal_Out_Mode[1:0] = 01 level mode, then this action will change invert the level of the corresponding Signal_Out[3:0] output as a result of an event match. If Signal_Out_Mode[1:0] = 11 signal out qualifier mode, then this action will set the corresponding Signal_Out[3:0] output to the value of Set_Signal_Out action qualifier. This is a new mode for CTS 2.0.

13.7.40 CLAUSE_ACTION_CONTROL_C2S1: Clause Action Control Registers C2S1

CSR Register Name: CLAUSE_ACTION_CONTROL_C2S1: Clause Action Control Registers C2S1				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 031A0	**Offset End:** 031A3
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	0	SET_SIGNAL_VALUE	Set_Signal_Value is an action qualifier for the Set_Signal_Out[3:0] actions for each clause of each state if Signal_Out_Mode[1:0] is set to 11. In this mode, the driven level of the Signal_Out output wire will be set or cleared to the value of the Set_Signal_Value action qualifier bit if the corresponding Set_Signal_Out[3:0] action is taken. The Set_Signal_Value is only used in the signal out qualifier mode of operation.
30:27	RW	0	SET_STATE	**Go to state m as result of clause/state event match, where m is the contents of this register.** The default reset value for each clause will be the IDLE state. The user can set the next state to any state to build a desired sequencer flow.
23	RW	0	LCT0_CLEAR	**Clear Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. LCT0_start_inc is needed to restart the timer, but can be set simultaneously with LCT0_clear. When in Counter mode, a 1 on this action will clear the counter. When in Peak Timer mode, a 1 on this action clears the timer, and halts its operation while simultaneously updating the Peak Timer Status register if the LCT exceeds the current contents of the Peak Timer Status register.
20	RW	0	SCT1_CLEAR	**Clear Small 20-bit Counter/Timer1 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT1_start_inc is needed to restart the timer, but can be set simultaneously with SCT1_clear. When in Counter mode, a 1 on this action will clear the counter.

CSR Register Name: CLAUSE_ACTION_CONTROL_C2S1: Clause Action Control Registers C2S1			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 031A0	**Offset End:** 031A3

Bits	Access	Default	Label	Bit Description
19	RW	0	SCT0_CLEAR	**Clear Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT0_start_inc is needed to restart the timer, but can be set simultaneously with SCT0_clear. When in Counter mode, a 1 on this action will clear the counter.
15	RW	0	LCT0_STOP	**Stop Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will stop the timer at its current value. This enables starting and stopping without losing timer value. When in Counter mode, this action is undefined.
11	RW	0	LCT0_START_INC	**Enable Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer or Peak Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
8	RW	0	SCT1_START_INC	**Enable Small 20-bit Counter/Timer1 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
7	RW	0	SCT0_START_INC	**Enable Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
6	RW	0	STOP_STORAGE	**Clear to logical 0 the Stor_Qual0 output, indicating to IOT storage to stop or suspend trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will clear CaptureDone.
5	RW	0	START_STORAGE	**Set to logical 1 the Stor_Qual0 output, indicating to IOT storage to begin or resume trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will set CaptureDone.

CSR Register Name: CLAUSE_ACTION_CONTROL_C2S1: Clause Action Control Registers C2S1

Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 031A0	**Offset End:** 031A3

Bits	Access	Default	Label	Bit Description
4	RW	0	SET_TRIG_OUT	**Pulse the Trig_Out output.** All counter/ timers will continue to operate until a transition to the IDLE state, which may or may not be coincident with set_trig_out. This allows for continued operation past the Trig_Out event to control things like Stor_Qual or counting of events or time beyond a trigger for example.
3:0	RW	0	SET_SIGNAL_OUT	If Signal_Out_Mode[1:0] = 00 pulsed mode, then this action will pulse the corresponding Signal_Out[3:0] output for one cycle as result of an event match. If Signal_Out_Mode[1:0] = 01 level mode, then this action will change invert the level of the corresponding Signal_Out[3:0] output as a result of an event match. If Signal_Out_Mode[1:0] = 11 signal out qualifier mode, then this action will set the corresponding Signal_Out[3:0] output to the value of Set_Signal_Out action qualifier. This is a new mode for CTS 2.0.

13.7.41 CLAUSE_ACTION_CONTROL_C3S1: Clause Action Control Registers C3S1

CSR Register Name: CLAUSE_ACTION_CONTROL_C3S1: Clause Action Control Registers C3S1

Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 031A4	**Offset End:** 031A7

Bits	Access	Default	Label	Bit Description
31	RW	0	SET_SIGNAL_VALUE	Set_Signal_Value is an action qualifier for the Set_Signal_Out[3:0] actions for each clause of each state if Signal_Out_Mode[1:0] is set to 11. In this mode, the driven level of the Signal_Out output wire will be set or cleared to the value of the Set_Signal_Value action qualifier bit if the corresponding Set_Signal_Out[3:0] action is taken. The Set_Signal_Value is only used in the signal out qualifier mode of operation.
30:27	RW	0	SET_STATE	**Go to state m as result of clause/state event match, where m is the contents of this register.** The default reset value for each clause will be the IDLE state. The user can set the next state to any state to build a desired sequencer flow.
23	RW	0	LCT0_CLEAR	**Clear Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. LCT0_start_inc is needed to restart the timer, but can be set simultaneously with LCT0_clear. When in Counter mode, a 1 on this action will clear the counter. When in Peak Timer mode, a 1 on this action clears the timer, and halts its operation while simultaneously updating the Peak Timer Status register if the LCT exceeds the current contents of the Peak Timer Status register.
20	RW	0	SCT1_CLEAR	**Clear Small 20-bit Counter/Timer1 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT1_start_inc is needed to restart the timer, but can be set simultaneously with SCT1_clear. When in Counter mode, a 1 on this action will clear the counter.
19	RW	0	SCT0_CLEAR	**Clear Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT0_start_inc is needed to restart the timer, but can be set simultaneously with SCT0_clear. When in Counter mode, a 1 on this action will clear the counter.

CSR Register Name: CLAUSE_ACTION_CONTROL_C3S1: Clause Action Control Registers C3S1				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 031A4	**Offset End:** 031A7
Bits	**Access**	**Default**	**Label**	**Bit Description**
15	RW	0	LCT0_STOP	**Stop Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will stop the timer at its current value. This enables starting and stopping without losing timer value. When in Counter mode, this action is undefined.
11	RW	0	LCT0_START_INC	**Enable Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer or Peak Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
8	RW	0	SCT1_START_INC	**Enable Small 20-bit Counter/Timer1 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
7	RW	0	SCT0_START_INC	**Enable Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
6	RW	0	STOP_STORAGE	**Clear to logical 0 the Stor_Qual0 output, indicating to IOT storage to stop or suspend trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will clear CaptureDone.
5	RW	0	START_STORAGE	**Set to logical 1 the Stor_Qual0 output, indicating to IOT storage to begin or resume trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will set CaptureDone.
4	RW	0	SET_TRIG_OUT	**Pulse the Trig_Out output.** All counter/ timers will continue to operate until a transition to the IDLE state, which may or may not be coincident with set_trig_out. This allows for continued operation past the Trig_Out event to control things like Stor_Qual or counting of events or time beyond a trigger for example.

CSR Register Name: CLAUSE_ACTION_CONTROL_C3S1: Clause Action Control Registers C3S1				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 031A4	**Offset End:** 031A7
Bits	**Access**	**Default**	**Label**	**Bit Description**
3:0	RW	0	SET_SIGNAL_OUT	If Signal_Out_Mode[1:0] = 00 pulsed mode, then this action will pulse the corresponding Signal_Out[3:0] output for one cycle as result of an event match. If Signal_Out_Mode[1:0] = 01 level mode, then this action will change invert the level of the corresponding Signal_Out[3:0] output as a result of an event match. If Signal_Out_Mode[1:0] = 11 signal out qualifier mode, then this action will set the corresponding Signal_Out[3:0] output to the value of Set_Signal_Out action qualifier. This is a new mode for CTS 2.0.

13.7.42 CLAUSE_ACTION_CONTROL_C0S2: Clause Action Control Registers C0S2

CSR Register Name: CLAUSE_ACTION_CONTROL_C0S2: Clause Action Control Registers C0S2				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 031B0	**Offset End:** 031B3
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	0	SET_SIGNAL_VALUE	Set_Signal_Value is an action qualifier for the Set_Signal_Out[3:0] actions for each clause of each state if Signal_Out_Mode[1:0] is set to 11. In this mode, the driven level of the Signal_Out output wire will be set or cleared to the value of the Set_Signal_Value action qualifier bit if the corresponding Set_Signal_Out[3:0] action is taken. The Set_Signal_Value is only used in the signal out qualifier mode of operation.
30:27	RW	0	SET_STATE	**Go to state m as result of clause/state event match, where m is the contents of this register.** The default reset value for each clause will be the IDLE state. The user can set the next state to any state to build a desired sequencer flow.

CSR Register Name: CLAUSE_ACTION_CONTROL_C0S2: Clause Action Control Registers C0S2				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 031B0 **Offset End:** 031B3
Bits	**Access**	**Default**	**Label**	**Bit Description**
23	RW	0	LCT0_CLEAR	**Clear Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. LCT0_start_inc is needed to restart the timer, but can be set simultaneously with LCT0_clear. When in Counter mode, a 1 on this action will clear the counter. When in Peak Timer mode, a 1 on this action clears the timer, and halts its operation while simultaneously updating the Peak Timer Status register if the LCT exceeds the current contents of the Peak Timer Status register.
20	RW	0	SCT1_CLEAR	**Clear Small 20-bit Counter/Timer1 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT1_start_inc is needed to restart the timer, but can be set simultaneously with SCT1_clear. When in Counter mode, a 1 on this action will clear the counter.
19	RW	0	SCT0_CLEAR	**Clear Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT0_start_inc is needed to restart the timer, but can be set simultaneously with SCT0_clear. When in Counter mode, a 1 on this action will clear the counter.
15	RW	0	LCT0_STOP	**Stop Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will stop the timer at its current value. This enables starting and stopping without losing timer value. When in Counter mode, this action is undefined.
11	RW	0	LCT0_START_INC	**Enable Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer or Peak Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
8	RW	0	SCT1_START_INC	**Enable Small 20-bit Counter/Timer1 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter

CSR Register Name: CLAUSE_ACTION_CONTROL_C0S2: Clause Action Control Registers C0S2				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 031B0	**Offset End:** 031B3
Bits	**Access**	**Default**	**Label**	**Bit Description**
7	RW	0	SCT0_START_INC	**Enable Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
6	RW	0	STOP_STORAGE	**Clear to logical 0 the Stor_Qual0 output, indicating to IOT storage to stop or suspend trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will clear CaptureDone.
5	RW	0	START_STORAGE	**Set to logical 1 the Stor_Qual0 output, indicating to IOT storage to begin or resume trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will set CaptureDone.
4	RW	0	SET_TRIG_OUT	**Pulse the Trig_Out output.** All counter/ timers will continue to operate until a transition to the IDLE state, which may or may not be coincident with set_trig_out. This allows for continued operation past the Trig_Out event to control things like Stor_Qual or counting of events or time beyond a trigger for example.
3:0	RW	0	SET_SIGNAL_OUT	If Signal_Out_Mode[1:0] = 00 pulsed mode, then this action will pulse the corresponding Signal_Out[3:0] output for one cycle as result of an event match. If Signal_Out_Mode[1:0] = 01 level mode, then this action will change invert the level of the corresponding Signal_Out[3:0] output as a result of an event match. If Signal_Out_Mode[1:0] = 11 signal out qualifier mode, then this action will set the corresponding Signal_Out[3:0] output to the value of Set_Signal_Out action qualifier. This is a new mode for CTS 2.0.

13.7.43 CLAUSE_ACTION_CONTROL_C1S2: Clause Action Control Registers C1S2

CSR Register Name: CLAUSE_ACTION_CONTROL_C1S2: Clause Action Control Registers C1S2				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 031B4	**Offset End:** 031B7

Bits	Access	Default	Label	Bit Description
31	RW	0	SET_SIGNAL_VALUE	Set_Signal_Value is an action qualifier for the Set_Signal_Out[3:0] actions for each clause of each state if Signal_Out_Mode[1:0] is set to 11. In this mode, the driven level of the Signal_Out output wire will be set or cleared to the value of the Set_Signal_Value action qualifier bit if the corresponding Set_Signal_Out[3:0] action is taken. The Set_Signal_Value is only used in the signal out qualifier mode of operation.
30:27	RW	0	SET_STATE	**Go to state m as result of clause/state event match, where m is the contents of this register.** The default reset value for each clause will be the IDLE state. The user can set the next state to any state to build a desired sequencer flow.
23	RW	0	LCT0_CLEAR	**Clear Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. LCT0_start_inc is needed to restart the timer, but can be set simultaneously with LCT0_clear. When in Counter mode, a 1 on this action will clear the counter. When in Peak Timer mode, a 1 on this action clears the timer, and halts its operation while simultaneously updating the Peak Timer Status register if the LCT exceeds the current contents of the Peak Timer Status register.
20	RW	0	SCT1_CLEAR	**Clear Small 20-bit Counter/Timer1 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT1_start_inc is needed to restart the timer, but can be set simultaneously with SCT1_clear. When in Counter mode, a 1 on this action will clear the counter.

<table>
<tr><td colspan="6"><b>CSR Register Name:</b> <b>CLAUSE_ACTION_CONTROL_C1S2</b>: Clause Action Control Registers C1S2</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b>
031B4</td><td><b>Offset End:</b>
031B7</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>19</td><td>RW</td><td>0</td><td>SCT0_CLEAR</td><td colspan="2"><b>Clear Small 20-bit Counter/Timer0 on clause/state event match.</b> When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT0_start_inc is needed to restart the timer, but can be set simultaneously with SCT0_clear. When in Counter mode, a 1 on this action will clear the counter.</td></tr>
<tr><td>15</td><td>RW</td><td>0</td><td>LCT0_STOP</td><td colspan="2"><b>Stop Large 45-bit Counter/Timer0 on clause/state event match.</b> When in Timer mode, a 1 on this action will stop the timer at its current value. This enables starting and stopping without losing timer value. When in Counter mode, this action is undefined.</td></tr>
<tr><td>11</td><td>RW</td><td>0</td><td>LCT0_START_INC</td><td colspan="2"><b>Enable Large 45-bit Counter/Timer0 on clause/state event match.</b> When in Timer or Peak Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter</td></tr>
<tr><td>8</td><td>RW</td><td>0</td><td>SCT1_START_INC</td><td colspan="2"><b>Enable Small 20-bit Counter/Timer1 on clause/state event match.</b> When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter</td></tr>
<tr><td>7</td><td>RW</td><td>0</td><td>SCT0_START_INC</td><td colspan="2"><b>Enable Small 20-bit Counter/Timer0 on clause/state event match.</b> When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter</td></tr>
<tr><td>6</td><td>RW</td><td>0</td><td>STOP_STORAGE</td><td colspan="2"><b>Clear to logical 0 the Stor_Qual0 output, indicating to IOT storage to stop or suspend trace storage.</b> In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will clear CaptureDone.</td></tr>
<tr><td>5</td><td>RW</td><td>0</td><td>START_STORAGE</td><td colspan="2"><b>Set to logical 1 the Stor_Qual0 output, indicating to IOT storage to begin or resume trace storage.</b> In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will set CaptureDone.</td></tr>
</table>

CSR Register Name: CLAUSE_ACTION_CONTROL_C1S2: Clause Action Control Registers C1S2				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 031B4	**Offset End:** 031B7
Bits	**Access**	**Default**	**Label**	**Bit Description**
4	RW	0	SET_TRIG_OUT	**Pulse the Trig_Out output.** All counter/ timers will continue to operate until a transition to the IDLE state, which may or may not be coincident with set_trig_out. This allows for continued operation past the Trig_Out event to control things like Stor_Qual or counting of events or time beyond a trigger for example.
3:0	RW	0	SET_SIGNAL_OUT	If Signal_Out_Mode[1:0] = 00 pulsed mode, then this action will pulse the corresponding Signal_Out[3:0] output for one cycle as result of an event match. If Signal_Out_Mode[1:0] = 01 level mode, then this action will change invert the level of the corresponding Signal_Out[3:0] output as a result of an event match. If Signal_Out_Mode[1:0] = 11 signal out qualifier mode, then this action will set the corresponding Signal_Out[3:0] output to the value of Set_Signal_Out action qualifier. This is a new mode for CTS 2.0.

13.7.44 CLAUSE_ACTION_CONTROL_C2S2: Clause Action Control Registers C2S2

CSR Register Name: CLAUSE_ACTION_CONTROL_C2S2: Clause Action Control Registers C2S2			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 031B8	**Offset End:** 031BB

Bits	Access	Default	Label	Bit Description
31	RW	0	SET_SIGNAL_VALUE	Set_Signal_Value is an action qualifier for the Set_Signal_Out[3:0] actions for each clause of each state if Signal_Out_Mode[1:0] is set to 11. In this mode, the driven level of the Signal_Out output wire will be set or cleared to the value of the Set_Signal_Value action qualifier bit if the corresponding Set_Signal_Out[3:0] action is taken. The Set_Signal_Value is only used in the signal out qualifier mode of operation.
30:27	RW	0	SET_STATE	**Go to state m as result of clause/state event match, where m is the contents of this register.** The default reset value for each clause will be the IDLE state. The user can set the next state to any state to build a desired sequencer flow.
23	RW	0	LCT0_CLEAR	**Clear Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. LCT0_start_inc is needed to restart the timer, but can be set simultaneously with LCT0_clear. When in Counter mode, a 1 on this action will clear the counter. When in Peak Timer mode, a 1 on this action clears the timer, and halts its operation while simultaneously updating the Peak Timer Status register if the LCT exceeds the current contents of the Peak Timer Status register.
20	RW	0	SCT1_CLEAR	**Clear Small 20-bit Counter/Timer1 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT1_start_inc is needed to restart the timer, but can be set simultaneously with SCT1_clear. When in Counter mode, a 1 on this action will clear the counter.
19	RW	0	SCT0_CLEAR	**Clear Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT0_start_inc is needed to restart the timer, but can be set simultaneously with SCT0_clear. When in Counter mode, a 1 on this action will clear the counter.

<table>
<tr><td colspan="5"><b>CSR Register Name: CLAUSE_ACTION_CONTROL_C2S2</b>: Clause Action Control Registers C2S2</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 031B8
<b>Offset End:</b> 031BB</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th>Bit Description</th></tr>
<tr><td>15</td><td>RW</td><td>0</td><td>LCT0_STOP</td><td><b>Stop Large 45-bit Counter/Timer0 on clause/state event match.</b> When in Timer mode, a 1 on this action will stop the timer at its current value. This enables starting and stopping without losing timer value. When in Counter mode, this action is undefined.</td></tr>
<tr><td>11</td><td>RW</td><td>0</td><td>LCT0_START_INC</td><td><b>Enable Large 45-bit Counter/Timer0 on clause/state event match.</b> When in Timer or Peak Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter</td></tr>
<tr><td>8</td><td>RW</td><td>0</td><td>SCT1_START_INC</td><td><b>Enable Small 20-bit Counter/Timer1 on clause/state event match.</b> When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter</td></tr>
<tr><td>7</td><td>RW</td><td>0</td><td>SCT0_START_INC</td><td><b>Enable Small 20-bit Counter/Timer0 on clause/state event match.</b> When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter</td></tr>
<tr><td>6</td><td>RW</td><td>0</td><td>STOP_STORAGE</td><td><b>Clear to logical 0 the Stor_Qual0 output, indicating to IOT storage to stop or suspend trace storage.</b> In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will clear CaptureDone.</td></tr>
<tr><td>5</td><td>RW</td><td>0</td><td>START_STORAGE</td><td><b>Set to logical 1 the Stor_Qual0 output, indicating to IOT storage to begin or resume trace storage.</b> In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will set CaptureDone.</td></tr>
<tr><td>4</td><td>RW</td><td>0</td><td>SET_TRIG_OUT</td><td><b>Pulse the Trig_Out output.</b> All counter/ timers will continue to operate until a transition to the IDLE state, which may or may not be coincident with set_trig_out. This allows for continued operation past the Trig_Out event to control things like Stor_Qual or counting of events or time beyond a trigger for example.</td></tr>
</table>

CSR Register Name: CLAUSE_ACTION_CONTROL_C2S2: Clause Action Control Registers C2S2				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 031B8	**Offset End:** 031BB
Bits	**Access**	**Default**	**Label**	**Bit Description**
3:0	RW	0	SET_SIGNAL_OUT	If Signal_Out_Mode[1:0] = 00 pulsed mode, then this action will pulse the corresponding Signal_Out[3:0] output for one cycle as result of an event match. If Signal_Out_Mode[1:0] = 01 level mode, then this action will change invert the level of the corresponding Signal_Out[3:0] output as a result of an event match. If Signal_Out_Mode[1:0] = 11 signal out qualifier mode, then this action will set the corresponding Signal_Out[3:0] output to the value of Set_Signal_Out action qualifier. This is a new mode for CTS 2.0.

13.7.45 CLAUSE_ACTION_CONTROL_C3S2: Clause Action Control Registers C3S2

CSR Register Name: CLAUSE_ACTION_CONTROL_C3S2: Clause Action Control Registers C3S2				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 031BC	**Offset End:** 031BF
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	0	SET_SIGNAL_VALUE	Set_Signal_Value is an action qualifier for the Set_Signal_Out[3:0] actions for each clause of each state if Signal_Out_Mode[1:0] is set to 11. In this mode, the driven level of the Signal_Out output wire will be set or cleared to the value of the Set_Signal_Value action qualifier bit if the corresponding Set_Signal_Out[3:0] action is taken. The Set_Signal_Value is only used in the signal out qualifier mode of operation.
30:27	RW	0	SET_STATE	**Go to state m as result of clause/state event match, where m is the contents of this register.** The default reset value for each clause will be the IDLE state. The user can set the next state to any state to build a desired sequencer flow.

CSR Register Name: CLAUSE_ACTION_CONTROL_C3S2: Clause Action Control Registers C3S2				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 031BC **Offset End:** 031BF
Bits	**Access**	**Default**	**Label**	**Bit Description**
23	RW	0	LCT0_CLEAR	**Clear Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. LCT0_start_inc is needed to restart the timer, but can be set simultaneously with LCT0_clear. When in Counter mode, a 1 on this action will clear the counter. When in Peak Timer mode, a 1 on this action clears the timer, and halts its operation while simultaneously updating the Peak Timer Status register if the LCT exceeds the current contents of the Peak Timer Status register.
20	RW	0	SCT1_CLEAR	**Clear Small 20-bit Counter/Timer1 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT1_start_inc is needed to restart the timer, but can be set simultaneously with SCT1_clear. When in Counter mode, a 1 on this action will clear the counter.
19	RW	0	SCT0_CLEAR	**Clear Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT0_start_inc is needed to restart the timer, but can be set simultaneously with SCT0_clear. When in Counter mode, a 1 on this action will clear the counter.
15	RW	0	LCT0_STOP	**Stop Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will stop the timer at its current value. This enables starting and stopping without losing timer value. When in Counter mode, this action is undefined.
11	RW	0	LCT0_START_ INC	**Enable Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer or Peak Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
8	RW	0	SCT1_START_ INC	**Enable Small 20-bit Counter/Timer1 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter

CSR Register Name: CLAUSE_ACTION_CONTROL_C3S2: Clause Action Control Registers C3S2

Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 031BC	**Offset End:** 031BF

Bits	Access	Default	Label	Bit Description
7	RW	0	SCT0_START_INC	**Enable Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
6	RW	0	STOP_STORAGE	**Clear to logical 0 the Stor_Qual0 output, indicating to IOT storage to stop or suspend trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will clear CaptureDone.
5	RW	0	START_STORAGE	**Set to logical 1 the Stor_Qual0 output, indicating to IOT storage to begin or resume trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will set CaptureDone.
4	RW	0	SET_TRIG_OUT	**Pulse the Trig_Out output.** All counter/ timers will continue to operate until a transition to the IDLE state, which may or may not be coincident with set_trig_out. This allows for continued operation past the Trig_Out event to control things like Stor_Qual or counting of events or time beyond a trigger for example.
3:0	RW	0	SET_SIGNAL_OUT	If Signal_Out_Mode[1:0] = 00 pulsed mode, then this action will pulse the corresponding Signal_Out[3:0] output for one cycle as result of an event match. If Signal_Out_Mode[1:0] = 01 level mode, then this action will change invert the level of the corresponding Signal_Out[3:0] output as a result of an event match. If Signal_Out_Mode[1:0] = 11 signal out qualifier mode, then this action will set the corresponding Signal_Out[3:0] output to the value of Set_Signal_Out action qualifier. This is a new mode for CTS 2.0.

13.7.46 CLAUSE_ACTION_CONTROL_C0S3: Clause Action Control Registers C0S3

<table>
<tr><td colspan="6"><b>CSR Register Name: CLAUSE_ACTION_CONTROL_C0S3</b>: Clause Action Control Registers C0S3</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b>
031C8</td><td><b>Offset End:</b>
031CB</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31</td><td>RW</td><td>0</td><td>SET_SIGNAL_VALUE</td><td colspan="2">Set_Signal_Value is an action qualifier for the Set_Signal_Out[3:0] actions for each clause of each state if Signal_Out_Mode[1:0] is set to 11. In this mode, the driven level of the Signal_Out output wire will be set or cleared to the value of the Set_Signal_Value action qualifier bit if the corresponding Set_Signal_Out[3:0] action is taken. The Set_Signal_Value is only used in the signal out qualifier mode of operation.</td></tr>
<tr><td>30:27</td><td>RW</td><td>0</td><td>SET_STATE</td><td colspan="2"><b>Go to state m as result of clause/state event match, where m is the contents of this register.</b> The default reset value for each clause will be the IDLE state. The user can set the next state to any state to build a desired sequencer flow.</td></tr>
<tr><td>23</td><td>RW</td><td>0</td><td>LCT0_CLEAR</td><td colspan="2"><b>Clear Large 45-bit Counter/Timer0 on clause/state event match.</b> When in Timer mode, a 1 on this action clears the timer and halts its operation. LCT0_start_inc is needed to restart the timer, but can be set simultaneously with LCT0_clear. When in Counter mode, a 1 on this action will clear the counter. When in Peak Timer mode, a 1 on this action clears the timer, and halts its operation while simultaneously updating the Peak Timer Status register if the LCT exceeds the current contents of the Peak Timer Status register.</td></tr>
<tr><td>20</td><td>RW</td><td>0</td><td>SCT1_CLEAR</td><td colspan="2"><b>Clear Small 20-bit Counter/Timer1 on clause/state event match.</b> When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT1_start_inc is needed to restart the timer, but can be set simultaneously with SCT1_clear. When in Counter mode, a 1 on this action will clear the counter.</td></tr>
</table>

CSR Register Name: CLAUSE_ACTION_CONTROL_C0S3: Clause Action Control Registers C0S3

Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 031C8	**Offset End:** 031CB

Bits	Access	Default	Label	Bit Description
19	RW	0	SCT0_CLEAR	**Clear Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT0_start_inc is needed to restart the timer, but can be set simultaneously with SCT0_clear. When in Counter mode, a 1 on this action will clear the counter.
15	RW	0	LCT0_STOP	**Stop Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will stop the timer at its current value. This enables starting and stopping without losing timer value. When in Counter mode, this action is undefined.
11	RW	0	LCT0_START_INC	**Enable Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer or Peak Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
8	RW	0	SCT1_START_INC	**Enable Small 20-bit Counter/Timer1 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
7	RW	0	SCT0_START_INC	**Enable Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
6	RW	0	STOP_STORAGE	**Clear to logical 0 the Stor_Qual0 output, indicating to IOT storage to stop or suspend trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will clear CaptureDone.
5	RW	0	START_STORAGE	**Set to logical 1 the Stor_Qual0 output, indicating to IOT storage to begin or resume trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will set CaptureDone.

CSR Register Name: **CLAUSE_ACTION_CONTROL_C0S3**: Clause Action Control Registers C0S3				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 031C8	**Offset End:** 031CB
Bits	**Access**	**Default**	**Label**	**Bit Description**
4	RW	0	SET_TRIG_OUT	**Pulse the Trig_Out output.** All counter/ timers will continue to operate until a transition to the IDLE state, which may or may not be coincident with set_trig_out. This allows for continued operation past the Trig_Out event to control things like Stor_Qual or counting of events or time beyond a trigger for example.
3:0	RW	0	SET_SIGNAL_OUT	If Signal_Out_Mode[1:0] = 00 pulsed mode, then this action will pulse the corresponding Signal_Out[3:0] output for one cycle as result of an event match. If Signal_Out_Mode[1:0] = 01 level mode, then this action will change invert the level of the corresponding Signal_Out[3:0] output as a result of an event match. If Signal_Out_Mode[1:0] = 11 signal out qualifier mode, then this action will set the corresponding Signal_Out[3:0] output to the value of Set_Signal_Out action qualifier. This is a new mode for CTS 2.0.

13.7.47 CLAUSE_ACTION_CONTROL_C1S3: Clause Action Control Registers C1S3

CSR Register Name: **CLAUSE_ACTION_CONTROL_C1S3**: Clause Action Control Registers C1S3			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 031CC	**Offset End:** 031CF

Bits	Access	Default	Label	Bit Description
31	RW	0	SET_SIGNAL_ VALUE	Set_Signal_Value is an action qualifier for the Set_Signal_Out[3:0] actions for each clause of each state if Signal_Out_Mode[1:0] is set to 11. In this mode, the driven level of the Signal_Out output wire will be set or cleared to the value of the Set_Signal_Value action qualifier bit if the corresponding Set_Signal_Out[3:0] action is taken. The Set_Signal_Value is only used in the signal out qualifier mode of operation.
30:27	RW	0	SET_STATE	**Go to state m as result of clause/state event match, where m is the contents of this register.** The default reset value for each clause will be the IDLE state. The user can set the next state to any state to build a desired sequencer flow.
23	RW	0	LCT0_CLEAR	**Clear Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. LCT0_start_inc is needed to restart the timer, but can be set simultaneously with LCT0_clear. When in Counter mode, a 1 on this action will clear the counter. When in Peak Timer mode, a 1 on this action clears the timer, and halts its operation while simultaneously updating the Peak Timer Status register if the LCT exceeds the current contents of the Peak Timer Status register.
20	RW	0	SCT1_CLEAR	**Clear Small 20-bit Counter/Timer1 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT1_start_inc is needed to restart the timer, but can be set simultaneously with SCT1_clear. When in Counter mode, a 1 on this action will clear the counter.
19	RW	0	SCT0_CLEAR	**Clear Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT0_start_inc is needed to restart the timer, but can be set simultaneously with SCT0_clear. When in Counter mode, a 1 on this action will clear the counter.

CSR Register Name: CLAUSE_ACTION_CONTROL_C1S3: Clause Action Control Registers C1S3

Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 031CC	**Offset End:** 031CF

Bits	Access	Default	Label	Bit Description
15	RW	0	LCT0_STOP	**Stop Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will stop the timer at its current value. This enables starting and stopping without losing timer value. When in Counter mode, this action is undefined.
11	RW	0	LCT0_START_INC	**Enable Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer or Peak Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
8	RW	0	SCT1_START_INC	**Enable Small 20-bit Counter/Timer1 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
7	RW	0	SCT0_START_INC	**Enable Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
6	RW	0	STOP_STORAGE	**Clear to logical 0 the Stor_Qual0 output, indicating to IOT storage to stop or suspend trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will clear CaptureDone.
5	RW	0	START_STORAGE	**Set to logical 1 the Stor_Qual0 output, indicating to IOT storage to begin or resume trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will set CaptureDone.
4	RW	0	SET_TRIG_OUT	**Pulse the Trig_Out output.** All counter/ timers will continue to operate until a transition to the IDLE state, which may or may not be coincident with set_trig_out. This allows for continued operation past the Trig_Out event to control things like Stor_Qual or counting of events or time beyond a trigger for example.

CSR Register Name: CLAUSE_ACTION_CONTROL_C1S3: Clause Action Control Registers C1S3				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 031CC	**Offset End:** 031CF

Bits	Access	Default	Label	Bit Description
3:0	RW	0	SET_SIGNAL_OUT	If Signal_Out_Mode[1:0] = 00 pulsed mode, then this action will pulse the corresponding Signal_Out[3:0] output for one cycle as result of an event match. If Signal_Out_Mode[1:0] = 01 level mode, then this action will change invert the level of the corresponding Signal_Out[3:0] output as a result of an event match. If Signal_Out_Mode[1:0] = 11 signal out qualifier mode, then this action will set the corresponding Signal_Out[3:0] output to the value of Set_Signal_Out action qualifier. This is a new mode for CTS 2.0.

13.7.48 CLAUSE_ACTION_CONTROL_C2S3: Clause Action Control Registers C2S3

CSR Register Name: CLAUSE_ACTION_CONTROL_C2S3: Clause Action Control Registers C2S3				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 031D0	**Offset End:** 031D3

Bits	Access	Default	Label	Bit Description
31	RW	0	SET_SIGNAL_VALUE	Set_Signal_Value is an action qualifier for the Set_Signal_Out[3:0] actions for each clause of each state if Signal_Out_Mode[1:0] is set to 11. In this mode, the driven level of the Signal_Out output wire will be set or cleared to the value of the Set_Signal_Value action qualifier bit if the corresponding Set_Signal_Out[3:0] action is taken. The Set_Signal_Value is only used in the signal out qualifier mode of operation.
30:27	RW	0	SET_STATE	**Go to state m as result of clause/state event match, where m is the contents of this register.** The default reset value for each clause will be the IDLE state. The user can set the next state to any state to build a desired sequencer flow.

<table>
<tr><td colspan="5"><b>CSR Register Name:</b> <b>CLAUSE_ACTION_CONTROL_C2S3</b>: Clause Action Control Registers C2S3</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 031D0   <b>Offset End:</b> 031D3</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th>Bit Description</th></tr>
<tr><td>23</td><td>RW</td><td>0</td><td>LCT0_CLEAR</td><td><b>Clear Large 45-bit Counter/Timer0 on clause/state event match.</b> When in Timer mode, a 1 on this action clears the timer and halts its operation. LCT0_start_inc is needed to restart the timer, but can be set simultaneously with LCT0_clear. When in Counter mode, a 1 on this action will clear the counter. When in Peak Timer mode, a 1 on this action clears the timer, and halts its operation while simultaneously updating the Peak Timer Status register if the LCT exceeds the current contents of the Peak Timer Status register.</td></tr>
<tr><td>20</td><td>RW</td><td>0</td><td>SCT1_CLEAR</td><td><b>Clear Small 20-bit Counter/Timer1 on clause/state event match.</b> When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT1_start_inc is needed to restart the timer, but can be set simultaneously with SCT1_clear. When in Counter mode, a 1 on this action will clear the counter.</td></tr>
<tr><td>19</td><td>RW</td><td>0</td><td>SCT0_CLEAR</td><td><b>Clear Small 20-bit Counter/Timer0 on clause/state event match.</b> When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT0_start_inc is needed to restart the timer, but can be set simultaneously with SCT0_clear. When in Counter mode, a 1 on this action will clear the counter.</td></tr>
<tr><td>15</td><td>RW</td><td>0</td><td>LCT0_STOP</td><td><b>Stop Large 45-bit Counter/Timer0 on clause/state event match.</b> When in Timer mode, a 1 on this action will stop the timer at its current value. This enables starting and stopping without losing timer value. When in Counter mode, this action is undefined.</td></tr>
<tr><td>11</td><td>RW</td><td>0</td><td>LCT0_START_
INC</td><td><b>Enable Large 45-bit Counter/Timer0 on clause/state event match.</b> When in Timer or Peak Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter</td></tr>
<tr><td>8</td><td>RW</td><td>0</td><td>SCT1_START_
INC</td><td><b>Enable Small 20-bit Counter/Timer1 on clause/state event match.</b> When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter</td></tr>
</table>

CSR Register Name: CLAUSE_ACTION_CONTROL_C2S3: Clause Action Control Registers C2S3

Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 031D0	**Offset End:** 031D3

Bits	Access	Default	Label	Bit Description
7	RW	0	SCT0_START_INC	**Enable Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
6	RW	0	STOP_STORAGE	**Clear to logical 0 the Stor_Qual0 output, indicating to IOT storage to stop or suspend trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will clear CaptureDone.
5	RW	0	START_STORAGE	**Set to logical 1 the Stor_Qual0 output, indicating to IOT storage to begin or resume trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will set CaptureDone.
4	RW	0	SET_TRIG_OUT	**Pulse the Trig_Out output.** All counter/ timers will continue to operate until a transition to the IDLE state, which may or may not be coincident with set_trig_out. This allows for continued operation past the Trig_Out event to control things like Stor_Qual or counting of events or time beyond a trigger for example.
3:0	RW	0	SET_SIGNAL_OUT	If Signal_Out_Mode[1:0] = 00 pulsed mode, then this action will pulse the corresponding Signal_Out[3:0] output for one cycle as result of an event match. If Signal_Out_Mode[1:0] = 01 level mode, then this action will change invert the level of the corresponding Signal_Out[3:0] output as a result of an event match. If Signal_Out_Mode[1:0] = 11 signal out qualifier mode, then this action will set the corresponding Signal_Out[3:0] output to the value of Set_Signal_Out action qualifier. This is a new mode for CTS 2.0.

13.7.49 CLAUSE_ACTION_CONTROL_C3S3: Clause Action Control Registers C3S3

CSR Register Name: CLAUSE_ACTION_CONTROL_C3S3: Clause Action Control Registers C3S3				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 031D4	**Offset End:** 031D7
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	0	SET_SIGNAL_VALUE	Set_Signal_Value is an action qualifier for the Set_Signal_Out[3:0] actions for each clause of each state if Signal_Out_Mode[1:0] is set to 11. In this mode, the driven level of the Signal_Out output wire will be set or cleared to the value of the Set_Signal_Value action qualifier bit if the corresponding Set_Signal_Out[3:0] action is taken. The Set_Signal_Value is only used in the signal out qualifier mode of operation.
30:27	RW	0	SET_STATE	**Go to state m as result of clause/state event match, where m is the contents of this register.** The default reset value for each clause will be the IDLE state. The user can set the next state to any state to build a desired sequencer flow.
23	RW	0	LCT0_CLEAR	**Clear Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. LCT0_start_inc is needed to restart the timer, but can be set simultaneously with LCT0_clear. When in Counter mode, a 1 on this action will clear the counter. When in Peak Timer mode, a 1 on this action clears the timer, and halts its operation while simultaneously updating the Peak Timer Status register if the LCT exceeds the current contents of the Peak Timer Status register.
20	RW	0	SCT1_CLEAR	**Clear Small 20-bit Counter/Timer1 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT1_start_inc is needed to restart the timer, but can be set simultaneously with SCT1_clear. When in Counter mode, a 1 on this action will clear the counter.

CSR Register Name: CLAUSE_ACTION_CONTROL_C3S3: Clause Action Control Registers C3S3				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 031D4	**Offset End:** 031D7
Bits	**Access**	**Default**	**Label**	**Bit Description**
19	RW	0	SCT0_CLEAR	**Clear Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action clears the timer and halts its operation. SCT0_start_inc is needed to restart the timer, but can be set simultaneously with SCT0_clear. When in Counter mode, a 1 on this action will clear the counter.
15	RW	0	LCT0_STOP	**Stop Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will stop the timer at its current value. This enables starting and stopping without losing timer value. When in Counter mode, this action is undefined.
11	RW	0	LCT0_START_INC	**Enable Large 45-bit Counter/Timer0 on clause/state event match.** When in Timer or Peak Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
8	RW	0	SCT1_START_INC	**Enable Small 20-bit Counter/Timer1 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
7	RW	0	SCT0_START_INC	**Enable Small 20-bit Counter/Timer0 on clause/state event match.** When in Timer mode, a 1 on this action will start the timer; When in Counter mode, a 1 on this action will increment the counter
6	RW	0	STOP_STORAGE	**Clear to logical 0 the Stor_Qual0 output, indicating to IOT storage to stop or suspend trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will clear CaptureDone.
5	RW	0	START_STORAGE	**Set to logical 1 the Stor_Qual0 output, indicating to IOT storage to begin or resume trace storage.** In North Peak, Stor_Qual0 is connected to CaptureDone, so this action will set CaptureDone.

CSR Register Name: CLAUSE_ACTION_CONTROL_C3S3: Clause Action Control Registers C3S3				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 031D4	**Offset End:** 031D7
Bits	**Access**	**Default**	**Label**	**Bit Description**
4	RW	0	SET_TRIG_OUT	**Pulse the Trig_Out output.** All counter/ timers will continue to operate until a transition to the IDLE state, which may or may not be coincident with set_trig_out. This allows for continued operation past the Trig_Out event to control things like Stor_Qual or counting of events or time beyond a trigger for example.
3:0	RW	0	SET_SIGNAL_OUT	If Signal_Out_Mode[1:0] = 00 pulsed mode, then this action will pulse the corresponding Signal_Out[3:0] output for one cycle as result of an event match. If Signal_Out_Mode[1:0] = 01 level mode, then this action will change invert the level of the corresponding Signal_Out[3:0] output as a result of an event match. If Signal_Out_Mode[1:0] = 11 signal out qualifier mode, then this action will set the corresponding Signal_Out[3:0] output to the value of Set_Signal_Out action qualifier. This is a new mode for CTS 2.0.

13.7.50 SCT0_CONTROL: SCT0 Control Register

CSR Register Name: SCT0_CONTROL: SCT0 Control Register			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 03240	**Offset End:** 03243

Bits	Access	Default	Label	Bit Description
26:23	RW	0	SCT0_EVENTSEL	**New for CTS 2.**0 is the dedicated event counter mode of operation for the small counter /timers. This new SCT0_EventSel[3:0] field selects which Event_In[15:0] input to count if this SCT is configured in this new mode. F = Event_in15 . . 0 = Event_in0 Note that the number of bits in this field will be reduced to the minimum number to support the parameter selected for Event_Inn:0 inputs. If for example 4 Event_In wires is selected, then only 2 EventSel configuration bits will be required. The remaining bit positions will be grounded to 0.
22	RW	0	SCT0_MODE_1	SCT0_Mode1 is combined with SCT0_Mode0 in bit position 20 to create new SCT modes available for CTS 2.0. If SCT0_Mode1 is set to 0, then the SCT operation is backwards compatible to CTS 1.0 functionality. The mode definitions are now: 11 - Reserved 10 - dedicated event counter mode new for CTS 2.0 01 - backward compatible to CTS 1.0 to operate as a timer 00 - backward compatible to CTS 1.0 to operate as a counter
21	RW	0	SCT0_TIMEBASE	**SCT0_Timebase select.** If this Counter/Timer is running in timer mode, this bit will configure its time-base. 0 = increment continuously at LCLK frequency if enabled 1 = increment only if Reference_Pulse input is high.
20	RW	0	SCT0_MODE_0	SCT0_Mode0 combined with SCT0_Mode1 in bit position 22 which was added for CTS 2.0. The mode definitions are now: 11 - Reserved 10 - dedicated event counter mode new for CTS 2.0 01 - backward compatible to CTS 1.0 to operate as a timer 00 - backward compatible to CTS 1.0 to operate as a counter
19:0	RW	0	SCT0_MATCH_VALUE	**Small Counter/Timer 0 SCT0 Match Value.** The value programmed in this register will be compared against SCT0 every cycle to create a match event to the clause logic.

13.7.51 SCT1_CONTROL: SCT1 Control Register

CSR Register Name: SCT1_CONTROL: SCT1 Control Register				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 03244	**Offset End:** 03247
Bits	**Access**	**Default**	**Label**	**Bit Description**
26:23	RW	0	SCT1_EVENTSEL	**New for CTS 2.**0 is the dedicated event counter mode of operation for the small counter /timers. This new SCT1_EventSel[3:0] field selects which Event_In[15:0] input to count if this SCT is configured in this new mode. F = Event_in15 . . 0 = Event_in0 Note that the number of bits in this field will be reduced to the minimum number to support the parameter selected for Event_Inn:0 inputs. If for example 4 Event_In wires is selected, then only 2 EventSel configuration bits will be required. The remaining bit positions will be grounded to 0.
22	RW	0	SCT1_MODE_1	SCT1_Mode1 is combined with SCT1_Mode0 in bit position 20 to create new SCT modes available for CTS 2.0. If SCT1_Mode1 is set to 0, then the SCT operation is backwards compatible to CTS 1.0 functionality. The mode definitions are now: 11 - Reserved 10 - dedicated event counter mode new for CTS 2.0 01 - backward compatible to CTS 1.0 to operate as a timer 00 - backward compatible to CTS 1.0 to operate as a counte
21	RW	0	SCT1_TIMEBASE	**SCT1_Timebase select.** If this Counter/Timer is running in timer mode, this bit will configure its time-base. 0 = increment continuously at LCLK frequency if enabled 1 = increment only if Reference_Pulse input is high.

CSR Register Name: SCT1_CONTROL: SCT1 Control Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 03244	**Offset End:** 03247
Bits	**Access**	**Default**	**Label**	**Bit Description**
20	RW	0	SCT1_MODE_0	SCT1_Mode0 combined with SCT1_Mode1 in bit position 22 which was added for CTS 2.0. The mode definitions are now: 11 - Reserved 10 - dedicated event counter mode new for CTS 2.0 01 - backward compatible to CTS 1.0 to operate as a timer 00 - backward compatible to CTS 1.0 to operate as a counter
19:0	RW	0	SCT1_MATCH_VALUE	**Small Counter/Timer 1 SCT1 Match Value.** The value programmed in this register will be compared against SCT1 every cycle to create a match event to the clause logic.

13.7.52 LCT0_LOWER_CONTROL: LCT0 Lower Control Register

CSR Register Name: LCT0_LOWER_CONTROL: LCT0 Lower Control Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 03250	**Offset End:** 03253
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RW	0	LCT0_MATCH_VALUE	**Large Counter/Timer 0 LCT0 Lower Match value.** The value programmed in this register will be compared against LCT0 every cycle to create a match event to the clause logic. The contents of this register is combined with LCT0 Upper Control Register contents to create a 45-bit LCT0_match_value

13.7.53 LCT0_UPPER_CONTROL: LCT0 Upper Control Register

CSR Register Name: LCT0_UPPER_CONTROL: LCT0 Upper Control Register			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 03254	**Offset End:** 03257

Bits	Access	Default	Label	Bit Description
15	RW	0	LCT0_MODE_1	This is a second LCT0_Mode1 bit added for CTS 2.0 functionality to be combined with LCT0_Mode0 The LCT0_Mode[1:0] is decoded as follows to perform the following functions: 11- LCT operates in Peak Timer mode. This is new functionality for revision 2.0 of the CTS. 10 - Reserved. This is new functionality for revision 2.0 of the CTS. 01 - LCT operates as a Timer. This is backwards compatible with revision 1.0 of the CTS. 00 - LCT operates as a counter. This is backwards compatible with revision 1.0 of the CTS.
14	RW	0	LCT0_TIMEBASE	**LCT0_Timebase select.** If this Counter/Timer is running in timer mode, this bit will configure its timebase. 0 = increment continuously at LCLK frequency if enabled 1 = increment only if Reference_Pulse input is high.
13	RW	0	LCT0_MODE_0	**Mode control for the LCT0.** This is now combined with LCT0_Mode1 in bit position 15, which was added for CTS revision 2.0 functionality. See above for a description.

13.7.54 SCT0_CURRENT_STATUS: SCT0 Current Status Register

CSR Register Name: SCT0_CURRENT_STATUS: SCT0 Current Status Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 03270	**Offset End:** 03273
Bits	**Access**	**Default**	**Label**	**Bit Description**
20	RW	0	SCT0_RUNNING	The SCT0_Running status register will be high true if the SCT0 is running in timer mode, and is currently active counting due to a SCT0_start_inc action. The system software can read the status of this bit, and can optionally set it along with the Force_State bit to restore the current operating state of the SCT0 if in timer mode after a C6 power state.

<table>
<tr><td colspan="6"><b>CSR Register Name: SCT0_CURRENT_STATUS</b>: SCT0 Current Status Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 03270</td><td><b>Offset End:</b> 03273</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>19:0</td><td>RW</td><td>0</td><td>SCT0_CURRENT_VALUE</td><td colspan="2"><b>Small Counter/Timer 0 SCT0 Current Value.</b> This is a shadow status register reflecting the current state of the counter/timer that can be written & forced into the CTS functional register using the Force_State during a re-start.</td></tr>
</table>

13.7.55 SCT1_CURRENT_STATUS: SCT1 Current Status Register

<table>
<tr><td colspan="6"><b>CSR Register Name: SCT1_CURRENT_STATUS</b>: SCT1 Current Status Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 03274</td><td><b>Offset End:</b> 03277</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>20</td><td>RW</td><td>0</td><td>SCT1_RUNNING</td><td colspan="2">The SCT1_Running status register will be high true if the SCT1 is running in timer mode, and is currently active counting due to a SCT1_start_inc action. The system software can read the status of this bit, and can optionally set it along with the Force_State bit to restore the current operating state of the SCT1 if in timer mode after a C6 power state.</td></tr>
<tr><td>19:0</td><td>RW</td><td>0</td><td>SCT1_CURRENT_VALUE</td><td colspan="2"><b>Small Counter/Timer 1 SCT1 Current Status.</b> This is a shadow status register reflecting the current state of the counter/timer that can be written & forced into the CTS functional register using the Force_State during a re-start.</td></tr>
</table>

13.7.56 LCT0_LOWER_CURRENT_STATUS: LCT0 Lower Current Status Register

<table>
<tr><td colspan="4"><b>CSR Register Name: LCT0_LOWER_CURRENT_STATUS</b>: LCT0 Lower Current Status Register</td></tr>
<tr><td><b>Bar:</b> CSR_MTB_BAR</td><td><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 03280</td><td><b>Offset End:</b> 03283</td></tr>
</table>

Bits	Access	Default	Label	Bit Description
31:0	RW	0	LCT0_CURRENT_VALUE	**Large Counter/Timer 0 LCT0 Lower Current Status.** This is a shadow status register reflecting the current state of the counter/timer that can be written & forced into the CTS functional register using the Force_State during a re-start. The contents of this register is combined with LCT0 Upper Current Status register contents to create a 45-bit LCT0_current_value

13.7.57 LCT0_UPPER_CURRENT_STATUS: LCT0 Upper Current Status Register

CSR Register Name: LCT0_UPPER_CURRENT_STATUS: LCT0 Upper Current Status Register				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 03284	**Offset End:** 03287
Bits	**Access**	**Default**	**Label**	**Bit Description**
13	RW	0	LCT0_RUNNING	The LCT0_Running status register will be high true if the LCT0 is running in timer mode, and is currently active counting due to a LCT0_start_inc action. The system software can read the status of this bit, and can optionally set it along with the Force_State bit to restore the current operating state of the LCT0 if in timer mode after a C6 power state.

13.7.58 CTS_STATUS: CTS Status Register

CSR Register Name: CTS_STATUS: CTS Status Register			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 032A0	**Offset End:** 032A3

Bits	Access	Default	Label	Bit Description
22:16	RW	0	STORE_QUAL_INITIAL_VALUE	**Store_Qual7:1 output Initial Value.** These registers allow control of the Store_Qual outputs upon starting or re-starting the CTS. The Store_Qual7:1 outputs will take the contents of the respective Store_Qual_Initial_Value7:1 register upon starting or re-starting the CTS. These registers function as status registers that mirror the state of the Store_Qual7:1 outputs. This allows for trace storage to be ON when the CTS begins or re-starts operation. These registers are updated with the Store_Qual7:1 output values when Update_Status is set.
13	RO	0	CTS_PIPELINED	**CTS in pipelined mode status.** If CTS synthesis has trouble meeting the timing requirements of the target LCLK, then the logic can have a parameter set to pipeline event match within the clause & state logic. This status can be read via S/W to inform the user that any increment action would require an additional cycle before the updated match result would be available to the clause & state logic. This can be hard coded to the parameter setting in the .xml similar to the CTS_Revision.
12:10	RO	0	CTS_REVISION	**Hard coded CTS revision.** Start at revision 0. Increment any time a feature change that would impact the S/W occurs. 000 = CTS revision 1.0 001 = CTS revision 2.0
9:6	RW	0	SIGNAL_OUT_STATUS	**Signal_Out[3:0] output status.** This status register reflects the state of the signal_out wires. The state of the signal_out wires can be forced by the S/W writing the contents into this register and having the Force_State bit set when the sequencer is enabled, otherwise it s updated every cycle with the signal_out values when Update_Status is set.

CSR Register Name: CTS_STATUS: CTS Status Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 032A0	**Offset End:** 032A3
Bits	**Access**	**Default**	**Label**	**Bit Description**
5	RW	0	STOR_QUAL_INITIAL_VALUE	**Stor_Qual0 output Initial Value.** This register allows for control of the Stor_Qual0 output upon starting or re-starting the CTS. The Stor_Qual0 output will take the contents of this register upon starting or re-starting the CTS. This register functions as a status register that mirrors the state of the Stor_Qual0 output. This allows for trace storage to be ON when the CTS begins or re-starts operation. This is updated with the Stor_Qual0 value when Update_Status is set.
4	RW	0	SEQUENCER_TRIG	**Sequencer_Trig is the shadow status register for the internal sticky state bit called sequencer_trig.** sequencer_trig is set when the Trig_Out is asserted and held until the sequencer is re-started or until Cold_Reset. Warm_Reset has no effect. This register is updated when Update_Status is set. This register can be written & forced into the sequencer_trig state bit using the Force_State during a re-start.
3:0	RW	8	SEQUENCER_STATE	**Sequencer Current State Status.** This is a shadow status register that can be written & forced into the CTS functional register using the Force_State during a re-start. This is updated with the sequencer state when Update_Status is set.

13.7.59 CTS_CONTROL: CTS Control Register

CSR Register Name: CTS_CONTROL: CTS Control Register			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 032A4	**Offset End:** 032A7

Bits	Access	Default	Label	Bit Description
21:20	RW	0	SIGNAL_OUT_MODE	Signal_Out_Mode[1:0] selects the mode of all the Signal_Out[3:0] wires. The following describes the Signal_Out_Mode[1:0] settings: 11 - The Signal_Out output wire level driven will be set or cleared to the value of the Set_Signal_Value action qualifier bit if the corresponding Set_Signal_Out[3:0] action is taken. This effectively creates a signal out qualifier mode for each of the Signal_Out wires regardless of the current state of the Signal_Out[3:0] wire. In any clause the user can set or clear the driven level of individual Signal_Out wires that are selected by the Set_Signal_Out[3:0] action, but no combination of setting and clearing is supported. 10 - Reserved for future use. 01 - The Signal Out output wire will toggle their logical level when the corresponding Set_Signal_Out[3:0] action is taken. Using bit 0 as an example, if signal_out0 is currently driven to a logical 0, then if Set_signal_out0 action is true, the signal out0 output will transition to a logical 1. It will remain at this value level until the next Set_signal_out0 action is set where the signal_out0 output will transition toggle from a logical 1 to a logical 0 and remain at that new level. 00 - The Signal Out output wire will be pulsed to a logical 1 for a single LCLK cycle if the corresponding Set_Signal_Out[3:0] action is taken.
19	RW	0	SIGNAL_IN_EDGE_DET	**Signal_In[3:0] input Edge Detect Enable.** If = 1, will select a rising edge detector on the Signal_In[3:0] inputs If = 0, will bypass the edge detector on the Signal_In[3:0] inputs There is a single Signal_In_edge_det configuration bit that controls the operation of all signal_in input signals.
18	RW	0	UPDATE_STATUS	**Update_Status enables the shadow status register updates with the functional register.** This bit must be set then cleared to read a snapshot of all the status registers. In the cycle before this bit is cleared, all status will be updated.

CSR Register Name: **CTS_CONTROL**: CTS Control Register				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 032A4	**Offset End:** 032A7
Bits	**Access**	**Default**	**Label**	**Bit Description**
17	RW	0	FORCE_STATE	**Force the state of the sequencer.** When set to 1, the CTS will be forced to jam the contents of the RW_V status registers into the functional registers within CTS. These include Stor_Qual state, Sequencer_State[3:0], and all Counter/Timer contents, including the Timer_Running bits
16:1	RW	0	EVENT_IN_EDGE_DET	**Event_In[15:0] input Edge Detect Enable.** If = 1, will select a rising edge detector on the Event_In[15:0] inputs If = 0, will bypass the edge detector on the Event_In[15:0] inputs. There is a separate Event_In_edge_det to control the operation of each Event_In input. This number is parameterized
0	RW	0	SEQUENCER_ENABLE	The Sequencer_Enable configuration register is ANDed together with the CTS_En input pin to enable normal operation of the sequencer. 1 = The state will transition from IDLE to S0 for the normal operating case, and will transition from IDLE to the Sequencer_State[3:0] value if Force_State is set. 0 = Disable or hold the sequencer in the IDLE state, clear counters & timers

13.7.60 LCT0_PEAK_TIMER_LOWER_STATUS: LCT0 Peak Timer Lower Status Register

CSR Register Name: **LCT0_PEAK_TIMER_LOWER_STATUS**: LCT0 Peak Timer Lower Status Register			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 032A8	**Offset End:** 032AB

Bits	Access	Default	Label	Bit Description
31:0	RO	0	LCT0_PEAK_TIMER_VALUE	**Large Counter/Timer 0 LCT0 Peak Timer Lower Value.** This is a status register reflecting the lower 32 bits of the LCT0 peak timer value. The contents of this register are combined with LCT0 Upper Current Status register contents to create a 45-bit LCT0_current_value. This is a read only register of the peak timer value, and as such does not implement a shadow status register to save gates. It therefore is not affected by update_status, and is not writeable, or jamable .

13.7.61 LCT0_PEAK_TIMER_UPPER_STATUS: LCT0 Peak Timer Upper Status Register

CSR Register Name: LCT0_PEAK_TIMER_UPPER_STATUS: LCT0 Peak Timer Upper Status Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 032AC	**Offset End:** 032AF
Bits	**Access**	**Default**	**Label**	**Bit Description**
12:0	RO	0	LCT0_PEAK_TIMER_VALUE	**Large Counter/Timer 0 LCT0 Peak Timer Upper Value.** This is a status register reflecting the upper 13 bits of the LCT0 peak timer value. The contents of this register are combined with LCT0 Lower Current Status register contents to create a 45-bit LCT0_current_value. This is a read only register of the peak timer value, and as such does not implement a shadow status register to save gates. It therefore is not affected by update_status, and is not writeable, or jamable .

13.7.62 PARAMETER_STATUS: Parameter Status Register

CSR Register Name: PARAMETER_STATUS: Parameter Status Register			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 032C8	**Offset End:** 032CB

Bits	Access	Default	Label	Bit Description
29	RO	0	PIPELINE_LCT	This is a read only register that reflects the parameter compiled by this design to select additional pipelining in the LCTs.
28	RO	0	PIPELINE_SCT	This is a read only register that reflects the parameter compiled by this design to select additional pipelining in the SCTs.
27:25	RO	3	NUM_STATES	This is a read only register that reflects the parameter compiled by this design to select the number of states in the state machine in addition to IDLE. The range is from 1 to 8. The value read from this register is the parameter - 1, so the user will read from 0 to 7.
24:22	RO	3	NUM_CLAUSE S	**This is a read only register that reflects the parameter compiled by this design to select the number of clauses per state.** The range is from 1 to 6. The value read from this register is the parameter - 1, so the user will read from 0 to 7.
21:18	RO	D	NUM_EVENTS	**This is a read only register that reflects the parameter compiled by this design to select the number of Event inputs.** The range is from 1 to 16. The value read from this register is the parameter - 1, so the user will read from 0 to 15.
17:16	RO	3	NUM_SIGNAL S	This is a read only register that reflects the parameter compiled by this design to select the number of Signal_In inputs and Signal_Out outputs. The range is from 1 to 4. The value read from this register is the parameter - 1, so the user will read from 0 to 3.
15:13	RO	2	NUM_SCTS	This is a read only register that reflects the parameter compiled by this design to select the number of Small Counter Timers. The range is from 0 to 4. The value read from this register is the parameter without subtracting 1 from it.
12:10	RO	1	NUM_LCTS	This is a read only register that reflects the parameter compiled by this design to select the number of Large Counter Timers. The range is from 0 to 4. The value read from this register is the parameter without subtracting 1 from it.

<table>
<tr><td colspan="6"><b>CSR Register Name: PARAMETER_STATUS</b>: Parameter Status Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b>
032C8</td><td><b>Offset End:</b>
032CB</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>9:4</td><td>RO</td><td>1F</td><td>LCT_SIZE</td><td colspan="2"><b>This is a read only register that reflects the parameter compiled by this design to select the width size of the LCTs.</b> The range is from 1 to 45. The value read from this register is the parameter - 1, so the user will read from 0 to 44.</td></tr>
<tr><td>3</td><td>RO</td><td>1</td><td>PEAK_TIMER_SELECTED</td><td colspan="2">This is a read only register that reflects the parameter compiled by this design to select the Peak Timer capability for LCTs.
0 = disabled, and
1 = enabled.</td></tr>
<tr><td>2:0</td><td>RO</td><td>3</td><td>NUM_STOR_QUAL</td><td colspan="2"><b>This is a read only register that reflects the parameter compiled by this design to select the number of Stor_Qual outputs.</b> The range is from 1 to 8. The value read from this register is the parameter - 1, so the user will read from 0 to 7.</td></tr>
</table>

13.7.63 EXTENDED_CLAUSE_ACTION_CONTROL_C0S0: Extended Clause Action Control Registers C0S0

<table>
<tr><td colspan="6"><b>CSR Register Name: EXTENDED_CLAUSE_ACTION_CONTROL_C0S0</b>: Extended Clause Action Control Registers C0S0</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b>
03300</td><td><b>Offset End:</b>
03303</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>11:9</td><td>RW</td><td>0</td><td>STOP_STORAGE</td><td colspan="2">Stop_storage7:1 will clear to a logical 0 the respective Store_Qual7:1 output.</td></tr>
<tr><td>3:1</td><td>RW</td><td>0</td><td>START_STORAGE</td><td colspan="2">Start_storage7:1 will set to a logical 1 the respective Store_Qual7:1 output.</td></tr>
</table>

13.7.64 EXTENDED_CLAUSE_ACTION_CONTROL_C1S0: Extended Clause Action Control Registers C1S0

<table>
<tr><td colspan="6"><b>CSR Register Name:</b> <b>EXTENDED_CLAUSE_ACTION_CONTROL_C1S0</b>: Extended Clause Action Control Registers C1S0</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 03304</td><td><b>Offset End:</b> 03307</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>11:9</td><td>RW</td><td>0</td><td>STOP_STORAGE</td><td colspan="2">Stop_storage7:1 will clear to a logical 0 the respective Store_Qual7:1 output.</td></tr>
<tr><td>3:1</td><td>RW</td><td>0</td><td>START_STORAGE</td><td colspan="2">Start_storage7:1 will set to a logical 1 the respective Store_Qual7:1 output.</td></tr>
</table>

13.7.65 EXTENDED_CLAUSE_ACTION_CONTROL_C2S0: Extended Clause Action Control Registers C2S0

<table>
<tr><td colspan="6"><b>CSR Register Name:</b> <b>EXTENDED_CLAUSE_ACTION_CONTROL_C2S0</b>: Extended Clause Action Control Registers C2S0</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 03308</td><td><b>Offset End:</b> 0330B</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>11:9</td><td>RW</td><td>0</td><td>STOP_STORAGE</td><td colspan="2">Stop_storage7:1 will clear to a logical 0 the respective Store_Qual7:1 output.</td></tr>
<tr><td>3:1</td><td>RW</td><td>0</td><td>START_STORAGE</td><td colspan="2">Start_storage7:1 will set to a logical 1 the respective Store_Qual7:1 output.</td></tr>
</table>

13.7.66 EXTENDED_CLAUSE_ACTION_CONTROL_C3S0: Extended Clause Action Control Registers C3S0

<table>
<tr><td colspan="6"><b>CSR Register Name:</b> <b>EXTENDED_CLAUSE_ACTION_CONTROL_C3S0</b>: Extended Clause Action Control Registers C3S0</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 0330C</td><td><b>Offset End:</b> 0330F</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>11:9</td><td>RW</td><td>0</td><td>STOP_STORAGE</td><td colspan="2">Stop_storage7:1 will clear to a logical 0 the respective Store_Qual7:1 output.</td></tr>
<tr><td>3:1</td><td>RW</td><td>0</td><td>START_STORAGE</td><td colspan="2">Start_storage7:1 will set to a logical 1 the respective Store_Qual7:1 output.</td></tr>
</table>

13.7.67 EXTENDED_CLAUSE_ACTION_CONTROL_C0S1: Extended Clause Action Control Registers C0S1

<table>
<tr><td colspan="6"><b>CSR Register Name: EXTENDED_CLAUSE_ACTION_CONTROL_C0S1</b>: Extended Clause Action Control Registers C0S1</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 03318</td><td><b>Offset End:</b> 0331B</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>11:9</td><td>RW</td><td>0</td><td>STOP_STORAGE</td><td colspan="2">Stop_storage7:1 will clear to a logical 0 the respective Store_Qual7:1 output.</td></tr>
<tr><td>3:1</td><td>RW</td><td>0</td><td>START_STORAGE</td><td colspan="2">Start_storage7:1 will set to a logical 1 the respective Store_Qual7:1 output.</td></tr>
</table>

13.7.68 EXTENDED_CLAUSE_ACTION_CONTROL_C1S1: Extended Clause Action Control Registers C1S1

<table>
<tr><td colspan="6"><b>CSR Register Name: EXTENDED_CLAUSE_ACTION_CONTROL_C1S1</b>: Extended Clause Action Control Registers C1S1</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 0331C</td><td><b>Offset End:</b> 0331F</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>11:9</td><td>RW</td><td>0</td><td>STOP_STORAGE</td><td colspan="2">Stop_storage7:1 will clear to a logical 0 the respective Store_Qual7:1 output.</td></tr>
<tr><td>3:1</td><td>RW</td><td>0</td><td>START_STORAGE</td><td colspan="2">Start_storage7:1 will set to a logical 1 the respective Store_Qual7:1 output.</td></tr>
</table>

13.7.69 EXTENDED_CLAUSE_ACTION_CONTROL_C2S1: Extended Clause Action Control Registers C2S1

<table>
<tr><td colspan="6"><b>CSR Register Name: EXTENDED_CLAUSE_ACTION_CONTROL_C2S1</b>: Extended Clause Action Control Registers C2S1</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 03320</td><td><b>Offset End:</b> 03323</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>11:9</td><td>RW</td><td>0</td><td>STOP_STORAGE</td><td colspan="2">Stop_storage7:1 will clear to a logical 0 the respective Store_Qual7:1 output.</td></tr>
</table>

CSR Register Name: EXTENDED_CLAUSE_ACTION_CONTROL_C2S1: Extended Clause Action Control Registers C2S1				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 03320	**Offset End:** 03323
Bits	**Access**	**Default**	**Label**	**Bit Description**
3:1	RW	0	START_STORAGE	Start_storage7:1 will set to a logical 1 the respective Store_Qual7:1 output.

13.7.70 EXTENDED_CLAUSE_ACTION_CONTROL_C3S1: Extended Clause Action Control Registers C3S1

CSR Register Name: EXTENDED_CLAUSE_ACTION_CONTROL_C3S1: Extended Clause Action Control Registers C3S1				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 03324	**Offset End:** 03327
Bits	**Access**	**Default**	**Label**	**Bit Description**
11:9	RW	0	STOP_STORAGE	Stop_storage7:1 will clear to a logical 0 the respective Store_Qual7:1 output.
3:1	RW	0	START_STORAGE	Start_storage7:1 will set to a logical 1 the respective Store_Qual7:1 output.

13.7.71 EXTENDED_CLAUSE_ACTION_CONTROL_C0S2: Extended Clause Action Control Registers C0S2

CSR Register Name: EXTENDED_CLAUSE_ACTION_CONTROL_C0S2: Extended Clause Action Control Registers C0S2				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 03330	**Offset End:** 03333
Bits	**Access**	**Default**	**Label**	**Bit Description**
11:9	RW	0	STOP_STORAGE	Stop_storage7:1 will clear to a logical 0 the respective Store_Qual7:1 output.
3:1	RW	0	START_STORAGE	Start_storage7:1 will set to a logical 1 the respective Store_Qual7:1 output.

13.7.72 EXTENDED_CLAUSE_ACTION_CONTROL_C1S2: Extended Clause Action Control Registers C1S2

CSR Register Name: EXTENDED_CLAUSE_ACTION_CONTROL_C1S2: Extended Clause Action Control Registers C1S2

Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 03334	**Offset End:** 03337

Bits	Access	Default	Label	Bit Description
11:9	RW	0	STOP_STORAGE	Stop_storage7:1 will clear to a logical 0 the respective Store_Qual7:1 output.
3:1	RW	0	START_STORAGE	Start_storage7:1 will set to a logical 1 the respective Store_Qual7:1 output.

13.7.73 EXTENDED_CLAUSE_ACTION_CONTROL_C2S2: Extended Clause Action Control Registers C2S2

CSR Register Name: EXTENDED_CLAUSE_ACTION_CONTROL_C2S2: Extended Clause Action Control Registers C2S2

Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 03338	**Offset End:** 0333B

Bits	Access	Default	Label	Bit Description
11:9	RW	0	STOP_STORAGE	Stop_storage7:1 will clear to a logical 0 the respective Store_Qual7:1 output.
3:1	RW	0	START_STORAGE	Start_storage7:1 will set to a logical 1 the respective Store_Qual7:1 output.

13.7.74 EXTENDED_CLAUSE_ACTION_CONTROL_C3S2: Extended Clause Action Control Registers C3S2

CSR Register Name: EXTENDED_CLAUSE_ACTION_CONTROL_C3S2: Extended Clause Action Control Registers C3S2

Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 0333C	**Offset End:** 0333F

Bits	Access	Default	Label	Bit Description
11:9	RW	0	STOP_STORAGE	Stop_storage7:1 will clear to a logical 0 the respective Store_Qual7:1 output.
3:1	RW	0	START_STORAGE	Start_storage7:1 will set to a logical 1 the respective Store_Qual7:1 output.

13.7.75 EXTENDED_CLAUSE_ACTION_CONTROL_C0S3: Extended Clause Action Control Registers C0S3

CSR Register Name: EXTENDED_CLAUSE_ACTION_CONTROL_C0S3: Extended Clause Action Control Registers C0S3					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 03348	**Offset End:** 0334B
Bits	**Access**	**Default**	**Label**	**Bit Description**	
11:9	RW	0	STOP_STORAGE	Stop_storage7:1 will clear to a logical 0 the respective Store_Qual7:1 output.	
3:1	RW	0	START_STORAGE	Start_storage7:1 will set to a logical 1 the respective Store_Qual7:1 output.	

13.7.76 EXTENDED_CLAUSE_ACTION_CONTROL_C1S3: Extended Clause Action Control Registers C1S3

CSR Register Name: EXTENDED_CLAUSE_ACTION_CONTROL_C1S3: Extended Clause Action Control Registers C1S3					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 0334C	**Offset End:** 0334F
Bits	**Access**	**Default**	**Label**	**Bit Description**	
11:9	RW	0	STOP_STORAGE	Stop_storage7:1 will clear to a logical 0 the respective Store_Qual7:1 output.	
3:1	RW	0	START_STORAGE	Start_storage7:1 will set to a logical 1 the respective Store_Qual7:1 output.	

13.7.77 EXTENDED_CLAUSE_ACTION_CONTROL_C2S3: Extended Clause Action Control Registers C2S3

CSR Register Name: EXTENDED_CLAUSE_ACTION_CONTROL_C2S3: Extended Clause Action Control Registers C2S3					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 03350	**Offset End:** 03353
Bits	**Access**	**Default**	**Label**	**Bit Description**	
11:9	RW	0	STOP_STORAGE	Stop_storage7:1 will clear to a logical 0 the respective Store_Qual7:1 output.	

CSR Register Name: **EXTENDED_CLAUSE_ACTION_CONTROL_C2S3**: Extended Clause Action Control Registers C2S3				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 03350	**Offset End:** 03353
Bits	**Access**	**Default**	**Label**	**Bit Description**
3:1	RW	0	START_STORAGE	Start_storage7:1 will set to a logical 1 the respective Store_Qual7:1 output.

13.7.78 EXTENDED_CLAUSE_ACTION_CONTROL_C3S3: Extended Clause Action Control Registers C3S3

CSR Register Name: **EXTENDED_CLAUSE_ACTION_CONTROL_C3S3**: Extended Clause Action Control Registers C3S3				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 03354	**Offset End:** 03357
Bits	**Access**	**Default**	**Label**	**Bit Description**
11:9	RW	0	STOP_STORAGE	Stop_storage7:1 will clear to a logical 0 the respective Store_Qual7:1 output.
3:1	RW	0	START_STORAGE	Start_storage7:1 will set to a logical 1 the respective Store_Qual7:1 output.

13.8 PCI Configuration

13.8.1 PCI Configuration Registers Summary

PCI Configuration Registers			
Offset Start	**Offset End**	**Symbol**	**Register Name/Function**
0	3	VID	Vendor ID
4	7	CMD	Command and Status Register
8	b	RID	Class Code Register
8	b	RID	Revision ID Register
c	f	HT	Header Type Register
10	13	MTB_LBAR	MTB Lower Base Address Register
14	17	MTB_UBAR	MTB Upper Base Address Register

PCI Configuration Registers			
Offset Start	**Offset End**	**Symbol**	**Register Name/Function**
18	1b	SW_LBAR	Software Lower Base Address Register
1c	1f	SW_UBAR	Software Upper Base Address Register
20	23	RTIT_LBAR	Intel® PT Lower Base Address Register
24	27	RTIT_UBAR	Intel® PT Upper Base Address Register
2c	2f	SVID	Subsystem Vendor ID
34	37	CAP	Capabilities Pointer
3C	3F	INTL	Interrupt Line and Interrupt Pin
40	43	MSICID	MSI Capability ID,MSI Next Capability Pointer,MSI Message Control Register
44	47	MSILMA	MSI Lower Message Address
48	4b	MSIUMA	MSI Upper Message Address
4c	4f	MSIMD	MSI Message Data
70	73	FW_LBAR	Firmware Lower Bar
74	77	FW_UBAR	Firmware Upper Bar
80	83	NPKDSC	NPK Device Specific Control and Device Specific Status
84	87	ISTOT	Inbound Switch Timeout Timer
88	8B	ICTOT	Inbound CCB Timeout Timer
90	93	NPKDSD	NPK Device Specific Defeature

13.8.2 VID: Vendor ID

CSR Register Name: **VID**: Vendor ID			
Bar: PCI Cfg	**Reset:** host_rst_b	**Offset Start:** 0	**Offset End:** 3

Bits	Access	Default	Label	Bit Description
31:16	RO	varies	DID	Device ID: The value that uniquely identifies the North Peak from all other PCI devices.
15:0	RO	8086	VID	**Vendor ID:** 8086 is Intel Vendor Identification code

13.8.3 CMD: Command and Status Register

CSR Register Name: **CMD**: Command and Status Register				
Bar: PCI Cfg		**Reset:** host_rst_b	**Offset Start:** 4	**Offset End:** 7
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RO	0	Reserved	Reserved
30	RW/1C	0	SSE	Signaled System Error: This bit is set when the device has detected an un-correctable error and reported it via SERR message over sideband. This requires SERR Enable bit to be set in Command register.
29	RW/1C	0	RMA	Received Master Abort Status: This bit is set when device receives a Completion transaction with Unsupported Request completion status. No error will be reported
28	RW/1C	0	RTA	Received Target Abort Status: This bit is set when device receives a Completion transaction with Completer Abort completion status. No error will be reported
27	RW/1C	0	STA	Signaled Target Abort Status: Set by the device when aborting a request that violates the device programming model. When SERR Enable is set SERR message will be send over sideband
26:21	RO	0	Reserved	Reserved
20	RO	1	CLIST	**Capabilities List:** Indicates the controller contains a capabilities pointer list and the capability pointer register is implemented at offset 0x40 in the configuration space

CSR Register Name: CMD: Command and Status Register				
Bar: PCI Cfg		**Reset:** host_rst_b		**Offset Start:** 4 **Offset End:** 7
Bits	**Access**	**Default**	**Label**	**Bit Description**
19	RO	0	INSTAT	**Interrupt Status: Reflects the state of the interrupt pin at the input of the enable/disable circuit.** When the interrupt is asserted, and cleared when the interrupt is cleared (independent of the state of Interrupt Disable bit in command register. This bit is only associated with the INTx messages and has no meaning if the device is using MSI
18:16	RO	0	Reserved	Reserved
15:11	RO	0	Reserved	Reserved
10	RW	0	IE	**Interrupt Disable: Disables the function to generate INTx interrupt.** A value of 0 enables the function to generate INTA messages on the sideband fabric. Note: this bit has no effect on MSI generation.
9	RO	0	Reserved	Reserved
8	RW	0	SERREN	**System Error Enable:** Setting this bit enables the generation of System Error message, when required through sideband interface
7:3	RO	0	Reserved	Reserved
2	RW	0	BME	Bus Master Enable: When set enables the ability to issue Memory or IO requests, including MSI.
1	RW	0	MSE	**Memory Space Enable:** When set, Memory Space Decoding is enabled and memory transactions targeting the device are accepted Note: The MSE has to be set to accept any memory transaction on the primary interface targeting any of North Peak s BARs including its FW_BAR
0	RO	0	IOSE	Reserved

13.8.4 RID: Class Code Register

CSR Register Name: RID: Class Code Register			
Bar: PCI Cfg	**Reset:** host_rst_b	**Offset Start:** 8	**Offset End:** b

Bits	Access	Default	Label	Bit Description
31:8	RO	13	CC	Class Code

13.8.5 RID: Revision ID Register

<table>
<tr><td colspan="6"><b>CSR Register Name: RID</b>: Revision ID Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> PCI Cfg</td><td colspan="2"><b>Reset:</b> host_rst_b</td><td><b>Offset Start:</b> 8</td><td><b>Offset End:</b> b</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>7:0</td><td>RO</td><td>0</td><td>RID</td><td colspan="2"><b>Revision ID:</b> Indicates the device specific revision identifier derived from and input strap</td></tr>
</table>

13.8.6 HT: Header Type Register

<table>
<tr><td colspan="6"><b>CSR Register Name: HT</b>: Header Type Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> PCI Cfg</td><td colspan="2"><b>Reset:</b> host_rst_b</td><td><b>Offset Start:</b> c</td><td><b>Offset End:</b> f</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:24</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">Reserved</td></tr>
<tr><td>23</td><td>RO</td><td>0</td><td>MFD</td><td colspan="2">Reserved</td></tr>
<tr><td>22:16</td><td>RO</td><td>0</td><td>HT</td><td colspan="2"><b>Header Type:</b> Implements a Type 0 configuration header</td></tr>
<tr><td>15:0</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">Reserved</td></tr>
</table>

13.8.7 MTB_LBAR: MTB Lower Base Address Register

<table>
<tr><td colspan="6"><b>CSR Register Name: MTB_LBAR</b>: MTB Lower Base Address Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> PCI Cfg</td><td colspan="2"><b>Reset:</b> host_rst_b</td><td><b>Offset Start:</b> 10</td><td><b>Offset End:</b> 13</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:20</td><td>RW</td><td>0</td><td>LADDR</td><td colspan="2"><b>Lower Base Address:</b> Lower programmable Base Address</td></tr>
<tr><td>19:4</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">Hardwired to 0 to indicate the memory space size required by this CSR_MTB_BAR MMIO space is 1MB</td></tr>
<tr><td>3</td><td>RO</td><td>0</td><td>PF</td><td colspan="2"><b>Prefetchable:</b> Value of 0 indicates the BAR cannot be prefetched</td></tr>
</table>

<table>
<tr><td colspan="6"><b>CSR Register Name: MTB_LBAR</b>: MTB Lower Base Address Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> PCI Cfg</td><td colspan="2"><b>Reset:</b> host_rst_b</td><td><b>Offset Start:</b> 10</td><td><b>Offset End:</b> 13</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>2:1</td><td>RO</td><td>2</td><td>ADRNG</td><td colspan="2">Address Range: Value of 0x2 indicates that the BAR is located anywhere system memory space (i.e. 64-bit addressing)</td></tr>
<tr><td>0</td><td>RO</td><td>0</td><td>SPTY</td><td colspan="2"><b>Space Type:</b> Value of 0 indicates the BAR is located in memory space</td></tr>
</table>

13.8.8 MTB_UBAR: MTB Upper Base Address Register

<table>
<tr><td colspan="6"><b>CSR Register Name: MTB_UBAR</b>: MTB Upper Base Address Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> PCI Cfg</td><td colspan="2"><b>Reset:</b> host_rst_b</td><td><b>Offset Start:</b> 14</td><td><b>Offset End:</b> 17</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:0</td><td>RW</td><td>0</td><td>UADDR</td><td colspan="2"><b>Upper Base Address:</b> Upper programmable Base Address</td></tr>
</table>

13.8.9 SW_LBAR: Software Lower Base Address Register

<table>
<tr><td colspan="6"><b>CSR Register Name: SW_LBAR</b>: Software Lower Base Address Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> PCI Cfg</td><td colspan="2"><b>Reset:</b> host_rst_b</td><td><b>Offset Start:</b> 18</td><td><b>Offset End:</b> 1b</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:21</td><td>RW</td><td>0</td><td>ADDR</td><td colspan="2"><b>Lower Base Address:</b> Lower programmable Base Address</td></tr>
</table>

CSR Register Name: SW_LBAR: Software Lower Base Address Register				
Bar: PCI Cfg	**Reset:** host_rst_b		**Offset Start:** 18	**Offset End:** 1b
Bits	**Access**	**Default**	**Label**	**Bit Description**
20:4	RO	0	Reserved	Reserved: Hardwired to 0 to indicate the memory space size required by this SW_BAR MMIO space is given by the following formula with 5 <= N <= 30 (this allows for up to 1GB of address space) Size=2^N bytes Where N can be computed from the number of implemented STMR masters (SW_MSTR_NUM), the number of channels per master (SW_CHCNT), and the size of each channel which is fixed at 64B. To this end, N needs to be selected such that the MMIO space size is equal to 64*SW_MSTR_NUM *SW_CHCNT. Hence, N can be computed from the following equation: N=log2(64*SW_MSTR_NUM*SW_CHCNT)
3	RO	0	PF	**Prefetchable:** Value of 0 indicates the BAR cannot be prefetched
2:1	RO	2	ADRNG	Address Range: Value of 0x2 indicates that the BAR is located anywhere system memory space (i.e. 64-bit addressing)
0	RO	0	SPTY	**Space Type:** Value of 0 indicates the BAR is located in memory space

13.8.10 SW_UBAR: Software Upper Base Address Register

CSR Register Name: SW_UBAR: Software Upper Base Address Register				
Bar: PCI Cfg	**Reset:** host_rst_b		**Offset Start:** 1c	**Offset End:** 1f
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RW	0	UADDR	**Upper Base Address:** Upper programmable Base Address

13.8.11 RTIT_LBAR: Intel® PT Lower Base Address Register

<table>
<tr><td colspan="6"><b>CSR Register Name: RTIT_LBAR</b>: Intel® PT Lower Base Address Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> PCI Cfg</td><td colspan="2"><b>Reset:</b> host_rst_b</td><td><b>Offset Start:</b> 20</td><td><b>Offset End:</b> 23</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:N</td><td>RW</td><td>0</td><td>LADDR</td><td colspan="2"><b>Lower Base Address:</b> Lower programmable Base Address</td></tr>
<tr><td>N-1:4</td><td>RO</td><td>0</td><td>ZERO</td><td colspan="2">Reserved: Hardwired to 0 to indicate the memory space size required by the RTMR space. The value of N in this field, and in the ADDR field, is determined by the number of logical processors supported (N):
<table>
<tr><td>Num LP</td><td>N</td><td>Comments</td></tr>
<tr><td>0</td><td>-</td><td>No Intel(R) PT Srcs</td></tr>
<tr><td>1</td><td>8</td><td>256B</td></tr>
<tr><td>2</td><td>9</td><td>512B</td></tr>
<tr><td>3</td><td>10</td><td>1024B</td></tr>
<tr><td>4</td><td>10</td><td>1024B</td></tr>
<tr><td>5-8</td><td>11</td><td>2048B</td></tr>
<tr><td>9-16</td><td>12</td><td>4096B</td></tr>
</table></td></tr>
<tr><td>3</td><td>RO</td><td>0</td><td>PF</td><td colspan="2"><b>Prefetchable:</b> Value of 0 indicates the BAR cannot be prefetched</td></tr>
<tr><td>2:1</td><td>RO</td><td>2</td><td>TYPE</td><td colspan="2">Address Range: Value of 0x2 indicates that the BAR is located anywhere system memory space (i.e. 64-bit addressing)</td></tr>
<tr><td>0</td><td>RO</td><td>0</td><td>MEM</td><td colspan="2"><b>Space Type:</b> Value of 0 indicates the BAR is located in memory space</td></tr>
</table>

13.8.12 RTIT_UBAR: Intel® PT Upper Base Address Register

<table>
<tr><td colspan="6"><b>CSR Register Name: RTIT_UBAR</b>: Intel® PT Upper Base Address Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> PCI Cfg</td><td colspan="2"><b>Reset:</b> host_rst_b</td><td><b>Offset Start:</b> 24</td><td><b>Offset End:</b> 27</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:0</td><td>RW</td><td>0</td><td>UADDR</td><td colspan="2"><b>Upper Base Address:</b> Upper programmable Base Address</td></tr>
</table>

13.8.13 SVID: Subsystem Vendor ID

CSR Register Name: **SVID**: Subsystem Vendor ID				
Bar: PCI Cfg		**Reset:** host_rst_b		**Offset Start:** 2c / **Offset End:** 2f
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:16	RO	varies	SID	**Subsystem ID:** Default value derived from strap then writable only once
15:0	RO	varies	SVID	**Subsystem Vendor ID:** Derived from strapCan be set once by BIOS then becomes RO

13.8.14 CAP: Capabilities Pointer

CSR Register Name: **CAP**: Capabilities Pointer				
Bar: PCI Cfg		**Reset:** host_rst_b		**Offset Start:** 34 / **Offset End:** 37
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:8	RO	0	Reserved	Reserved
7:0	RO	40	CP	**Capability Pointer:** Pointer to first capability structure at 40h

13.8.15 INTL: Interrupt Line and Interrupt Pin

CSR Register Name: **INTL**: Interrupt Line and Interrupt Pin				
Bar: PCI Cfg		**Reset:** host_rst_b		**Offset Start:** 3C / **Offset End:** 3F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:16	RO	0	Reserved	Reserved
15:8	RO	1	INTPIN	**Interrupt Pin.** North Peak uses a single INTx interrupt bonded to INTA
7:0	RW	FF	INTL	**Interrupt Line: Hardware does not use this field.** Rather it is programmed by system software and device drivers to communicate interrupt line routing information

13.8.16 MSICID: MSI Capability ID,MSI Next Capability Pointer,MSI Message Control Register

CSR Register Name: MSICID: MSI Capability ID,MSI Next Capability Pointer,MSI Message Control Register				
Bar: PCI Cfg		**Reset:** host_rst_b		**Offset Start:** 40 / **Offset End:** 43
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:24	RO	0	Reserved	Reserved
23	RO	1	BAC64	**64-bit Address Capable:** North Peak is capable of generating 64-bit memory addresses
22:20	RW	0	MME	**Multiple Message Enable:** Indicates the number of messages allocated to the device
19:17	RO	0	MMC	**Multiple Message Capable:** Value of 0 indicates the device only support single interrupt message
16	RW	0	MSIE	**MSI Enable:** If set, MSI is enabled and the legacy interrupts messages (over the sideband fabric) will not be generated
15:8	RO	0	MSINCP	**Multiple Message Enable:** Indicates the number of messages allocated to the device
7:0	RO	5	MSICID	MSI Capability ID with a value of 05h indicating the presence of the MSI capability register set

13.8.17 MSILMA: MSI Lower Message Address

CSR Register Name: MSILMA: MSI Lower Message Address				
Bar: PCI Cfg		**Reset:** host_rst_b		**Offset Start:** 44 / **Offset End:** 47
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:2	RW	0	MSILMA	**MSI Message Lower Address:** Lower 32-bits of system software assigned message address to the device with bits[1:0] always cleared indicating message address has to always be DW aligned
1:0	RO	0	Reserved	Value of 0 indicates the memory address is always DW aligned

13.8.18 MSIUMA: MSI Upper Message Address

<table>
<tr><td colspan="6"><b>CSR Register Name: MSIUMA</b>: MSI Upper Message Address</td></tr>
<tr><td colspan="2"><b>Bar:</b> PCI Cfg</td><td colspan="2"><b>Reset:</b> host_rst_b</td><td><b>Offset Start:</b> 48</td><td><b>Offset End:</b> 4b</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>31:0</td><td>RW</td><td>0</td><td>MSIUMA</td><td colspan="2"><b>MSI Message Upper Address:</b> Upper 32-bits of system software assigned message address to the device</td></tr>
</table>

13.8.19 MSIMD: MSI Message Data

<table>
<tr><td colspan="6"><b>CSR Register Name: MSIMD</b>: MSI Message Data</td></tr>
<tr><td colspan="2"><b>Bar:</b> PCI Cfg</td><td colspan="2"><b>Reset:</b> host_rst_b</td><td><b>Offset Start:</b> 4c</td><td><b>Offset End:</b> 4f</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>31:16</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">Reserved</td></tr>
<tr><td>15:0</td><td>RW</td><td>0</td><td>MSIMD</td><td colspan="2"><b>MSI Message Data: 16-bit message data pattern assigned by the system software to the device.</b> When MSI is generated the actual data is 32-bit and the upper 16 bits are always 0</td></tr>
</table>

13.8.20 FW_LBAR: Firmware Lower Bar

<table>
<tr><td colspan="6"><b>CSR Register Name: FW_LBAR</b>: Firmware Lower Bar</td></tr>
<tr><td colspan="2"><b>Bar:</b> PCI Cfg</td><td colspan="2"><b>Reset:</b> host_rst_b</td><td><b>Offset Start:</b> 70</td><td><b>Offset End:</b> 73</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>31:18</td><td>RW</td><td>0</td><td>LADDR</td><td colspan="2"><b>Lower Base Address:</b> Lower programmable Base Address</td></tr>
</table>

CSR Register Name: FW_LBAR: Firmware Lower Bar				
Bar: PCI Cfg		**Reset:** host_rst_b		**Offset Start:** 70 **Offset End:** 73
Bits	**Access**	**Default**	**Label**	**Bit Description**
17:4	RO	0	Reserved	Reserved: Hardwired to 0 to indicate the memory space size required by this FW_BAR MMIO space is given by the following formula with 5 <= N <= 30 (this allows for up to 2GB of address space) Size=2^N bytes Where N can be computed from the number of implemented FW masters (FW_MSTR_STP -FW_MSTR_STRT)), the number of channels per master (SW_CHCNT), and the size of each channel which is fixed at 64B. To this end, N needs to be selected such that the MMIO space size is equal to 64 * (FW_MSTR_STP-FW_MSTR_STRT) * SW_CHCNT).
3	RO	0	PF	**Prefetchable:** Value of 0 indicates the BAR cannot be prefetched
2:1	RO	2	ADRNG	Address Range: Value of 0x2 indicates that the BAR is located anywhere system memory space (i.e. 64-bit addressing)
0	RO	0	SPTY	**Space Type:** Value of 0 indicates the BAR is located in memory space

13.8.21 FW_UBAR: Firmware Upper Bar

CSR Register Name: FW_UBAR: Firmware Upper Bar				
Bar: PCI Cfg		**Reset:** host_rst_b		**Offset Start:** 74 **Offset End:** 77
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RW	0	UADDR	**Upper Base Address:** Upper programmable Base Address

13.8.22 NPKDSC: NPK Device Specific Control and Device Specific Status

CSR Register Name: NPKDSC: NPK Device Specific Control and Device Specific Status			
Bar: PCI Cfg	**Reset:** host_rst_b	**Offset Start:** 80	**Offset End:** 83

Bits	Access	Default	Label	Bit Description
31:16	RO	0	Reserved	Reserved
15:11	RO	0	Reserved	Reserved
10	RW/1C	0	URD	**Unsupported Request Detect:** This bit is set when an unsupported request is detected
9	RW/1C	0	CTOE	**CCB Timeout Timer Expired:** This bit is asserted if the Inbound CCB timeout timer has expired at least once
8	RW/1C	0	STOE	**Switch Timeout Timer Expired:** This bit is asserted if the Inbound Switch timeout timer has expired at least once
7:4	RO	0	Reserved	Reserved
3	RW	0	URRE	**Unsupported Request Reporting Enable:** When set, this bit enables the reporting unsupported requests as system errors
2	RW/1C	0	LIA	**Legacy Interrupt Asserted: This bit is set when an assert INTx message has been send over sideband.** Writing a 1 to this bit will send the dessert message over sideband as well
1	WO	0	FLR	**Software Reset: This is North Peak s software controlled based functional level reset.** Writing a 1 to this bit will assert the reset signals. Reading this bit will always return a zero
0	RW	1	RSVD	Reserved

13.8.23 ISTOT: Inbound Switch Timeout Timer

CSR Register Name: ISTOT: Inbound Switch Timeout Timer			
Bar: PCI Cfg	**Reset:** host_rst_b	**Offset Start:** 84	**Offset End:** 87

Bits	Access	Default	Label	Bit Description
31:0	RW	32FF	TOT	**Inbound Switch Timeout Timer Value.** Timeout timer value used to protect the Intel(R) Trace Hub from inbound transactions that are functionally not supported. A value of zero programmed into this register will disable the timer and it will never timeout. Note: It is important to note that setting this value to a small value can result in an unpredicted behavior of the system is addressing certain functional blocks within the Intel(R) Trace Hub.

13.8.24 ICTOT: Inbound CCB Timeout Timer

CSR Register Name: ICTOT: Inbound CCB Timeout Timer			
Bar: PCI Cfg	**Reset:** host_rst_b	**Offset Start:** 88	**Offset End:** 8B

Bits	Access	Default	Label	Bit Description
31:0	RW	7FF	TOT	**Inbound CCB Timeout Timer Value.** Timeout timer value used to protect NPK from inbound transactions that are functionally not supported. A value of zero programmed into this register will disable the timer and it will never timeout. Note: It is important to note that setting this value to a small value can result in an unpredicted behavior of the system is addressing certain functional blocks within NPK.

13.8.25 NPKDSD: NPK Device Specific Defeature

CSR Register Name: NPKDSD: NPK Device Specific Defeature			
Bar: PCI Cfg	**Reset:** host_rst_b	**Offset Start:** 90	**Offset End:** 93

Bits	Access	Default	Label	Bit Description
0	RW	0	FON	**Force On:** If set, this bit will prevent the Intel® Trace Hub logic from gating its clock or de-asserting its power ok signals

13.9 PTI Registers

13.9.1 PTI Registers Summary

<table>
<tr><th colspan="4">PTI Registers</th></tr>
<tr><th>Offset Start</th><th>Offset End</th><th>Symbol</th><th>Register Name/Function</th></tr>
<tr><td>1C00</td><td>1C03</td><td>PTI_CTL</td><td>PTI Control Register</td></tr>
</table>

13.9.2 PTI_CTL: PTI Control Register

<table>
<tr><td colspan="6"><b>CSR Register Name: PTI_CTL</b>: PTI Control Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 1C00</td><td><b>Offset End:</b> 1C03</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:24</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">PTI_CTL Reserved</td></tr>
<tr><td>23:20</td><td>RW</td><td>0</td><td>PATGENMOD</td><td colspan="2">23: Reserved
22-20: Pattern Generation Mode: This field selects one of the five MIPI-PTI training/calibration patterns to drive onto the PTI Port pins. See PTI HAS for details.
001: Pattern 1
010: Pattern 2
011: Pattern 3
100: Pattern 4
101: Pattern 5
110: Pattern 6
All others reserved for future use.</td></tr>
<tr><td>19:16</td><td>RW</td><td>1</td><td>PTICLKDIV</td><td colspan="2">19,18: Reserved.
17,16: Clock Divider: This field selects the scaled down version of the clock to use for the trace clock and send to GPIO pins:
00: 1x (Use North Peak clock)
01: 2x (Use 1/2 North Peak clock)
10: 4x (Use 1/4 North Peak clock)
11: 8x (Use 1/8 North Peak clock</td></tr>
<tr><td>15:8</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">PTI_CTL Reserved</td></tr>
</table>

CSR Register Name: PTI_CTL: PTI Control Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 1C00	**Offset End:** 1C03

Bits	Access	Default	Label	Bit Description
7:4	RW	0	PTI_MODE	PTI Mode Select : This register selects the PTI port output Mode. The encoding for the field is as follows: 0001: 4-bit mode 0010: 8-bit mode 0100: 12-bit mode 1000: 16-bit mode All others reserved for future use
3:2	RO	0	Reserved	PTI_CTL Reserved
1	RW	0	PTI_FCEN	**PTI Free-Running Clock.** 0 : the trace_clock acts as a strobe which only toggles when valid data is present on trace_data pins. 1 : the trace_clock will act as a free-running clock
0	RW	0	PTI_EN	PTI_EN : This bit allows the PTI output mode bits(PTI_OUT_MODE_SEL) that propagate to the PTI IO logic to be enabled. It is also used to clock gate the PTI controller when no used

13.10 SoCHAP Registers

13.10.1 SoCHAP Registers Summary

SoCHAP Registers			
Offset Start	**Offset End**	**Symbol**	**Register Name/Function**
5000	5003	SDC	SoCHAP Device Capabilities
5004	5007	SSS	SoCHAP Software Register
5008	500B	CEC_0	Customizable Event Creation 0
500C	500F	CEC_1	Customizable Event Creation 1
5010	5013	CEC_2	Customizable Event Creation 2
5014	5017	CEC_3	Customizable Event Creation 3

SoCHAP Registers			
Offset Start	**Offset End**	**Symbol**	**Register Name/Function**
5018	501B	CEC_4	Customizable Event Creation 4
501C	501F	CEC_5	Customizable Event Creation 5
5020	5023	CEC_6	Customizable Event Creation 6
5024	5027	CEC_7	Customizable Event Creation 7
5028	502B	SOCHAPCMD_0	SoCHAP Command Register 0
502C	502F	SOCHAPEV_0	SoCHAP Events Register 0
5030	5033	SOCHAPSTAT_0	SoCHAP Status Register 0
5034	5037	SOCHAPDATA_0	SoCHAP Data Register 0
5038	503B	SOCHAPCMD_1	SoCHAP Command Register 1
503C	503F	SOCHAPEV_1	SoCHAP Events Register 1
5040	5043	SOCHAPSTAT_1	SoCHAP Status Register 1
5044	5047	SOCHAPDATA_1	SoCHAP Data Register 1
5048	504B	SOCHAPCMD_2	SoCHAP Command Register 2
504C	504F	SOCHAPEV_2	SoCHAP Events Register 2
5050	5053	SOCHAPSTAT_2	SoCHAP Status Register 2
5054	5057	SOCHAPDATA_2	SoCHAP Data Register 2
5058	505B	SOCHAPCMD_3	SoCHAP Command Register 3
505C	505F	SOCHAPEV_3	SoCHAP Events Register 3
5060	5063	SOCHAPSTAT_3	SoCHAP Status Register 3
5064	5067	SOCHAPDATA_3	SoCHAP Data Register 3

SoCHAP Registers			
Offset Start	**Offset End**	**Symbol**	**Register Name/Function**
5068	506B	SOCHAPCMD_4	SoCHAP Command Register 4
506C	506F	SOCHAPEV_4	SoCHAP Events Register 4
5070	5073	SOCHAPSTAT_4	SoCHAP Status Register 4
5074	5077	SOCHAPDATA_4	SoCHAP Data Register 4
5078	507B	SOCHAPCMD_5	SoCHAP Command Register 5
507C	507F	SOCHAPEV_5	SoCHAP Events Register 5
5080	5083	SOCHAPSTAT_5	SoCHAP Status Register 5
5084	5087	SOCHAPDATA_5	SoCHAP Data Register 5
5088	508B	SOCHAPCMD_6	SoCHAP Command Register 6
508C	508F	SOCHAPEV_6	SoCHAP Events Register 6
5090	5093	SOCHAPSTAT_6	SoCHAP Status Register 6
5094	5097	SOCHAPDATA_6	SoCHAP Data Register 6
5098	509B	SOCHAPCMD_7	SoCHAP Command Register 7
509C	509F	SOCHAPEV_7	SoCHAP Events Register 7
50A0	50A3	SOCHAPSTAT_7	SoCHAP Status Register 7
50A4	50A7	SOCHAPDATA_7	SoCHAP Data Register 7
50A8	50AB	SOCHAPPKTCTRL	SoCHAP Packet Control Register

13.10.2 SDC: SoCHAP Device Capabilities

CSR Register Name: **SDC**: SoCHAP Device Capabilities				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 5000 **Offset End:** 5003
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:16	RO	0	Reserved	SDC Reserved
15	RW	0	CE	**CHAP Enable: This register enables the SoCHAP block.** When this bit is cleared, the logic is clock gated with the exception of this register.
14	RW	0	IU	**In Use: Software Mutex bit.** After a RESET, a read to this bit returns a 0. After the first read, subsequent reads will return a 1. A write of a 1 to this bit will reset the next read value to 0. Writing a 0 to this bit has no effect Software can poll this bit until it reads a 0, and will then own the usage of the corresponding counter. This bit has no other effect on any SoCHAP Counter registers, and is only used as a semaphore among various independent software threads that may need to utilize this performance counter. Software that reads this register, but does not intend to claim exclusive access, must write a 1 to this bit (if it reads a 0), in order to allow other software threads to claim it. This bit is particularly useful when supporting multi-threaded environments.
13	RO	0	Reserved	SDC Reserved
12:8	RO	F	VR	**VIS Resources: This register indicates how many VIS input signals are being fed into this block.** Up to 32 inputs are supported through VIS. Note, actual resources = VR + 1.
7:4	RO	7	CR	**Counter Resources: This value exposes the number of counters available to the SoCHAP block.** For each counter exposed, there exists 4 registers for command, event, status, and data. The default value of 0h corresponds to 1 SoCHAP counter resource. Note, actual resources = CR + 1.

CSR Register Name: SDC: SoCHAP Device Capabilities				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 5000	**Offset End:** 5003
Bits	**Access**	**Default**	**Label**	**Bit Description**
3:0	RO	7	CECR	**Customizable Event Creation Resources: This register exposes the number of custom events available.** The default value of 0h corresponds to one SoCHAP customizable event resource. For each customizable event 1 control register is present. Note, actual resources = CECR + 1.

13.10.3 SSS: SoCHAP Software Register

CSR Register Name: SSS: SoCHAP Software Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 5004	**Offset End:** 5007
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RW	0	SRTCHPAD	Scratchpad Register: This register is made available to keep track of the Counter and Custom event usage within software.

13.10.4 CEC_0: Customizable Event Creation 0

CSR Register Name: CEC_0: Customizable Event Creation 0			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 5008	**Offset End:** 500B

Bits	Access	Default	Label	Bit Description
31:24	RW/P	0	DLYSEL	**Delay Selection: Bit field LSB corresponds to VIS bit 0.** 0: The input resource bit is considered in event calculations. 1: The VIS bit is delayed by 1 clock before considering it in event calculations. This is particularly useful for doing state machine arc coverage. For example, VIS bits 3:0 and VIS 7:4 could be programmed to the same 4 present state, state machine signals. The appropriate inversion selections would be made depending on which state transition is of interest. Bits 31:28 in the delay selection would be programmed to 1111, indicating use a pipe delayed version of the state signals. The resulting AND of the now preconditioned VIS 7:4 and VIS 3:0 signals would indicate the number of times the arc of interest was taken. This could be recorded with the SoCHAP counters
23:16	RW/P	0	MSKE	**Mask Enable: Bit field LSB corresponds to input resource bit 0.** These 8 bits are used to mask off entries from the comparison. For each bit: 0: This input resource bit is considered in event calculations. 1: This input resource bit is ignored in event calculations.

<table>
<tr><td colspan="5"><b>CSR Register Name: CEC_0</b>: Customizable Event Creation 0</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5008
<b>Offset End:</b> 500B</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th>Bit Description</th></tr>
<tr><td>15:8</td><td>RW/P</td><td>0</td><td>CMPVAL</td><td>Compare Value: Bit field LSB corresponds to input resource bit 0. This field is loaded to compare against the 8 input resource signals that are fed into this block. The type of comparison that is done is controlled by the Compare Function.
When the compare function is true, then the signal for the input resource event is asserted. This in turn can be counted by any of the SoCHAP counters.
When the CEC is programmed for alternative mode (CMPFCN == ALT), this register is used to control the type of mode that is being used.
CMPVAL == 8b0: ONES mode. In this mode, the number of signals that are asserted are added up and passed as a binary value to the counters as the event.Example: input = b11110000, count by 4.
CMPVAL == 8b1: COUNTER mode. The masked value is passed directly to the counters as a binary value. This allows the counters to increment or decrement by up to 255 at any one time. Example: input = b11110000, count by 240.
CMPVAL > 8b1: Reserved for future use</td></tr>
<tr><td>7:6</td><td>RW/P</td><td>0</td><td>CVG</td><td>Clock Valid Generation: This selects which mode is being used for qualification of the data with the corresponding valid bits. This is needed since signals from different domains may not have their valid pulses aligned.
00: Only evaluate expression when all valid signals are asserted. This mode is used when signals come from the same clock domain.
01: Evaluate expression when any of the expressions corresponding valid pulses are asserted.
10: Reserved.
11: Bypass the valid and evaluate expression based on the raw data alone. The output of the Custom Event block asserts its valid signal continuously.</td></tr>
</table>

13.10.5 CEC_1: Customizable Event Creation 1

<table>
<tr><td colspan="5"><b>CSR Register Name: CEC_1</b>: Customizable Event Creation 1</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 500C
<b>Offset End:</b> 500F</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th>Bit Description</th></tr>
<tr><td>31:24</td><td>RW/P</td><td>0</td><td>DLYSEL</td><td><b>Delay Selection: Bit field LSB corresponds to VIS bit 0.</b>
0: The input resource bit is considered in event calculations.
1: The VIS bit is delayed by 1 clock before considering it in event calculations.
This is particularly useful for doing state machine arc coverage.
For example, VIS bits 3:0 and VIS 7:4 could be programmed to the same 4 present state, state machine signals. The appropriate inversion selections would be made depending on which state transition is of interest. Bits 31:28 in the delay selection would be programmed to 1111, indicating use a pipe delayed version of the state signals.
The resulting AND of the now preconditioned VIS 7:4 and VIS 3:0 signals would indicate the number of times the arc of interest was taken. This could be recorded with the SoCHAP counters</td></tr>
<tr><td>23:16</td><td>RW/P</td><td>0</td><td>MSKE</td><td><b>Mask Enable: Bit field LSB corresponds to input resource bit 0.</b>
These 8 bits are used to mask off entries from the comparison. For each bit:
0: This input resource bit is considered in event calculations.
1: This input resource bit is ignored in event calculations.</td></tr>
</table>

<table>
<tr><td colspan="6"><b>CSR Register Name:</b> <b>CEC_1</b>: Customizable Event Creation 1</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 500C</td><td><b>Offset End:</b> 500F</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>15:8</td><td>RW/P</td><td>0</td><td>CMPVAL</td><td colspan="2">Compare Value: Bit field LSB corresponds to input resource bit 0. This field is loaded to compare against the 8 input resource signals that are fed into this block. The type of comparison that is done is controlled by the Compare Function.
When the compare function is true, then the signal for the input resource event is asserted. This in turn can be counted by any of the SoCHAP counters.
When the CEC is programmed for alternative mode (CMPFCN == ALT), this register is used to control the type of mode that is being used.
CMPVAL == 8b0: ONES mode. In this mode, the number of signals that are asserted are added up and passed as a binary value to the counters as the event.Example: input = b11110000, count by 4.
CMPVAL == 8b1: COUNTER mode. The masked value is passed directly to the counters as a binary value. This allows the counters to increment or decrement by up to 255 at any one time. Example: input = b11110000, count by 240.
CMPVAL > 8b1: Reserved for future use</td></tr>
<tr><td>7:6</td><td>RW/P</td><td>0</td><td>CVG</td><td colspan="2">Clock Valid Generation: This selects which mode is being used for qualification of the data with the corresponding valid bits. This is needed since signals from different domains may not have their valid pulses aligned.
00: Only evaluate expression when all valid signals are asserted. This mode is used when signals come from the same clock domain.
01: Evaluate expression when any of the expressions corresponding valid pulses are asserted.
10: Reserved.
11: Bypass the valid and evaluate expression based on the raw data alone. The output of the Custom Event block asserts its valid signal continuously.</td></tr>
</table>

13.10.6 CEC_2: Customizable Event Creation 2

CSR Register Name: CEC_2: Customizable Event Creation 2				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 5010	**Offset End:** 5013
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:24	RW/P	0	DLYSEL	**Delay Selection: Bit field LSB corresponds to VIS bit 0.** 0: The input resource bit is considered in event calculations. 1: The VIS bit is delayed by 1 clock before considering it in event calculations. This is particularly useful for doing state machine arc coverage. For example, VIS bits 3:0 and VIS 7:4 could be programmed to the same 4 present state, state machine signals. The appropriate inversion selections would be made depending on which state transition is of interest. Bits 31:28 in the delay selection would be programmed to 1111, indicating use a pipe delayed version of the state signals. The resulting AND of the now preconditioned VIS 7:4 and VIS 3:0 signals would indicate the number of times the arc of interest was taken. This could be recorded with the SoCHAP counters
23:16	RW/P	0	MSKE	**Mask Enable: Bit field LSB corresponds to input resource bit 0.** These 8 bits are used to mask off entries from the comparison. For each bit: 0: This input resource bit is considered in event calculations. 1: This input resource bit is ignored in event calculations.

<table>
<tr><td colspan="5"><b>CSR Register Name:</b> <b>CEC_2</b>: Customizable Event Creation 2</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5010</td><td><b>Offset End:</b> 5013</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>15:8</td><td>RW/P</td><td>0</td><td>CMPVAL</td><td colspan="2">Compare Value: Bit field LSB corresponds to input resource bit 0. This field is loaded to compare against the 8 input resource signals that are fed into this block. The type of comparison that is done is controlled by the Compare Function.
When the compare function is true, then the signal for the input resource event is asserted. This in turn can be counted by any of the SoCHAP counters.
When the CEC is programmed for alternative mode (CMPFCN == ALT), this register is used to control the type of mode that is being used.
CMPVAL == 8b0: ONES mode. In this mode, the number of signals that are asserted are added up and passed as a binary value to the counters as the event.Example: input = b11110000, count by 4.
CMPVAL == 8b1: COUNTER mode. The masked value is passed directly to the counters as a binary value. This allows the counters to increment or decrement by up to 255 at any one time. Example: input = b11110000, count by 240.
CMPVAL > 8b1: Reserved for future use</td></tr>
<tr><td>7:6</td><td>RW/P</td><td>0</td><td>CVG</td><td colspan="2">Clock Valid Generation: This selects which mode is being used for qualification of the data with the corresponding valid bits. This is needed since signals from different domains may not have their valid pulses aligned. 00: Only evaluate expression when all valid signals are asserted. This mode is used when signals come from the same clock domain. 01: Evaluate expression when any of the expressions corresponding valid pulses are asserted. 10: Reserved. 11: Bypass the valid and evaluate expression based on the raw data alone. The output of the Custom Event block asserts its valid signal continuously.</td></tr>
</table>

<table>
<tr><td colspan="6"><b>CSR Register Name: CEC_2</b>: Customizable Event Creation 2</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5010</td><td><b>Offset End:</b> 5013</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>5:3</td><td>RW/P</td><td>0</td><td>IRN</td><td colspan="2"><b>Input Resource Number: Each Custom Event can handle up to 8 input lanes.</b> The mapping of the lanes is given by the CEC lane mapping fields in the SoCHAP capabilities register.
000: Resource #1 VIS lane 0 (xbar[7:0])
001: Resource #2 Previous CEC flopped path
010: Resource #3 CEC compare out [7:0]
011: Resource #4 Threshold Event [7:0]
100: Resource #5 VIS Lane 1 (xbar[15:8])
101: Resource #6 VIS Lane 2 (xbar[23:16])
110: Resource #7 CEC Compare out [15:8] or VIS Lane 3 if only 8 CEC Resources
111: Resource #8 Threshold Event [15:8] or VIS Lane 3 (xbar[31:24]) if only 8 Counter Resources</td></tr>
<tr><td>2:0</td><td>RW/P</td><td>0</td><td>CMPFCN</td><td colspan="2">Compare Function: 000: Compare and assert if any are equal (Can be used as OR function) 001: Compare and output signal if greater than 010: Compare and assert output if equal to (Can also be used as AND function) 011: Compare and assert output if greater than or equal 100: Compare and assert output if less than 101: Compare and assert output if not equal 110: Compare and assert output if less than or equal 111: Alternative mode (non-compare modes)This mode enables the alternative mode for the counters. These modes use the compare value (CMPVAL) as an encoding to determine what to do (see CMPVAL for alternative encodings). * NOTE:When using the additive mode, the delay selection bits must be set to not take the flopped version of the signals. Also, when used as the trigger event, the add event only uses bit 0 for the trigger</td></tr>
</table>

13.10.7 CEC_3: Customizable Event Creation 3

CSR Register Name: **CEC_3**: Customizable Event Creation 3			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 5014	**Offset End:** 5017

Bits	Access	Default	Label	Bit Description
31:24	RW/P	0	DLYSEL	**Delay Selection: Bit field LSB corresponds to VIS bit 0.** 0: The input resource bit is considered in event calculations. 1: The VIS bit is delayed by 1 clock before considering it in event calculations. This is particularly useful for doing state machine arc coverage. For example, VIS bits 3:0 and VIS 7:4 could be programmed to the same 4 present state, state machine signals. The appropriate inversion selections would be made depending on which state transition is of interest. Bits 31:28 in the delay selection would be programmed to 1111, indicating use a pipe delayed version of the state signals. The resulting AND of the now preconditioned VIS 7:4 and VIS 3:0 signals would indicate the number of times the arc of interest was taken. This could be recorded with the SoCHAP counters
23:16	RW/P	0	MSKE	**Mask Enable: Bit field LSB corresponds to input resource bit 0.** These 8 bits are used to mask off entries from the comparison. For each bit: 0: This input resource bit is considered in event calculations. 1: This input resource bit is ignored in event calculations.

CSR Register Name: CEC_3: Customizable Event Creation 3				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 5014 **Offset End:** 5017
Bits	**Access**	**Default**	**Label**	**Bit Description**
15:8	RW/P	0	CMPVAL	Compare Value: Bit field LSB corresponds to input resource bit 0. This field is loaded to compare against the 8 input resource signals that are fed into this block. The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the input resource event is asserted. This in turn can be counted by any of the SoCHAP counters. When the CEC is programmed for alternative mode (CMPFCN == ALT), this register is used to control the type of mode that is being used. CMPVAL == 8b0: ONES mode. In this mode, the number of signals that are asserted are added up and passed as a binary value to the counters as the event.Example: input = b11110000, count by 4. CMPVAL == 8b1: COUNTER mode. The masked value is passed directly to the counters as a binary value. This allows the counters to increment or decrement by up to 255 at any one time. Example: input = b11110000, count by 240. CMPVAL > 8b1: Reserved for future use
7:6	RW/P	0	CVG	Clock Valid Generation: This selects which mode is being used for qualification of the data with the corresponding valid bits. This is needed since signals from different domains may not have their valid pulses aligned. 00: Only evaluate expression when all valid signals are asserted. This mode is used when signals come from the same clock domain. 01: Evaluate expression when any of the expressions corresponding valid pulses are asserted. 10: Reserved. 11: Bypass the valid and evaluate expression based on the raw data alone. The output of the Custom Event block asserts its valid signal continuously.

13.10.8 CEC_4: Customizable Event Creation 4

CSR Register Name: **CEC_4**: Customizable Event Creation 4				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 5018 **Offset End:** 501B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:24	RW/P	0	DLYSEL	**Delay Selection: Bit field LSB corresponds to VIS bit 0.** 0: The input resource bit is considered in event calculations. 1: The VIS bit is delayed by 1 clock before considering it in event calculations. This is particularly useful for doing state machine arc coverage. For example, VIS bits 3:0 and VIS 7:4 could be programmed to the same 4 present state, state machine signals. The appropriate inversion selections would be made depending on which state transition is of interest. Bits 31:28 in the delay selection would be programmed to 1111, indicating use a pipe delayed version of the state signals. The resulting AND of the now preconditioned VIS 7:4 and VIS 3:0 signals would indicate the number of times the arc of interest was taken. This could be recorded with the SoCHAP counters
23:16	RW/P	0	MSKE	**Mask Enable: Bit field LSB corresponds to input resource bit 0.** These 8 bits are used to mask off entries from the comparison. For each bit: 0: This input resource bit is considered in event calculations. 1: This input resource bit is ignored in event calculations.

CSR Register Name: CEC_4: Customizable Event Creation 4				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 5018	**Offset End:** 501B
Bits	**Access**	**Default**	**Label**	**Bit Description**
15:8	RW/P	0	CMPVAL	Compare Value: Bit field LSB corresponds to input resource bit 0. This field is loaded to compare against the 8 input resource signals that are fed into this block. The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the input resource event is asserted. This in turn can be counted by any of the SoCHAP counters. When the CEC is programmed for alternative mode (CMPFCN == ALT), this register is used to control the type of mode that is being used. CMPVAL == 8b0: ONES mode. In this mode, the number of signals that are asserted are added up and passed as a binary value to the counters as the event.Example: input = b11110000, count by 4. CMPVAL == 8b1: COUNTER mode. The masked value is passed directly to the counters as a binary value. This allows the counters to increment or decrement by up to 255 at any one time. Example: input = b11110000, count by 240. CMPVAL > 8b1: Reserved for future use
7:6	RW/P	0	CVG	Clock Valid Generation: This selects which mode is being used for qualification of the data with the corresponding valid bits. This is needed since signals from different domains may not have their valid pulses aligned. 00: Only evaluate expression when all valid signals are asserted. This mode is used when signals come from the same clock domain. 01: Evaluate expression when any of the expressions corresponding valid pulses are asserted. 10: Reserved. 11: Bypass the valid and evaluate expression based on the raw data alone. The output of the Custom Event block asserts its valid signal continuously.

13.10.9 CEC_5: Customizable Event Creation 5

<table>
<tr><td colspan="6"><b>CSR Register Name: CEC_5</b>: Customizable Event Creation 5</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 501C</td><td><b>Offset End:</b> 501F</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>31:24</td><td>RW/P</td><td>0</td><td>DLYSEL</td><td colspan="2"><b>Delay Selection: Bit field LSB corresponds to VIS bit 0.</b>

0: The input resource bit is considered in event calculations.

1: The VIS bit is delayed by 1 clock before considering it in event calculations.
This is particularly useful for doing state machine arc coverage.
For example, VIS bits 3:0 and VIS 7:4 could be programmed to the same 4 present state, state machine signals. The appropriate inversion selections would be made depending on which state transition is of interest. Bits 31:28 in the delay selection would be programmed to 1111, indicating use a pipe delayed version of the state signals.
The resulting AND of the now preconditioned VIS 7:4 and VIS 3:0 signals would indicate the number of times the arc of interest was taken. This could be recorded with the SoCHAP counters</td></tr>
<tr><td>23:16</td><td>RW/P</td><td>0</td><td>MSKE</td><td colspan="2"><b>Mask Enable: Bit field LSB corresponds to input resource bit 0.</b>
These 8 bits are used to mask off entries from the comparison. For each bit:

0: This input resource bit is considered in event calculations.

1: This input resource bit is ignored in event calculations.</td></tr>
</table>

CSR Register Name: CEC_5: Customizable Event Creation 5				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 501C	**Offset End:** 501F
Bits	**Access**	**Default**	**Label**	**Bit Description**
15:8	RW/P	0	CMPVAL	Compare Value: Bit field LSB corresponds to input resource bit 0. This field is loaded to compare against the 8 input resource signals that are fed into this block. The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the input resource event is asserted. This in turn can be counted by any of the SoCHAP counters. When the CEC is programmed for alternative mode (CMPFCN == ALT), this register is used to control the type of mode that is being used. CMPVAL == 8b0: ONES mode. In this mode, the number of signals that are asserted are added up and passed as a binary value to the counters as the event.Example: input = b11110000, count by 4. CMPVAL == 8b1: COUNTER mode. The masked value is passed directly to the counters as a binary value. This allows the counters to increment or decrement by up to 255 at any one time. Example: input = b11110000, count by 240. CMPVAL > 8b1: Reserved for future use
7:6	RW/P	0	CVG	Clock Valid Generation: This selects which mode is being used for qualification of the data with the corresponding valid bits. This is needed since signals from different domains may not have their valid pulses aligned. 00: Only evaluate expression when all valid signals are asserted. This mode is used when signals come from the same clock domain. 01: Evaluate expression when any of the expressions corresponding valid pulses are asserted. 10: Reserved. 11: Bypass the valid and evaluate expression based on the raw data alone. The output of the Custom Event block asserts its valid signal continuously.

13.10.10 CEC_6: Customizable Event Creation 6

CSR Register Name: CEC_6: Customizable Event Creation 6					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 5020	**Offset End:** 5023

Bits	Access	Default	Label	Bit Description
31:24	RW/P	0	DLYSEL	**Delay Selection: Bit field LSB corresponds to VIS bit 0.** 0: The input resource bit is considered in event calculations. 1: The VIS bit is delayed by 1 clock before considering it in event calculations. This is particularly useful for doing state machine arc coverage. For example, VIS bits 3:0 and VIS 7:4 could be programmed to the same 4 present state, state machine signals. The appropriate inversion selections would be made depending on which state transition is of interest. Bits 31:28 in the delay selection would be programmed to 1111, indicating use a pipe delayed version of the state signals. The resulting AND of the now preconditioned VIS 7:4 and VIS 3:0 signals would indicate the number of times the arc of interest was taken. This could be recorded with the SoCHAP counters
23:16	RW/P	0	MSKE	**Mask Enable: Bit field LSB corresponds to input resource bit 0.** These 8 bits are used to mask off entries from the comparison. For each bit: 0: This input resource bit is considered in event calculations. 1: This input resource bit is ignored in event calculations.

CSR Register Name: CEC_6: Customizable Event Creation 6				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 5020	**Offset End:** 5023
Bits	**Access**	**Default**	**Label**	**Bit Description**
15:8	RW/P	0	CMPVAL	Compare Value: Bit field LSB corresponds to input resource bit 0. This field is loaded to compare against the 8 input resource signals that are fed into this block. The type of comparison that is done is controlled by the Compare Function. When the compare function is true, then the signal for the input resource event is asserted. This in turn can be counted by any of the SoCHAP counters. When the CEC is programmed for alternative mode (CMPFCN == ALT), this register is used to control the type of mode that is being used. CMPVAL == 8b0: ONES mode. In this mode, the number of signals that are asserted are added up and passed as a binary value to the counters as the event.Example: input = b11110000, count by 4. CMPVAL == 8b1: COUNTER mode. The masked value is passed directly to the counters as a binary value. This allows the counters to increment or decrement by up to 255 at any one time. Example: input = b11110000, count by 240. CMPVAL > 8b1: Reserved for future use
7:6	RW/P	0	CVG	Clock Valid Generation: This selects which mode is being used for qualification of the data with the corresponding valid bits. This is needed since signals from different domains may not have their valid pulses aligned. 00: Only evaluate expression when all valid signals are asserted. This mode is used when signals come from the same clock domain. 01: Evaluate expression when any of the expressions corresponding valid pulses are asserted. 10: Reserved. 11: Bypass the valid and evaluate expression based on the raw data alone. The output of the Custom Event block asserts its valid signal continuously.

13.10.11 CEC_7: Customizable Event Creation 7

CSR Register Name: CEC_7: Customizable Event Creation 7				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 5024	**Offset End:** 5027
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:24	RW/P	0	DLYSEL	**Delay Selection: Bit field LSB corresponds to VIS bit 0.** 0: The input resource bit is considered in event calculations. 1: The VIS bit is delayed by 1 clock before considering it in event calculations. This is particularly useful for doing state machine arc coverage. For example, VIS bits 3:0 and VIS 7:4 could be programmed to the same 4 present state, state machine signals. The appropriate inversion selections would be made depending on which state transition is of interest. Bits 31:28 in the delay selection would be programmed to 1111, indicating use a pipe delayed version of the state signals. The resulting AND of the now preconditioned VIS 7:4 and VIS 3:0 signals would indicate the number of times the arc of interest was taken. This could be recorded with the SoCHAP counters
23:16	RW/P	0	MSKE	**Mask Enable: Bit field LSB corresponds to input resource bit 0.** These 8 bits are used to mask off entries from the comparison. For each bit: 0: This input resource bit is considered in event calculations. 1: This input resource bit is ignored in event calculations.

<table>
<tr><td colspan="5"><b>CSR Register Name:</b> <b>CEC_7</b>: Customizable Event Creation 7</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5024   <b>Offset End:</b> 5027</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th>Bit Description</th></tr>
<tr><td>15:8</td><td>RW/P</td><td>0</td><td>CMPVAL</td><td>Compare Value: Bit field LSB corresponds to input resource bit 0. This field is loaded to compare against the 8 input resource signals that are fed into this block. The type of comparison that is done is controlled by the Compare Function.
When the compare function is true, then the signal for the input resource event is asserted. This in turn can be counted by any of the SoCHAP counters.
When the CEC is programmed for alternative mode (CMPFCN == ALT), this register is used to control the type of mode that is being used.
CMPVAL == 8b0: ONES mode. In this mode, the number of signals that are asserted are added up and passed as a binary value to the counters as the event.Example: input = b11110000, count by 4.
CMPVAL == 8b1: COUNTER mode. The masked value is passed directly to the counters as a binary value. This allows the counters to increment or decrement by up to 255 at any one time. Example: input = b11110000, count by 240.
CMPVAL > 8b1: Reserved for future use</td></tr>
<tr><td>7:6</td><td>RW/P</td><td>0</td><td>CVG</td><td>Clock Valid Generation: This selects which mode is being used for qualification of the data with the corresponding valid bits. This is needed since signals from different domains may not have their valid pulses aligned.
00: Only evaluate expression when all valid signals are asserted. This mode is used when signals come from the same clock domain.
01: Evaluate expression when any of the expressions corresponding valid pulses are asserted.
10: Reserved.
11: Bypass the valid and evaluate expression based on the raw data alone. The output of the Custom Event block asserts its valid signal continuously.</td></tr>
</table>

13.10.12 SOCHAPCMD_0: SoCHAP Command Register 0

CSR Register Name: SOCHAPCMD_0: SoCHAP Command Register 0				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 5028	**Offset End:** 502B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:30	RO	0	Reserved	SOCHAPCMD_0 Reserved
29:28	RW	0	Reserved	SOCHAPCMD_0 Reserved
27	RW/P	0	PE	**PMI Enable:** **0: PMIout is not asserted when the indicators are valid.** 1: The PMIout signal is asserted when the conditions enabled via the indicator enable bits are true.
26	RW/P	0	OUIE	Overflow/Underflow Indicator Enable: 0: No indication provided when a counter overflow or underflow occurs except for setting the Overflow/Underflow Indicator (OUI) status bit. 1: When the overflow/underflow condition is true, the counter will set the status bit and possibly assert the PMIout pin, which depending on SoC implementation, can be fed to another logic block or interrupt source.
25	RW/P	0	CTIE	Command Trigger Indicator Enable: 0: No indication provided when a command is triggered except for setting the Command Trigger Indicator (CTI) status bit. 1: When the command trigger condition is true, the counter will set the status bit and possibly assert the PMIout pin, which depending on SoC implementation, can be fed to another logic block or interrupt source.

CSR Register Name: SOCHAPCMD_0: SoCHAP Command Register 0				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 5028	**Offset End:** 502B
Bits	**Access**	**Default**	**Label**	**Bit Description**
24	RW/P	0	THIE	Threshold Indicator Enable: 0: No indication provided when threshold condition is true except for setting the Threshold Indicator (TI) status bit. 1: When the threshold condition is true, the counter will set the status bit and possibly assert the PMIout pin, which depending on SoC implementation, can be fed to another logic block or interrupt source.
23:21	RW/P	0	CC	Condition Code: This field contains the code that indicates what type of threshold compare is done between the counter and the data register. For all non-0 values of this field, the counter s data register will contain the threshold value. The outcome of this compare will generate a threshold event and potentially an interrupt if that capability is enabled.Bit 23 is for less than (<) Bit 22 is for equal (=) Bit 21 is for greater than (>) Select the proper bit mask for desired threshold condition: 000: False (no threshold compare) 001: Greater Than 010: Equal 011: Greater Than or Equal 100: Less Than 101: Not Equal 110: Less Than or Equal 111: True (always generate threshold event)

CSR Register Name: SOCHAPCMD_0: SoCHAP Command Register 0

Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 5028	**Offset End:** 502B

Bits	Access	Default	Label	Bit Description
20	RW/P	0	SAC	Select ALL Counters: This bit has a slightly different functoin in Counter #0 compared to the other counters. When this bit is set, the opcode in bits 19:16 is applied to all counters. When cleared, the opcode in bits 19:16 only apply to the current counter. The rest of the SoCHAP command register is not affected by the setting of this bit. 0: The opcode is applied only to the counter associated with this command register. 1: The opcode is applied to ALL counters. This means that every command register is written to with the same value that was written to this particular command register. This is particularly handy for resetting all counters with a single command or starting or stopping all counters simultaneously. This bit is only valid in Command Register 0 and should be reserved in all non-0 command registers. Globally executed commands (by setting this bit in Command Register 0) always override locally executed commands.

CSR Register Name: SOCHAPCMD_0: SoCHAP Command Register 0				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 5028	**Offset End:** 502B
Bits	**Access**	**Default**	**Label**	**Bit Description**
19:16	RW/P	0	OPCODE	Opcode: 0000: Stop. The corresponding counter does not count. 0001: Start. The corresponding counter begins counting. A counter increments by ICV+1 if the corresponding increment event occurs or decrements by DCV+1 if the corresponding decrement event occurs. All duration type events toggle every SoCHAP unit clock tick for which the event is true. The desired increment and decrement events must be selected before this command executes. 0010: Sample. The corresponding counter value is latched into the corresponding data register, which can then be accessed by reading the appropriate data register. Counter functionality is unaffected by a Sample Opcode. The counter continues to count without being reset. If the Condition Code is NOT False then the Data Register is not written when a sample takes place. In other words the Data Register will NOT be updated with counter value if the Condition Code is > 0. 0100: Reset. The corresponding counter is reset to 0000 0000h. The 32 bit wide counter allows 4 billion clock ticks or occurrences to be counted between sample commands. When the counter rolls over, the overflow status bit will be set in the corresponding status register. 0101: Restart. The corresponding counter resets, and begins counting again. This is essentially a Reset & Start command. This functionality will facilitate histogram creation by allowing an event to trigger to clear the counter and resume counting with no further intervention. 0110: Sample & Restart. The Sample command happens and is followed immediately by the Restart command. 1111: Preload. The corresponding counter is set to the value that is located in the associated Data Register. This facilitates rollover and overflow validation. The counter remains in the same state when preloaded. If the counter was counting before the preload was executed it will continue to count after the preload. It is software s responsibility to ensure that the counter is in the desired state (example: execute stop command) prior to issuing a preload command. All others reserved.

CSR Register Name: SOCHAPCMD_0: SoCHAP Command Register 0				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 5028 **Offset End:** 502B
Bits	**Access**	**Default**	**Label**	**Bit Description**
15	RO	0	Reserved	SOCHAPCMD_0 Reserved
14:13	RW	0	Reserved	SOCHAPCMD_0 Reserved
12	RW/P	0	TBV	**Trigger Bypass Valid Protocol: In this mode, the VIS valid protocol is not used.** The counter counts every clock regardless of valid data. Since signals from VIS may span multiple clocks this mode is not recommended for triggers since multiple back to back triggers will likely occur.
11:10	RW	0	Reserved	SOCHAPCMD_0 Reserved
9	RW/P	0	TTM	Threshold Trigger Mode: When this bit is set, if the operation of the trigger includes a sample (Sample, Sample and Restart opcodes), then the data value is only written if the threshold comparison is true. This mode can be used to store maximum and minimum values into the data register.
8:0	RW/P	0	CT	Command Trigger: This field contains the Event Selection Code (ESC) that the unit will be required to detect before executing the opcode. The previously programmed opcode continues to execute until this command trigger is detected. The ESC of 000h (the default value) is a special case. This special ESC causes the command to be triggered immediately upon a write to the command register. The special command trigger also causes the target command to be executed (triggered) once and only once. All other (non-0) ESCs cause the command to be re-executed every time the trigger is detected. Only one trigger can occur per VIS clock. If the trigger source clock is running faster than the VIS clock and more than one trigger event occurs per VIS clock, these additional triggers will be dropped. See the Event Tables for valid values. ESC values are limited to the range of 000-1FFh

13.10.13 SOCHAPEV_0: SoCHAP Events Register 0

CSR Register Name: SOCHAPEV_0: SoCHAP Events Register 0				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 502C	**Offset End:** 502F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW/P	0	DECOCE	Decrement Occurrence Count Enable: 0: Decrement Duration Count: the counter is decremented for each clock for which the decrement event signal is asserted logic high. 1: Decrement Occurrence Count: the counter is decremented each time a rising edge of the decrement event signal is detected.
30	RW	0	Reserved	SOCHAPEV_0 Reserved
29	RW/P	0	DECEVOVR	**Decrement Event Override: When this bit is set, the decrement event is tied to logical 1 .** This in turn causes the counter to count the number clocks that occurred in the native domain of the signal.
28	RW/P	0	DECBYP	**Decrement Bypass Valid Protocol: In this mode, the VIS valid protocol is not used.** The counter decrements each clock regardless of valid data.
27:25	RW/P	0	DECVAL	**Decrement Counting Value:By setting this, the decrement value used by the counter can be changed.** Note, decrement value = DECVAL+1. 000: Decrement by 1 001: Decrement by 2 010: Decrement by 3 ... 111: Decrement by 8
24:16	RW/P	0	DECE	Decrement Event: This field contains the Event Selection Code (ESC) that the unit will be required to detect before decrementing the associated counter. This field is only applicable for opcodes that put the counter in a state where it is counting. See the Event tables for valid values. This limits ESC codes to the range of 000-1FFh.

<table>
<tr><td colspan="6"><b>CSR Register Name: SOCHAPEV_0</b>: SoCHAP Events Register 0</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 502C</td><td><b>Offset End:</b> 502F</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>15</td><td>RW/P</td><td>0</td><td>INCOCE</td><td colspan="2">Increment Occurrence Count Enable: 0: Increment Duration Count: the counter is incremented for each clock for which the increment event signal is asserted logic high. 1: Increment Occurrence Count: the counter is incremented each time a rising edge of the increment event signal is detected.</td></tr>
<tr><td>14</td><td>RW</td><td>0</td><td>Reserved</td><td colspan="2">SOCHAPEV_0 Reserved</td></tr>
<tr><td>13</td><td>RW/P</td><td>0</td><td>INCEVOVR</td><td colspan="2"><b>Increment Event Override: When this bit is set, the increment event is tied to logical 1 .</b> This in turn causes the counter to count the number clocks that occurred in the native domain of the signal.</td></tr>
<tr><td>12</td><td>RW/P</td><td>0</td><td>INCBYP</td><td colspan="2"><b>Increment Bypass Valid Protocol: In this mode, the VIS valid protocol is not used.</b> The counter increments each clock regardless of valid data.</td></tr>
<tr><td>11:9</td><td>RW/P</td><td>0</td><td>INCVAL</td><td colspan="2"><b>Increment Counting Value: By setting this, the increment value used by the counter can be changed.</b>
Note, increment value = INCVAL+1.
000: Increment by 1
001: Increment by 2
010: Increment by 3
...
111: Increment by 8</td></tr>
<tr><td>8:0</td><td>RW/P</td><td>0</td><td>INCE</td><td colspan="2">Increment Event: This field contains the Event Selection Code (ESC) that the unit will be required to detect before incrementing the associated counter. This field is only applicable for opcodes that put the counter in a state where it is counting. See the Event tables for valid values. This limits ESC codes to the range of 000-1FFh</td></tr>
</table>

13.10.14 SOCHAPSTAT_0: SoCHAP Status Register 0

<table>
<tr><td colspan="4"><b>CSR Register Name: SOCHAPSTAT_0</b>: SoCHAP Status Register 0</td></tr>
<tr><td><b>Bar:</b> CSR_MTB_BAR</td><td><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5030</td><td><b>Offset End:</b> 5033</td></tr>
</table>

Bits	Access	Default	Label	Bit Description
31	RW/1C/P	0	DCI	**Decrement Counter Indicator: The decrement event was detected in the absence of the increment event.** The counter may or may not have decremented depending on the counter state when the decrement event occurred.
30	RW/1C/P	0	ICI	**Increment Counter Indicator: The increment event was detected in the absence of the decrement event.** The counter may or may not have incremented depending on the counter state when the increment event occurred.
29	RO	0	CAI	Counter Active Indicator: 0: The associated counter is in a state that does NOT allow it to be incremented or decremented if the appropriate event(s) are detected. 1: The associated counter is in a state that allows it to be incremented or decremented if the appropriate event(s) are detected. This bit is updated every clock.
28	RW/1C/P	0	IU	**In Use: Software Mutex bit.** After a RESET, a read to this bit returns a 0. After the first read, subsequent reads will return a 1. A write of a 1 to this bit will reset the next read value to 0. Writing a 0 to this bit has no effect Software can poll this bit until it reads a 0, and will then own the usage of the corresponding counter. This bit has no other effect on any SoCHAP Counter registers, and is only used as a semaphore among various independent software threads that may need to utilize this performance counter. Software that reads this register, but does not intend to claim exclusive access, must write a 1 to this bit (if it reads a 0), in order to allow other software threads to claim it. This bit is particularly useful when supporting multi-threaded environments.
27	RO	0	Reserved	SOCHAPSTAT_0 Reserved

CSR Register Name: SOCHAPSTAT_0: SoCHAP Status Register 0				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 5030	**Offset End:** 5033
Bits	**Access**	**Default**	**Label**	**Bit Description**
26	RW/1C/P	0	OUI	**Overflow/Underflow Indicator: 0: The associated 32 bit counter has NOT rolled over since the last time it was cleared.** 1: The associated 32 bit counter HAS rolled over since the last time it was cleared. This bit is set by hardware, and cleared by software writing a 1 to it.
25	RW/1C/P	0	CTI	**Command Trigger Indicator: 0: NO commands have been triggered since the last time this bit was cleared.** 1: A command WAS triggered since the last time this bit was cleared. Software can use this bit to know that a command that was pending earlier has been triggered. Once a command has been triggered, another command can be triggered to execute. This bit is set by hardware, and cleared by software writing a 1 to it.
24	RW/1C/P	0	THI	**Threshold Indicator: 0: No threshold event has been generated since the last time this bit was cleared.** 1: This counter generated a threshold event due to a true threshold condition compare since the last time this bit was cleared. This bit is set by hardware, and cleared by software writing a 1 to it.
23:0	RO	0	Reserved	SOCHAPSTAT_0 Reserved

13.10.15 SOCHAPDATA_0: SoCHAP Data Register 0

CSR Register Name: SOCHAPDATA_0: SoCHAP Data Register 0			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 5034	**Offset End:** 5037

Bits	Access	Default	Label	Bit Description
31:0	RW/P	0	CNTVAL	Counter Value: Contains either duration (number of clock ticks high) or occurrences (number of rising edges) contained in SoCHAP event counter 0 at time of sampling. The register is programmed to contain the threshold value that will be compared to the value in the event counter when non-0 condition codes have been selected.

13.10.16 SOCHAPCMD_1: SoCHAP Command Register 1

CSR Register Name: **SOCHAPCMD_1**: SoCHAP Command Register 1				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 5038	**Offset End:** 503B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:30	RO	0	Reserved	SOCHAPCMD_0 Reserved
29:28	RW	0	Reserved	SOCHAPCMD_0 Reserved
27	RW/P	0	PE	**PMI Enable: 0: PMIout is not asserted when the indicators are valid.** 1: The PMIout signal is asserted when the conditions enabled via the indicator enable bits are true.
26	RW/P	0	OUIE	Overflow/Underflow Indicator Enable: 0: No indication provided when a counter overflow or underflow occurs except for setting the Overflow/Underflow Indicator (OUI) status bit. 1: When the overflow/underflow condition is true, the counter will set the status bit and possibly assert the PMIout pin, which depending on SoC implementation, can be fed to another logic block or interrupt source.
25	RW/P	0	CTIE	Command Trigger Indicator Enable: 0: No indication provided when a command is triggered except for setting the Command Trigger Indicator (CTI) status bit. 1: When the command trigger condition is true, the counter will set the status bit and possibly assert the PMIout pin, which depending on SoC implementation, can be fed to another logic block or interrupt source.

<table>
<tr><td colspan="5"><b>CSR Register Name: SOCHAPCMD_1</b>: SoCHAP Command Register 1</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5038</td><td><b>Offset End:</b> 503B</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>24</td><td>RW/P</td><td>0</td><td>THIE</td><td colspan="2">Threshold Indicator Enable: 0: No indication provided when threshold condition is true except for setting the Threshold Indicator (TI) status bit. 1: When the threshold condition is true, the counter will set the status bit and possibly assert the PMIout pin, which depending on SoC implementation, can be fed to another logic block or interrupt source.</td></tr>
<tr><td>23:21</td><td>RW/P</td><td>0</td><td>CC</td><td colspan="2">Condition Code: This field contains the code that indicates what type of threshold compare is done between the counter and the data register. For all non-0 values of this field, the counter s data register will contain the threshold value. The outcome of this compare will generate a threshold event and potentially an interrupt if that capability is enabled.Bit 23 is for less than (<) Bit 22 is for equal (=) Bit 21 is for greater than (>) Select the proper bit mask for desired threshold condition: 000: False (no threshold compare) 001: Greater Than 010: Equal 011: Greater Than or Equal 100: Less Than 101: Not Equal 110: Less Than or Equal 111: True (always generate threshold event)</td></tr>
<tr><td>20</td><td>RW/P</td><td>0</td><td>SAC</td><td colspan="2">Select ALL Counters: When this bit is set, the opcode in bits 19:16 is applied to all counters. When cleared, the opcode in bits 19:16 only apply to the current counter. The rest of the SoCHAP command register is not affected by the setting of this bit. 0: The opcode is applied only to the counter associated with this command register. 1: The opcode is applied to ALL counters. This means that every command register is written to with the same value that was written to this particular command register. This is particularly handy for resetting all counters with a single command or starting or stopping all counters simultaneously. This bit is only valid in Command Register 0 and should be reserved in all non-0 command registers. Globally executed commands (by setting this bit in Command Register 0) always override locally executed commands.</td></tr>
</table>

CSR Register Name: SOCHAPCMD_1: SoCHAP Command Register 1				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 5038	**Offset End:** 503B
Bits	**Access**	**Default**	**Label**	**Bit Description**
19:16	RW/P	0	OPCODE	Opcode: 0000: Stop. The corresponding counter does not count. 0001: Start. The corresponding counter begins counting. A counter increments by ICV+1 if the corresponding increment event occurs or decrements by DCV+1 if the corresponding decrement event occurs. All duration type events toggle every SoCHAP unit clock tick for which the event is true. The desired increment and decrement events must be selected before this command executes. 0010: Sample. The corresponding counter value is latched into the corresponding data register, which can then be accessed by reading the appropriate data register. Counter functionality is unaffected by a Sample Opcode. The counter continues to count without being reset. If the Condition Code is NOT False then the Data Register is not written when a sample takes place. In other words the Data Register will NOT be updated with counter value if the Condition Code is > 0. 0100: Reset. The corresponding counter is reset to 0000 0000h. The 32 bit wide counter allows 4 billion clock ticks or occurrences to be counted between sample commands. When the counter rolls over, the overflow status bit will be set in the corresponding status register. 0101: Restart. The corresponding counter resets, and begins counting again. This is essentially a Reset & Start command. This functionality will facilitate histogram creation by allowing an event to trigger to clear the counter and resume counting with no further intervention. 0110: Sample & Restart. The Sample command happens and is followed immediately by the Restart command. 1111: Preload. The corresponding counter is set to the value that is located in the associated Data Register. This facilitates rollover and overflow validation. The counter remains in the same state when preloaded. If the counter was counting before the preload was executed it will continue to count after the preload. It is software s responsibility to ensure that the counter is in the desired state (example: execute stop command) prior to issuing a preload command. All others reserved.

<table>
<tr><td colspan="6"><b>CSR Register Name: SOCHAPCMD_1</b>: SoCHAP Command Register 1</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5038</td><td><b>Offset End:</b> 503B</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>15</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">SOCHAPCMD_0 Reserved</td></tr>
<tr><td>14:13</td><td>RW</td><td>0</td><td>Reserved</td><td colspan="2">SOCHAPCMD_0 Reserved</td></tr>
<tr><td>12</td><td>RW/P</td><td>0</td><td>TBV</td><td colspan="2"><b>Trigger Bypass Valid Protocol: In this mode, the VIS valid protocol is not used.</b> The counter counts every clock regardless of valid data. Since signals from VIS may span multiple clocks this mode is not recommended for triggers since multiple back to back triggers will likely occur.</td></tr>
<tr><td>11:10</td><td>RW</td><td>0</td><td>Reserved</td><td colspan="2">SOCHAPCMD_0 Reserved</td></tr>
<tr><td>9</td><td>RW/P</td><td>0</td><td>TTM</td><td colspan="2">Threshold Trigger Mode: When this bit is set, if the operation of the trigger includes a sample (Sample, Sample and Restart opcodes), then the data value is only written if the threshold comparison is true. This mode can be used to store maximum and minimum values into the data register.</td></tr>
<tr><td>8:0</td><td>RW/P</td><td>0</td><td>CT</td><td colspan="2">Command Trigger: This field contains the Event Selection Code (ESC) that the unit will be required to detect before executing the opcode. The previously programmed opcode continues to execute until this command trigger is detected. The ESC of 000h (the default value) is a special case. This special ESC causes the command to be triggered immediately upon a write to the command register. The special command trigger also causes the target command to be executed (triggered) once and only once. All other (non-0) ESCs cause the command to be re-executed every time the trigger is detected. Only one trigger can occur per VIS clock. If the trigger source clock is running faster than the VIS clock and more than one trigger event occurs per VIS clock, these additional triggers will be dropped. See the Event Tables for valid values. ESC values are limited to the range of 000-1FFh</td></tr>
</table>

13.10.17 SOCHAPEV_1: SoCHAP Events Register 1

CSR Register Name: SOCHAPEV_1: SoCHAP Events Register 1			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 503C	**Offset End:** 503F

Bits	Access	Default	Label	Bit Description
31	RW/P	0	DECOCE	Decrement Occurrence Count Enable: 0: Decrement Duration Count: the counter is decremented for each clock for which the decrement event signal is asserted logic high. 1: Decrement Occurrence Count: the counter is decremented each time a rising edge of the decrement event signal is detected.
30	RW	0	Reserved	SOCHAPEV_0 Reserved
29	RW/P	0	DECEVOVR	**Decrement Event Override: When this bit is set, the decrement event is tied to logical 1 .** This in turn causes the counter to count the number clocks that occurred in the native domain of the signal.
28	RW/P	0	DECBYP	**Decrement Bypass Valid Protocol: In this mode, the VIS valid protocol is not used.** The counter decrements each clock regardless of valid data.
27:25	RW/P	0	DECVAL	**Decrement Counting Value:By setting this, the decrement value used by the counter can be changed.** Note, decrement value = DECVAL+1. 000: Decrement by 1 001: Decrement by 2 010: Decrement by 3 ... 111: Decrement by 8
24:16	RW/P	0	DECE	Decrement Event: This field contains the Event Selection Code (ESC) that the unit will be required to detect before decrementing the associated counter. This field is only applicable for opcodes that put the counter in a state where it is counting. See the Event tables for valid values. This limits ESC codes to the range of 000-1FFh.
15	RW/P	0	INCOCE	Increment Occurrence Count Enable: 0: Increment Duration Count: the counter is incremented for each clock for which the increment event signal is asserted logic high. 1: Increment Occurrence Count: the counter is incremented each time a rising edge of the increment event signal is detected.

CSR Register Name: SOCHAPEV_1: SoCHAP Events Register 1

Bar: CSR_MTB_BAR | **Reset:** npk_rst_b | **Offset Start:** 503C | **Offset End:** 503F

Bits	Access	Default	Label	Bit Description
14	RW	0	Reserved	SOCHAPEV_0 Reserved
13	RW/P	0	INCEVOVR	**Increment Event Override: When this bit is set, the increment event is tied to logical 1 .** This in turn causes the counter to count the number clocks that occurred in the native domain of the signal.
12	RW/P	0	INCBYP	**Increment Bypass Valid Protocol: In this mode, the VIS valid protocol is not used.** The counter increments each clock regardless of valid data.
11:9	RW/P	0	INCVAL	**Increment Counting Value: By setting this, the increment value used by the counter can be changed.** Note, increment value = INCVAL+1. 000: Increment by 1 001: Increment by 2 010: Increment by 3 ... 111: Increment by 8
8:0	RW/P	0	INCE	Increment Event: This field contains the Event Selection Code (ESC) that the unit will be required to detect before incrementing the associated counter. This field is only applicable for opcodes that put the counter in a state where it is counting. See the Event tables for valid values. This limits ESC codes to the range of 000-1FFh

13.10.18 SOCHAPSTAT_1: SoCHAP Status Register 1

CSR Register Name: SOCHAPSTAT_1: SoCHAP Status Register 1

Bar: CSR_MTB_BAR | **Reset:** npk_rst_b | **Offset Start:** 5040 | **Offset End:** 5043

Bits	Access	Default	Label	Bit Description
31	RW/1C/P	0	DCI	**Decrement Counter Indicator: The decrement event was detected in the absence of the increment event.** The counter may or may not have decremented depending on the counter state when the decrement event occurred.

<table>
<tr><td colspan="6"><b>CSR Register Name: SOCHAPSTAT_1</b>: SoCHAP Status Register 1</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5040</td><td><b>Offset End:</b> 5043</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>30</td><td>RW/1C/P</td><td>0</td><td>ICI</td><td colspan="2"><b>Increment Counter Indicator: The increment event was detected in the absence of the decrement event.</b> The counter may or may not have incremented depending on the counter state when the increment event occurred.</td></tr>
<tr><td>29</td><td>RO</td><td>0</td><td>CAI</td><td colspan="2">Counter Active Indicator: 0: The associated counter is in a state that does NOT allow it to be incremented or decremented if the appropriate event(s) are detected. 1: The associated counter is in a state that allows it to be incremented or decremented if the appropriate event(s) are detected. This bit is updated every clock.</td></tr>
<tr><td>28</td><td>RW/1C/P</td><td>0</td><td>IU</td><td colspan="2"><b>In Use: Software Mutex bit.</b> After a RESET, a read to this bit returns a 0. After the first read, subsequent reads will return a 1.
A write of a 1 to this bit will reset the next read value to 0.
Writing a 0 to this bit has no effect
Software can poll this bit until it reads a 0, and will then own the usage of the corresponding counter. This bit has no other effect on any SoCHAP Counter registers, and is only used as a semaphore among various independent software threads that may need to utilize this performance counter.
Software that reads this register, but does not intend to claim exclusive access, must write a 1 to this bit (if it reads a 0), in order to allow other software threads to claim it. This bit is particularly useful when supporting multi-threaded environments.</td></tr>
<tr><td>27</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">SOCHAPSTAT_0 Reserved</td></tr>
<tr><td>26</td><td>RW/1C/P</td><td>0</td><td>OUI</td><td colspan="2"><b>Overflow/Underflow Indicator: 0: The associated 32 bit counter has NOT rolled over since the last time it was cleared.</b> 1: The associated 32 bit counter HAS rolled over since the last time it was cleared. This bit is set by hardware, and cleared by software writing a 1 to it.</td></tr>
</table>

<table>
<tr><td colspan="6"><b>CSR Register Name: SOCHAPSTAT_1</b>: SoCHAP Status Register 1</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5040</td><td><b>Offset End:</b> 5043</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>25</td><td>RW/1C/P</td><td>0</td><td>CTI</td><td colspan="2"><b>Command Trigger Indicator: 0: NO commands have been triggered since the last time this bit was cleared.</b> 1: A command WAS triggered since the last time this bit was cleared. Software can use this bit to know that a command that was pending earlier has been triggered. Once a command has been triggered, another command can be triggered to execute. This bit is set by hardware, and cleared by software writing a 1 to it.</td></tr>
<tr><td>24</td><td>RW/1C/P</td><td>0</td><td>THI</td><td colspan="2"><b>Threshold Indicator: 0: No threshold event has been generated since the last time this bit was cleared.</b> 1: This counter generated a threshold event due to a true threshold condition compare since the last time this bit was cleared. This bit is set by hardware, and cleared by software writing a 1 to it.</td></tr>
<tr><td>23:0</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">SOCHAPSTAT_0 Reserved</td></tr>
</table>

13.10.19 SOCHAPDATA_1: SoCHAP Data Register 1

<table>
<tr><td colspan="6"><b>CSR Register Name: SOCHAPDATA_1</b>: SoCHAP Data Register 1</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5044</td><td><b>Offset End:</b> 5047</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:0</td><td>RW/P</td><td>0</td><td>CNTVAL</td><td colspan="2">Counter Value: Contains either duration (number of clock ticks high) or occurrences (number of rising edges) contained in SoCHAP event counter 0 at time of sampling. The register is programmed to contain the threshold value that will be compared to the value in the event counter when non-0 condition codes have been selected.</td></tr>
</table>

13.10.20 SOCHAPCMD_2: SoCHAP Command Register 2

CSR Register Name: **SOCHAPCMD_2**: SoCHAP Command Register 2				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 5048	**Offset End:** 504B

Bits	Access	Default	Label	Bit Description
31:30	RO	0	Reserved	SOCHAPCMD_0 Reserved
29:28	RW	0	Reserved	SOCHAPCMD_0 Reserved
27	RW/P	0	PE	**PMI Enable: 0: PMIout is not asserted when the indicators are valid.** 1: The PMIout signal is asserted when the conditions enabled via the indicator enable bits are true.
26	RW/P	0	OUIE	Overflow/Underflow Indicator Enable: 0: No indication provided when a counter overflow or underflow occurs except for setting the Overflow/Underflow Indicator (OUI) status bit. 1: When the overflow/underflow condition is true, the counter will set the status bit and possibly assert the PMIout pin, which depending on SoC implementation, can be fed to another logic block or interrupt source.
25	RW/P	0	CTIE	Command Trigger Indicator Enable: 0: No indication provided when a command is triggered except for setting the Command Trigger Indicator (CTI) status bit. 1: When the command trigger condition is true, the counter will set the status bit and possibly assert the PMIout pin, which depending on SoC implementation, can be fed to another logic block or interrupt source.
24	RW/P	0	THIE	Threshold Indicator Enable: 0: No indication provided when threshold condition is true except for setting the Threshold Indicator (TI) status bit. 1: When the threshold condition is true, the counter will set the status bit and possibly assert the PMIout pin, which depending on SoC implementation, can be fed to another logic block or interrupt source.

CSR Register Name: SOCHAPCMD_2: SoCHAP Command Register 2				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 5048	**Offset End:** 504B
Bits	**Access**	**Default**	**Label**	**Bit Description**
23:21	RW/P	0	CC	Condition Code: This field contains the code that indicates what type of threshold compare is done between the counter and the data register. For all non-0 values of this field, the counter s data register will contain the threshold value. The outcome of this compare will generate a threshold event and potentially an interrupt if that capability is enabled.Bit 23 is for less than (<) Bit 22 is for equal (=) Bit 21 is for greater than (>) Select the proper bit mask for desired threshold condition: 000: False (no threshold compare) 001: Greater Than 010: Equal 011: Greater Than or Equal 100: Less Than 101: Not Equal 110: Less Than or Equal 111: True (always generate threshold event)
20	RW/P	0	SAC	Select ALL Counters: When this bit is set, the opcode in bits 19:16 is applied to all counters. When cleared, the opcode in bits 19:16 only apply to the current counter. The rest of the SoCHAP command register is not affected by the setting of this bit. 0: The opcode is applied only to the counter associated with this command register. 1: The opcode is applied to ALL counters. This means that every command register is written to with the same value that was written to this particular command register. This is particularly handy for resetting all counters with a single command or starting or stopping all counters simultaneously. This bit is only valid in Command Register 0 and should be reserved in all non-0 command registers. Globally executed commands (by setting this bit in Command Register 0) always override locally executed commands.

CSR Register Name: SOCHAPCMD_2: SoCHAP Command Register 2				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 5048	**Offset End:** 504B
Bits	**Access**	**Default**	**Label**	**Bit Description**
19:16	RW/P	0	OPCODE	Opcode: 0000: Stop. The corresponding counter does not count. 0001: Start. The corresponding counter begins counting. A counter increments by ICV+1 if the corresponding increment event occurs or decrements by DCV+1 if the corresponding decrement event occurs. All duration type events toggle every SoCHAP unit clock tick for which the event is true. The desired increment and decrement events must be selected before this command executes. 0010: Sample. The corresponding counter value is latched into the corresponding data register, which can then be accessed by reading the appropriate data register. Counter functionality is unaffected by a Sample Opcode. The counter continues to count without being reset. If the Condition Code is NOT False then the Data Register is not written when a sample takes place. In other words the Data Register will NOT be updated with counter value if the Condition Code is > 0. 0100: Reset. The corresponding counter is reset to 0000 0000h. The 32 bit wide counter allows 4 billion clock ticks or occurrences to be counted between sample commands. When the counter rolls over, the overflow status bit will be set in the corresponding status register. 0101: Restart. The corresponding counter resets, and begins counting again. This is essentially a Reset & Start command. This functionality will facilitate histogram creation by allowing an event to trigger to clear the counter and resume counting with no further intervention. 0110: Sample & Restart. The Sample command happens and is followed immediately by the Restart command. 1111: Preload. The corresponding counter is set to the value that is located in the associated Data Register. This facilitates rollover and overflow validation. The counter remains in the same state when preloaded. If the counter was counting before the preload was executed it will continue to count after the preload. It is software s responsibility to ensure that the counter is in the desired state (example: execute stop command) prior to issuing a preload command. All others reserved.

<table>
<tr><td colspan="6"><b>CSR Register Name: SOCHAPCMD_2</b>: SoCHAP Command Register 2</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5048</td><td><b>Offset End:</b> 504B</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>15</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">SOCHAPCMD_0 Reserved</td></tr>
<tr><td>14:13</td><td>RW</td><td>0</td><td>Reserved</td><td colspan="2">SOCHAPCMD_0 Reserved</td></tr>
<tr><td>12</td><td>RW/P</td><td>0</td><td>TBV</td><td colspan="2"><b>Trigger Bypass Valid Protocol: In this mode, the VIS valid protocol is not used.</b> The counter counts every clock regardless of valid data. Since signals from VIS may span multiple clocks this mode is not recommended for triggers since multiple back to back triggers will likely occur.</td></tr>
<tr><td>11:10</td><td>RW</td><td>0</td><td>Reserved</td><td colspan="2">SOCHAPCMD_0 Reserved</td></tr>
<tr><td>9</td><td>RW/P</td><td>0</td><td>TTM</td><td colspan="2">Threshold Trigger Mode: When this bit is set, if the operation of the trigger includes a sample (Sample, Sample and Restart opcodes), then the data value is only written if the threshold comparison is true. This mode can be used to store maximum and minimum values into the data register.</td></tr>
<tr><td>8:0</td><td>RW/P</td><td>0</td><td>CT</td><td colspan="2">Command Trigger: This field contains the Event Selection Code (ESC) that the unit will be required to detect before executing the opcode. The previously programmed opcode continues to execute until this command trigger is detected. The ESC of 000h (the default value) is a special case. This special ESC causes the command to be triggered immediately upon a write to the command register. The special command trigger also causes the target command to be executed (triggered) once and only once. All other (non-0) ESCs cause the command to be re-executed every time the trigger is detected. Only one trigger can occur per VIS clock. If the trigger source clock is running faster than the VIS clock and more than one trigger event occurs per VIS clock, these additional triggers will be dropped. See the Event Tables for valid values. ESC values are limited to the range of 000-1FFh</td></tr>
</table>

13.10.21 SOCHAPEV_2: SoCHAP Events Register 2

CSR Register Name: **SOCHAPEV_2**: SoCHAP Events Register 2				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 504C **Offset End:** 504F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW/P	0	DECOCE	Decrement Occurrence Count Enable: 0: Decrement Duration Count: the counter is decremented for each clock for which the decrement event signal is asserted logic high. 1: Decrement Occurrence Count: the counter is decremented each time a rising edge of the decrement event signal is detected.
30	RW	0	Reserved	SOCHAPEV_0 Reserved
29	RW/P	0	DECEVOVR	**Decrement Event Override: When this bit is set, the decrement event is tied to logical 1 .** This in turn causes the counter to count the number clocks that occurred in the native domain of the signal.
28	RW/P	0	DECBYP	**Decrement Bypass Valid Protocol: In this mode, the VIS valid protocol is not used.** The counter decrements each clock regardless of valid data.
27:25	RW/P	0	DECVAL	**Decrement Counting Value:By setting this, the decrement value used by the counter can be changed.** Note, decrement value = DECVAL+1. 000: Decrement by 1 001: Decrement by 2 010: Decrement by 3 ... 111: Decrement by 8
24:16	RW/P	0	DECE	Decrement Event: This field contains the Event Selection Code (ESC) that the unit will be required to detect before decrementing the associated counter. This field is only applicable for opcodes that put the counter in a state where it is counting. See the Event tables for valid values. This limits ESC codes to the range of 000-1FFh.
15	RW/P	0	INCOCE	Increment Occurrence Count Enable: 0: Increment Duration Count: the counter is incremented for each clock for which the increment event signal is asserted logic high. 1: Increment Occurrence Count: the counter is incremented each time a rising edge of the increment event signal is detected.

CSR Register Name: SOCHAPEV_2: SoCHAP Events Register 2				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 504C	**Offset End:** 504F	
Bits	**Access**	**Default**	**Label**	**Bit Description**
14	RW	0	Reserved	SOCHAPEV_0 Reserved
13	RW/P	0	INCEVOVR	**Increment Event Override: When this bit is set, the increment event is tied to logical 1 .** This in turn causes the counter to count the number clocks that occurred in the native domain of the signal.
12	RW/P	0	INCBYP	**Increment Bypass Valid Protocol: In this mode, the VIS valid protocol is not used.** The counter increments each clock regardless of valid data.
11:9	RW/P	0	INCVAL	**Increment Counting Value: By setting this, the increment value used by the counter can be changed.** Note, increment value = INCVAL+1. 000: Increment by 1 001: Increment by 2 010: Increment by 3 ... 111: Increment by 8
8:0	RW/P	0	INCE	Increment Event: This field contains the Event Selection Code (ESC) that the unit will be required to detect before incrementing the associated counter. This field is only applicable for opcodes that put the counter in a state where it is counting. See the Event tables for valid values. This limits ESC codes to the range of 000-1FFh

13.10.22 SOCHAPSTAT_2: SoCHAP Status Register 2

CSR Register Name: SOCHAPSTAT_2: SoCHAP Status Register 2				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 5050	**Offset End:** 5053	
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW/1C/P	0	DCI	**Decrement Counter Indicator: The decrement event was detected in the absence of the increment event.** The counter may or may not have decremented depending on the counter state when the decrement event occurred.

CSR Register Name: SOCHAPSTAT_2: SoCHAP Status Register 2				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 5050 **Offset End:** 5053
Bits	**Access**	**Default**	**Label**	**Bit Description**
30	RW/1C/P	0	ICI	**Increment Counter Indicator: The increment event was detected in the absence of the decrement event.** The counter may or may not have incremented depending on the counter state when the increment event occurred.
29	RO	0	CAI	Counter Active Indicator: 0: The associated counter is in a state that does NOT allow it to be incremented or decremented if the appropriate event(s) are detected. 1: The associated counter is in a state that allows it to be incremented or decremented if the appropriate event(s) are detected. This bit is updated every clock.
28	RW/1C/P	0	IU	**In Use: Software Mutex bit.** After a RESET, a read to this bit returns a 0. After the first read, subsequent reads will return a 1. A write of a 1 to this bit will reset the next read value to 0. Writing a 0 to this bit has no effect Software can poll this bit until it reads a 0, and will then own the usage of the corresponding counter. This bit has no other effect on any SoCHAP Counter registers, and is only used as a semaphore among various independent software threads that may need to utilize this performance counter. Software that reads this register, but does not intend to claim exclusive access, must write a 1 to this bit (if it reads a 0), in order to allow other software threads to claim it. This bit is particularly useful when supporting multi-threaded environments.
27	RO	0	Reserved	SOCHAPSTAT_0 Reserved
26	RW/1C/P	0	OUI	**Overflow/Underflow Indicator: 0: The associated 32 bit counter has NOT rolled over since the last time it was cleared.** 1: The associated 32 bit counter HAS rolled over since the last time it was cleared. This bit is set by hardware, and cleared by software writing a 1 to it.

<table>
<tr><td colspan="5"><b>CSR Register Name: SOCHAPSTAT_2</b>: SoCHAP Status Register 2</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5050 <b>Offset End:</b> 5053</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th>Bit Description</th></tr>
<tr><td>25</td><td>RW/1C/P</td><td>0</td><td>CTI</td><td><b>Command Trigger Indicator: 0: NO commands have been triggered since the last time this bit was cleared.</b> 1: A command WAS triggered since the last time this bit was cleared. Software can use this bit to know that a command that was pending earlier has been triggered. Once a command has been triggered, another command can be triggered to execute. This bit is set by hardware, and cleared by software writing a 1 to it.</td></tr>
<tr><td>24</td><td>RW/1C/P</td><td>0</td><td>THI</td><td><b>Threshold Indicator: 0: No threshold event has been generated since the last time this bit was cleared.</b> 1: This counter generated a threshold event due to a true threshold condition compare since the last time this bit was cleared. This bit is set by hardware, and cleared by software writing a 1 to it.</td></tr>
<tr><td>23:0</td><td>RO</td><td>0</td><td>Reserved</td><td>SOCHAPSTAT_0 Reserved</td></tr>
</table>

13.10.23 SOCHAPDATA_2: SoCHAP Data Register 2

<table>
<tr><td colspan="5"><b>CSR Register Name: SOCHAPDATA_2</b>: SoCHAP Data Register 2</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5054 <b>Offset End:</b> 5057</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th>Bit Description</th></tr>
<tr><td>31:0</td><td>RW/P</td><td>0</td><td>CNTVAL</td><td>Counter Value: Contains either duration (number of clock ticks high) or occurrences (number of rising edges) contained in SoCHAP event counter 0 at time of sampling. The register is programmed to contain the threshold value that will be compared to the value in the event counter when non-0 condition codes have been selected.</td></tr>
</table>

13.10.24 SOCHAPCMD_3: SoCHAP Command Register 3

CSR Register Name: SOCHAPCMD_3: SoCHAP Command Register 3			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 5058	**Offset End:** 505B

Bits	Access	Default	Label	Bit Description
31:30	RO	0	Reserved	SOCHAPCMD_0 Reserved
29:28	RW	0	Reserved	SOCHAPCMD_0 Reserved
27	RW/P	0	PE	**PMI Enable: 0: PMIout is not asserted when the indicators are valid.** 1: The PMIout signal is asserted when the conditions enabled via the indicator enable bits are true.
26	RW/P	0	OUIE	Overflow/Underflow Indicator Enable: 0: No indication provided when a counter overflow or underflow occurs except for setting the Overflow/Underflow Indicator (OUI) status bit. 1: When the overflow/underflow condition is true, the counter will set the status bit and possibly assert the PMIout pin, which depending on SoC implementation, can be fed to another logic block or interrupt source.
25	RW/P	0	CTIE	Command Trigger Indicator Enable: 0: No indication provided when a command is triggered except for setting the Command Trigger Indicator (CTI) status bit. 1: When the command trigger condition is true, the counter will set the status bit and possibly assert the PMIout pin, which depending on SoC implementation, can be fed to another logic block or interrupt source.
24	RW/P	0	THIE	Threshold Indicator Enable: 0: No indication provided when threshold condition is true except for setting the Threshold Indicator (TI) status bit. 1: When the threshold condition is true, the counter will set the status bit and possibly assert the PMIout pin, which depending on SoC implementation, can be fed to another logic block or interrupt source.

<table>
<tr><td colspan="6"><b>CSR Register Name: SOCHAPCMD_3</b>: SoCHAP Command Register 3</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5058</td><td><b>Offset End:</b> 505B</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>23:21</td><td>RW/P</td><td>0</td><td>CC</td><td colspan="2">Condition Code: This field contains the code that indicates what type of threshold compare is done between the counter and the data register. For all non-0 values of this field, the counter s data register will contain the threshold value. The outcome of this compare will generate a threshold event and potentially an interrupt if that capability is enabled.Bit 23 is for less than (<) Bit 22 is for equal (=) Bit 21 is for greater than (>) Select the proper bit mask for desired threshold condition: 000: False (no threshold compare) 001: Greater Than 010: Equal 011: Greater Than or Equal 100: Less Than 101: Not Equal 110: Less Than or Equal 111: True (always generate threshold event)</td></tr>
<tr><td>20</td><td>RW/P</td><td>0</td><td>SAC</td><td colspan="2">Select ALL Counters: When this bit is set, the opcode in bits 19:16 is applied to all counters. When cleared, the opcode in bits 19:16 only apply to the current counter. The rest of the SoCHAP command register is not affected by the setting of this bit. 0: The opcode is applied only to the counter associated with this command register. 1: The opcode is applied to ALL counters. This means that every command register is written to with the same value that was written to this particular command register. This is particularly handy for resetting all counters with a single command or starting or stopping all counters simultaneously. This bit is only valid in Command Register 0 and should be reserved in all non-0 command registers. Globally executed commands (by setting this bit in Command Register 0) always override locally executed commands.</td></tr>
</table>

CSR Register Name: SOCHAPCMD_3: SoCHAP Command Register 3				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 5058 **Offset End:** 505B
Bits	**Access**	**Default**	**Label**	**Bit Description**
19:16	RW/P	0	OPCODE	Opcode: 0000: Stop. The corresponding counter does not count. 0001: Start. The corresponding counter begins counting. A counter increments by ICV+1 if the corresponding increment event occurs or decrements by DCV+1 if the corresponding decrement event occurs. All duration type events toggle every SoCHAP unit clock tick for which the event is true. The desired increment and decrement events must be selected before this command executes. 0010: Sample. The corresponding counter value is latched into the corresponding data register, which can then be accessed by reading the appropriate data register. Counter functionality is unaffected by a Sample Opcode. The counter continues to count without being reset. If the Condition Code is NOT False then the Data Register is not written when a sample takes place. In other words the Data Register will NOT be updated with counter value if the Condition Code is > 0. 0100: Reset. The corresponding counter is reset to 0000 0000h. The 32 bit wide counter allows 4 billion clock ticks or occurrences to be counted between sample commands. When the counter rolls over, the overflow status bit will be set in the corresponding status register. 0101: Restart. The corresponding counter resets, and begins counting again. This is essentially a Reset & Start command. This functionality will facilitate histogram creation by allowing an event to trigger to clear the counter and resume counting with no further intervention. 0110: Sample & Restart. The Sample command happens and is followed immediately by the Restart command. 1111: Preload. The corresponding counter is set to the value that is located in the associated Data Register. This facilitates rollover and overflow validation. The counter remains in the same state when preloaded. If the counter was counting before the preload was executed it will continue to count after the preload. It is software s responsibility to ensure that the counter is in the desired state (example: execute stop command) prior to issuing a preload command. All others reserved.

CSR Register Name: SOCHAPCMD_3: SoCHAP Command Register 3				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 5058 **Offset End:** 505B
Bits	**Access**	**Default**	**Label**	**Bit Description**
15	RO	0	Reserved	SOCHAPCMD_0 Reserved
14:13	RW	0	Reserved	SOCHAPCMD_0 Reserved
12	RW/P	0	TBV	**Trigger Bypass Valid Protocol: In this mode, the VIS valid protocol is not used.** The counter counts every clock regardless of valid data. Since signals from VIS may span multiple clocks this mode is not recommended for triggers since multiple back to back triggers will likely occur.
11:10	RW	0	Reserved	SOCHAPCMD_0 Reserved
9	RW/P	0	TTM	Threshold Trigger Mode: When this bit is set, if the operation of the trigger includes a sample (Sample, Sample and Restart opcodes), then the data value is only written if the threshold comparison is true. This mode can be used to store maximum and minimum values into the data register.
8:0	RW/P	0	CT	Command Trigger: This field contains the Event Selection Code (ESC) that the unit will be required to detect before executing the opcode. The previously programmed opcode continues to execute until this command trigger is detected. The ESC of 000h (the default value) is a special case. This special ESC causes the command to be triggered immediately upon a write to the command register. The special command trigger also causes the target command to be executed (triggered) once and only once. All other (non-0) ESCs cause the command to be re-executed every time the trigger is detected. Only one trigger can occur per VIS clock. If the trigger source clock is running faster than the VIS clock and more than one trigger event occurs per VIS clock, these additional triggers will be dropped. See the Event Tables for valid values. ESC values are limited to the range of 000-1FFh

13.10.25 SOCHAPEV_3: SoCHAP Events Register 3

CSR Register Name: SOCHAPEV_3: SoCHAP Events Register 3				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 505C **Offset End:** 505F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW/P	0	DECOCE	Decrement Occurrence Count Enable: 0: Decrement Duration Count: the counter is decremented for each clock for which the decrement event signal is asserted logic high. 1: Decrement Occurrence Count: the counter is decremented each time a rising edge of the decrement event signal is detected.
30	RW	0	Reserved	SOCHAPEV_0 Reserved
29	RW/P	0	DECEVOVR	**Decrement Event Override: When this bit is set, the decrement event is tied to logical 1 .** This in turn causes the counter to count the number clocks that occurred in the native domain of the signal.
28	RW/P	0	DECBYP	**Decrement Bypass Valid Protocol: In this mode, the VIS valid protocol is not used.** The counter decrements each clock regardless of valid data.
27:25	RW/P	0	DECVAL	**Decrement Counting Value:By setting this, the decrement value used by the counter can be changed.** Note, decrement value = DECVAL+1. 000: Decrement by 1 001: Decrement by 2 010: Decrement by 3 ... 111: Decrement by 8
24:16	RW/P	0	DECE	Decrement Event: This field contains the Event Selection Code (ESC) that the unit will be required to detect before decrementing the associated counter. This field is only applicable for opcodes that put the counter in a state where it is counting. See the Event tables for valid values. This limits ESC codes to the range of 000-1FFh.
15	RW/P	0	INCOCE	Increment Occurrence Count Enable: 0: Increment Duration Count: the counter is incremented for each clock for which the increment event signal is asserted logic high. 1: Increment Occurrence Count: the counter is incremented each time a rising edge of the increment event signal is detected.

CSR Register Name: SOCHAPEV_3: SoCHAP Events Register 3				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 505C	**Offset End:** 505F
Bits	**Access**	**Default**	**Label**	**Bit Description**
14	RW	0	Reserved	SOCHAPEV_0 Reserved
13	RW/P	0	INCEVOVR	**Increment Event Override: When this bit is set, the increment event is tied to logical 1 .** This in turn causes the counter to count the number clocks that occurred in the native domain of the signal.
12	RW/P	0	INCBYP	**Increment Bypass Valid Protocol: In this mode, the VIS valid protocol is not used.** The counter increments each clock regardless of valid data.
11:9	RW/P	0	INCVAL	**Increment Counting Value: By setting this, the increment value used by the counter can be changed.** Note, increment value = INCVAL+1. 000: Increment by 1 001: Increment by 2 010: Increment by 3 ... 111: Increment by 8
8:0	RW/P	0	INCE	Increment Event: This field contains the Event Selection Code (ESC) that the unit will be required to detect before incrementing the associated counter. This field is only applicable for opcodes that put the counter in a state where it is counting. See the Event tables for valid values. This limits ESC codes to the range of 000-1FFh

13.10.26 SOCHAPSTAT_3: SoCHAP Status Register 3

CSR Register Name: SOCHAPSTAT_3: SoCHAP Status Register 3				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 5060	**Offset End:** 5063
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW/1C/P	0	DCI	**Decrement Counter Indicator: The decrement event was detected in the absence of the increment event.** The counter may or may not have decremented depending on the counter state when the decrement event occurred.

<table>
<tr><td colspan="6"><b>CSR Register Name:</b> <b>SOCHAPSTAT_3</b>: SoCHAP Status Register 3</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5060</td><td><b>Offset End:</b> 5063</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>30</td><td>RW/1C/P</td><td>0</td><td>ICI</td><td colspan="2"><b>Increment Counter Indicator: The increment event was detected in the absence of the decrement event.</b> The counter may or may not have incremented depending on the counter state when the increment event occurred.</td></tr>
<tr><td>29</td><td>RO</td><td>0</td><td>CAI</td><td colspan="2">Counter Active Indicator: 0: The associated counter is in a state that does NOT allow it to be incremented or decremented if the appropriate event(s) are detected. 1: The associated counter is in a state that allows it to be incremented or decremented if the appropriate event(s) are detected. This bit is updated every clock.</td></tr>
<tr><td>28</td><td>RW/1C/P</td><td>0</td><td>IU</td><td colspan="2"><b>In Use: Software Mutex bit.</b> After a RESET, a read to this bit returns a 0. After the first read, subsequent reads will return a 1.
A write of a 1 to this bit will reset the next read value to 0.
Writing a 0 to this bit has no effect
Software can poll this bit until it reads a 0, and will then own the usage of the corresponding counter. This bit has no other effect on any SoCHAP Counter registers, and is only used as a semaphore among various independent software threads that may need to utilize this performance counter.
Software that reads this register, but does not intend to claim exclusive access, must write a 1 to this bit (if it reads a 0), in order to allow other software threads to claim it. This bit is particularly useful when supporting multi-threaded environments.</td></tr>
<tr><td>27</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">SOCHAPSTAT_0 Reserved</td></tr>
<tr><td>26</td><td>RW/1C/P</td><td>0</td><td>OUI</td><td colspan="2"><b>Overflow/Underflow Indicator: 0: The associated 32 bit counter has NOT rolled over since the last time it was cleared.</b> 1: The associated 32 bit counter HAS rolled over since the last time it was cleared. This bit is set by hardware, and cleared by software writing a 1 to it.</td></tr>
</table>

<table>
<tr><td colspan="6"><b>CSR Register Name: SOCHAPSTAT_3</b>: SoCHAP Status Register 3</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5060</td><td><b>Offset End:</b> 5063</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>25</td><td>RW/1C/P</td><td>0</td><td>CTI</td><td colspan="2"><b>Command Trigger Indicator: 0: NO commands have been triggered since the last time this bit was cleared.</b> 1: A command WAS triggered since the last time this bit was cleared. Software can use this bit to know that a command that was pending earlier has been triggered. Once a command has been triggered, another command can be triggered to execute. This bit is set by hardware, and cleared by software writing a 1 to it.</td></tr>
<tr><td>24</td><td>RW/1C/P</td><td>0</td><td>THI</td><td colspan="2"><b>Threshold Indicator: 0: No threshold event has been generated since the last time this bit was cleared.</b> 1: This counter generated a threshold event due to a true threshold condition compare since the last time this bit was cleared. This bit is set by hardware, and cleared by software writing a 1 to it.</td></tr>
<tr><td>23:0</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">SOCHAPSTAT_0 Reserved</td></tr>
</table>

13.10.27 SOCHAPDATA_3: SoCHAP Data Register 3

<table>
<tr><td colspan="6"><b>CSR Register Name: SOCHAPDATA_3</b>: SoCHAP Data Register 3</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5064</td><td><b>Offset End:</b> 5067</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:0</td><td>RW/P</td><td>0</td><td>CNTVAL</td><td colspan="2">Counter Value: Contains either duration (number of clock ticks high) or occurrences (number of rising edges) contained in SoCHAP event counter 0 at time of sampling. The register is programmed to contain the threshold value that will be compared to the value in the event counter when non-0 condition codes have been selected.</td></tr>
</table>

13.10.28 SOCHAPCMD_4: SoCHAP Command Register 4

<table>
<tr><td colspan="6"><b>CSR Register Name: SOCHAPCMD_4</b>: SoCHAP Command Register 4</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5068</td><td><b>Offset End:</b> 506B</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:30</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">SOCHAPCMD_0 Reserved</td></tr>
<tr><td>29:28</td><td>RW</td><td>0</td><td>Reserved</td><td colspan="2">SOCHAPCMD_0 Reserved</td></tr>
<tr><td>27</td><td>RW/P</td><td>0</td><td>PE</td><td colspan="2"><b>PMI Enable: 0: PMIout is not asserted when the indicators are valid.</b> 1: The PMIout signal is asserted when the conditions enabled via the indicator enable bits are true.</td></tr>
<tr><td>26</td><td>RW/P</td><td>0</td><td>OUIE</td><td colspan="2">Overflow/Underflow Indicator Enable: 0: No indication provided when a counter overflow or underflow occurs except for setting the Overflow/Underflow Indicator (OUI) status bit. 1: When the overflow/underflow condition is true, the counter will set the status bit and possibly assert the PMIout pin, which depending on SoC implementation, can be fed to another logic block or interrupt source.</td></tr>
<tr><td>25</td><td>RW/P</td><td>0</td><td>CTIE</td><td colspan="2">Command Trigger Indicator Enable: 0: No indication provided when a command is triggered except for setting the Command Trigger Indicator (CTI) status bit. 1: When the command trigger condition is true, the counter will set the status bit and possibly assert the PMIout pin, which depending on SoC implementation, can be fed to another logic block or interrupt source.</td></tr>
<tr><td>24</td><td>RW/P</td><td>0</td><td>THIE</td><td colspan="2">Threshold Indicator Enable: 0: No indication provided when threshold condition is true except for setting the Threshold Indicator (TI) status bit. 1: When the threshold condition is true, the counter will set the status bit and possibly assert the PMIout pin, which depending on SoC implementation, can be fed to another logic block or interrupt source.</td></tr>
</table>

<table>
<tr><td colspan="6"><b>CSR Register Name: SOCHAPCMD_4</b>: SoCHAP Command Register 4</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5068</td><td><b>Offset End:</b> 506B</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>23:21</td><td>RW/P</td><td>0</td><td>CC</td><td colspan="2">Condition Code: This field contains the code that indicates what type of threshold compare is done between the counter and the data register. For all non-0 values of this field, the counter s data register will contain the threshold value. The outcome of this compare will generate a threshold event and potentially an interrupt if that capability is enabled.Bit 23 is for less than (<) Bit 22 is for equal (=) Bit 21 is for greater than (>) Select the proper bit mask for desired threshold condition: 000: False (no threshold compare) 001: Greater Than 010: Equal 011: Greater Than or Equal 100: Less Than 101: Not Equal 110: Less Than or Equal 111: True (always generate threshold event)</td></tr>
<tr><td>20</td><td>RW/P</td><td>0</td><td>SAC</td><td colspan="2">Select ALL Counters: When this bit is set, the opcode in bits 19:16 is applied to all counters. When cleared, the opcode in bits 19:16 only apply to the current counter. The rest of the SoCHAP command register is not affected by the setting of this bit. 0: The opcode is applied only to the counter associated with this command register. 1: The opcode is applied to ALL counters. This means that every command register is written to with the same value that was written to this particular command register. This is particularly handy for resetting all counters with a single command or starting or stopping all counters simultaneously. This bit is only valid in Command Register 0 and should be reserved in all non-0 command registers. Globally executed commands (by setting this bit in Command Register 0) always override locally executed commands.</td></tr>
</table>

<table>
<tr><td colspan="6"><b>CSR Register Name: SOCHAPCMD_4</b>: SoCHAP Command Register 4</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5068</td><td><b>Offset End:</b> 506B</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>19:16</td><td>RW/P</td><td>0</td><td>OPCODE</td><td colspan="2">Opcode:
0000: Stop. The corresponding counter does not count.
0001: Start. The corresponding counter begins counting. A counter increments by ICV+1 if the corresponding increment event occurs or decrements by DCV+1 if the corresponding decrement event occurs. All duration type events toggle every SoCHAP unit clock tick for which the event is true. The desired increment and decrement events must be selected before this command executes.
0010: Sample. The corresponding counter value is latched into the corresponding data register, which can then be accessed by reading the appropriate data register. Counter functionality is unaffected by a Sample Opcode. The counter continues to count without being reset. If the Condition Code is NOT False then the Data Register is not written when a sample takes place. In other words the Data Register will NOT be updated with counter value if the Condition Code is > 0.
0100: Reset. The corresponding counter is reset to 0000 0000h. The 32 bit wide counter allows 4 billion clock ticks or occurrences to be counted between sample commands. When the counter rolls over, the overflow status bit will be set in the corresponding status register.
0101: Restart. The corresponding counter resets, and begins counting again. This is essentially a Reset & Start command. This functionality will facilitate histogram creation by allowing an event to trigger to clear the counter and resume counting with no further intervention.
0110: Sample & Restart. The Sample command happens and is followed immediately by the Restart command.
1111: Preload. The corresponding counter is set to the value that is located in the associated Data Register. This facilitates rollover and overflow validation. The counter remains in the same state when preloaded. If the counter was counting before the preload was executed it will continue to count after the preload. It is software s responsibility to ensure that the counter is in the desired state (example: execute stop command) prior to issuing a preload command.
All others reserved.</td></tr>
</table>

CSR Register Name: SOCHAPCMD_4: SoCHAP Command Register 4				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 5068 **Offset End:** 506B
Bits	**Access**	**Default**	**Label**	**Bit Description**
15	RO	0	Reserved	SOCHAPCMD_0 Reserved
14:13	RW	0	Reserved	SOCHAPCMD_0 Reserved
12	RW/P	0	TBV	**Trigger Bypass Valid Protocol: In this mode, the VIS valid protocol is not used.** The counter counts every clock regardless of valid data. Since signals from VIS may span multiple clocks this mode is not recommended for triggers since multiple back to back triggers will likely occur.
11:10	RW	0	Reserved	SOCHAPCMD_0 Reserved
9	RW/P	0	TTM	Threshold Trigger Mode: When this bit is set, if the operation of the trigger includes a sample (Sample, Sample and Restart opcodes), then the data value is only written if the threshold comparison is true. This mode can be used to store maximum and minimum values into the data register.
8:0	RW/P	0	CT	Command Trigger: This field contains the Event Selection Code (ESC) that the unit will be required to detect before executing the opcode. The previously programmed opcode continues to execute until this command trigger is detected. The ESC of 000h (the default value) is a special case. This special ESC causes the command to be triggered immediately upon a write to the command register. The special command trigger also causes the target command to be executed (triggered) once and only once. All other (non-0) ESCs cause the command to be re-executed every time the trigger is detected. Only one trigger can occur per VIS clock. If the trigger source clock is running faster than the VIS clock and more than one trigger event occurs per VIS clock, these additional triggers will be dropped. See the Event Tables for valid values. ESC values are limited to the range of 000-1FFh

13.10.29 SOCHAPEV_4: SoCHAP Events Register 4

CSR Register Name: SOCHAPEV_4: SoCHAP Events Register 4				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 506C **Offset End:** 506F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW/P	0	DECOCE	Decrement Occurrence Count Enable: 0: Decrement Duration Count: the counter is decremented for each clock for which the decrement event signal is asserted logic high. 1: Decrement Occurrence Count: the counter is decremented each time a rising edge of the decrement event signal is detected.
30	RW	0	Reserved	SOCHAPEV_0 Reserved
29	RW/P	0	DECEVOVR	**Decrement Event Override: When this bit is set, the decrement event is tied to logical 1 .** This in turn causes the counter to count the number clocks that occurred in the native domain of the signal.
28	RW/P	0	DECBYP	**Decrement Bypass Valid Protocol: In this mode, the VIS valid protocol is not used.** The counter decrements each clock regardless of valid data.
27:25	RW/P	0	DECVAL	**Decrement Counting Value:By setting this, the decrement value used by the counter can be changed.** Note, decrement value = DECVAL+1. 000: Decrement by 1 001: Decrement by 2 010: Decrement by 3 ... 111: Decrement by 8
24:16	RW/P	0	DECE	Decrement Event: This field contains the Event Selection Code (ESC) that the unit will be required to detect before decrementing the associated counter. This field is only applicable for opcodes that put the counter in a state where it is counting. See the Event tables for valid values. This limits ESC codes to the range of 000-1FFh.
15	RW/P	0	INCOCE	Increment Occurrence Count Enable: 0: Increment Duration Count: the counter is incremented for each clock for which the increment event signal is asserted logic high. 1: Increment Occurrence Count: the counter is incremented each time a rising edge of the increment event signal is detected.

CSR Register Name: SOCHAPEV_4: SoCHAP Events Register 4				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 506C **Offset End:** 506F
Bits	**Access**	**Default**	**Label**	**Bit Description**
14	RW	0	Reserved	SOCHAPEV_0 Reserved
13	RW/P	0	INCEVOVR	**Increment Event Override: When this bit is set, the increment event is tied to logical 1 .** This in turn causes the counter to count the number clocks that occurred in the native domain of the signal.
12	RW/P	0	INCBYP	**Increment Bypass Valid Protocol: In this mode, the VIS valid protocol is not used.** The counter increments each clock regardless of valid data.
11:9	RW/P	0	INCVAL	**Increment Counting Value: By setting this, the increment value used by the counter can be changed.** Note, increment value = INCVAL+1. 000: Increment by 1 001: Increment by 2 010: Increment by 3 ... 111: Increment by 8
8:0	RW/P	0	INCE	Increment Event: This field contains the Event Selection Code (ESC) that the unit will be required to detect before incrementing the associated counter. This field is only applicable for opcodes that put the counter in a state where it is counting. See the Event tables for valid values. This limits ESC codes to the range of 000-1FFh

13.10.30 SOCHAPSTAT_4: SoCHAP Status Register 4

CSR Register Name: SOCHAPSTAT_4: SoCHAP Status Register 4				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 5070 **Offset End:** 5073
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW/1C/P	0	DCI	**Decrement Counter Indicator: The decrement event was detected in the absence of the increment event.** The counter may or may not have decremented depending on the counter state when the decrement event occurred.

<table>
<tr><td colspan="6"><b>CSR Register Name: SOCHAPSTAT_4</b>: SoCHAP Status Register 4</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5070</td><td><b>Offset End:</b> 5073</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>30</td><td>RW/1C/P</td><td>0</td><td>ICI</td><td colspan="2"><b>Increment Counter Indicator: The increment event was detected in the absence of the decrement event.</b> The counter may or may not have incremented depending on the counter state when the increment event occurred.</td></tr>
<tr><td>29</td><td>RO</td><td>0</td><td>CAI</td><td colspan="2">Counter Active Indicator: 0: The associated counter is in a state that does NOT allow it to be incremented or decremented if the appropriate event(s) are detected. 1: The associated counter is in a state that allows it to be incremented or decremented if the appropriate event(s) are detected. This bit is updated every clock.</td></tr>
<tr><td>28</td><td>RW/1C/P</td><td>0</td><td>IU</td><td colspan="2"><b>In Use: Software Mutex bit.</b> After a RESET, a read to this bit returns a 0. After the first read, subsequent reads will return a 1.
A write of a 1 to this bit will reset the next read value to 0.
Writing a 0 to this bit has no effect
Software can poll this bit until it reads a 0, and will then own the usage of the corresponding counter. This bit has no other effect on any SoCHAP Counter registers, and is only used as a semaphore among various independent software threads that may need to utilize this performance counter.
Software that reads this register, but does not intend to claim exclusive access, must write a 1 to this bit (if it reads a 0), in order to allow other software threads to claim it. This bit is particularly useful when supporting multi-threaded environments.</td></tr>
<tr><td>27</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">SOCHAPSTAT_0 Reserved</td></tr>
<tr><td>26</td><td>RW/1C/P</td><td>0</td><td>OUI</td><td colspan="2"><b>Overflow/Underflow Indicator: 0: The associated 32 bit counter has NOT rolled over since the last time it was cleared.</b> 1: The associated 32 bit counter HAS rolled over since the last time it was cleared. This bit is set by hardware, and cleared by software writing a 1 to it.</td></tr>
</table>

CSR Register Name: SOCHAPSTAT_4: SoCHAP Status Register 4				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 5070	**Offset End:** 5073
Bits	**Access**	**Default**	**Label**	**Bit Description**
25	RW/1C/P	0	CTI	**Command Trigger Indicator: 0: NO commands have been triggered since the last time this bit was cleared.** 1: A command WAS triggered since the last time this bit was cleared. Software can use this bit to know that a command that was pending earlier has been triggered. Once a command has been triggered, another command can be triggered to execute. This bit is set by hardware, and cleared by software writing a 1 to it.
24	RW/1C/P	0	THI	**Threshold Indicator: 0: No threshold event has been generated since the last time this bit was cleared.** 1: This counter generated a threshold event due to a true threshold condition compare since the last time this bit was cleared. This bit is set by hardware, and cleared by software writing a 1 to it.
23:0	RO	0	Reserved	SOCHAPSTAT_0 Reserved

13.10.31 SOCHAPDATA_4: SoCHAP Data Register 4

CSR Register Name: SOCHAPDATA_4: SoCHAP Data Register 4				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 5074	**Offset End:** 5077
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RW/P	0	CNTVAL	Counter Value: Contains either duration (number of clock ticks high) or occurrences (number of rising edges) contained in SoCHAP event counter 0 at time of sampling. The register is programmed to contain the threshold value that will be compared to the value in the event counter when non-0 condition codes have been selected.

13.10.32 SOCHAPCMD_5: SoCHAP Command Register 5

CSR Register Name: SOCHAPCMD_5: SoCHAP Command Register 5				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 5078	**Offset End:** 507B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:30	RO	0	Reserved	SOCHAPCMD_0 Reserved
29:28	RW	0	Reserved	SOCHAPCMD_0 Reserved
27	RW/P	0	PE	**PMI Enable: 0: PMIout is not asserted when the indicators are valid.** 1: The PMIout signal is asserted when the conditions enabled via the indicator enable bits are true.
26	RW/P	0	OUIE	Overflow/Underflow Indicator Enable: 0: No indication provided when a counter overflow or underflow occurs except for setting the Overflow/Underflow Indicator (OUI) status bit. 1: When the overflow/underflow condition is true, the counter will set the status bit and possibly assert the PMIout pin, which depending on SoC implementation, can be fed to another logic block or interrupt source.
25	RW/P	0	CTIE	Command Trigger Indicator Enable: 0: No indication provided when a command is triggered except for setting the Command Trigger Indicator (CTI) status bit. 1: When the command trigger condition is true, the counter will set the status bit and possibly assert the PMIout pin, which depending on SoC implementation, can be fed to another logic block or interrupt source.
24	RW/P	0	THIE	Threshold Indicator Enable: 0: No indication provided when threshold condition is true except for setting the Threshold Indicator (TI) status bit. 1: When the threshold condition is true, the counter will set the status bit and possibly assert the PMIout pin, which depending on SoC implementation, can be fed to another logic block or interrupt source.

<table>
<tr><td colspan="5"><b>CSR Register Name: SOCHAPCMD_5</b>: SoCHAP Command Register 5</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5078
<b>Offset End:</b> 507B</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th>Bit Description</th></tr>
<tr><td>23:21</td><td>RW/P</td><td>0</td><td>CC</td><td>Condition Code: This field contains the code that indicates what type of threshold compare is done between the counter and the data register. For all non-0 values of this field, the counter s data register will contain the threshold value. The outcome of this compare will generate a threshold event and potentially an interrupt if that capability is enabled.Bit 23 is for less than (<) Bit 22 is for equal (=) Bit 21 is for greater than (>) Select the proper bit mask for desired threshold condition: 000: False (no threshold compare) 001: Greater Than 010: Equal 011: Greater Than or Equal 100: Less Than 101: Not Equal 110: Less Than or Equal 111: True (always generate threshold event)</td></tr>
<tr><td>20</td><td>RW/P</td><td>0</td><td>SAC</td><td>Select ALL Counters: When this bit is set, the opcode in bits 19:16 is applied to all counters. When cleared, the opcode in bits 19:16 only apply to the current counter. The rest of the SoCHAP command register is not affected by the setting of this bit. 0: The opcode is applied only to the counter associated with this command register. 1: The opcode is applied to ALL counters. This means that every command register is written to with the same value that was written to this particular command register. This is particularly handy for resetting all counters with a single command or starting or stopping all counters simultaneously. This bit is only valid in Command Register 0 and should be reserved in all non-0 command registers. Globally executed commands (by setting this bit in Command Register 0) always override locally executed commands.</td></tr>
</table>

<table>
<tr><td colspan="5"><b>CSR Register Name: SOCHAPCMD_5</b>: SoCHAP Command Register 5</td></tr>
<tr><td><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5078</td><td><b>Offset End:</b> 507B</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th>Bit Description</th></tr>
<tr><td>19:16</td><td>RW/P</td><td>0</td><td>OPCODE</td><td>Opcode:
0000: Stop. The corresponding counter does not count.
0001: Start. The corresponding counter begins counting. A counter increments by ICV+1 if the corresponding increment event occurs or decrements by DCV+1 if the corresponding decrement event occurs. All duration type events toggle every SoCHAP unit clock tick for which the event is true. The desired increment and decrement events must be selected before this command executes.
0010: Sample. The corresponding counter value is latched into the corresponding data register, which can then be accessed by reading the appropriate data register. Counter functionality is unaffected by a Sample Opcode. The counter continues to count without being reset. If the Condition Code is NOT False then the Data Register is not written when a sample takes place. In other words the Data Register will NOT be updated with counter value if the Condition Code is > 0.
0100: Reset. The corresponding counter is reset to 0000 0000h. The 32 bit wide counter allows 4 billion clock ticks or occurrences to be counted between sample commands. When the counter rolls over, the overflow status bit will be set in the corresponding status register.
0101: Restart. The corresponding counter resets, and begins counting again. This is essentially a Reset & Start command. This functionality will facilitate histogram creation by allowing an event to trigger to clear the counter and resume counting with no further intervention.
0110: Sample & Restart. The Sample command happens and is followed immediately by the Restart command.
1111: Preload. The corresponding counter is set to the value that is located in the associated Data Register. This facilitates rollover and overflow validation. The counter remains in the same state when preloaded. If the counter was counting before the preload was executed it will continue to count after the preload. It is software s responsibility to ensure that the counter is in the desired state (example: execute stop command) prior to issuing a preload command.
All others reserved.</td></tr>
</table>

<table>
<tr><td colspan="6"><b>CSR Register Name: SOCHAPCMD_5</b>: SoCHAP Command Register 5</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5078</td><td><b>Offset End:</b> 507B</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>15</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">SOCHAPCMD_0 Reserved</td></tr>
<tr><td>14:13</td><td>RW</td><td>0</td><td>Reserved</td><td colspan="2">SOCHAPCMD_0 Reserved</td></tr>
<tr><td>12</td><td>RW/P</td><td>0</td><td>TBV</td><td colspan="2"><b>Trigger Bypass Valid Protocol: In this mode, the VIS valid protocol is not used.</b> The counter counts every clock regardless of valid data. Since signals from VIS may span multiple clocks this mode is not recommended for triggers since multiple back to back triggers will likely occur.</td></tr>
<tr><td>11:10</td><td>RW</td><td>0</td><td>Reserved</td><td colspan="2">SOCHAPCMD_0 Reserved</td></tr>
<tr><td>9</td><td>RW/P</td><td>0</td><td>TTM</td><td colspan="2">Threshold Trigger Mode: When this bit is set, if the operation of the trigger includes a sample (Sample, Sample and Restart opcodes), then the data value is only written if the threshold comparison is true. This mode can be used to store maximum and minimum values into the data register.</td></tr>
<tr><td>8:0</td><td>RW/P</td><td>0</td><td>CT</td><td colspan="2">Command Trigger: This field contains the Event Selection Code (ESC) that the unit will be required to detect before executing the opcode. The previously programmed opcode continues to execute until this command trigger is detected. The ESC of 000h (the default value) is a special case. This special ESC causes the command to be triggered immediately upon a write to the command register. The special command trigger also causes the target command to be executed (triggered) once and only once. All other (non-0) ESCs cause the command to be re-executed every time the trigger is detected. Only one trigger can occur per VIS clock. If the trigger source clock is running faster than the VIS clock and more than one trigger event occurs per VIS clock, these additional triggers will be dropped. See the Event Tables for valid values. ESC values are limited to the range of 000-1FFh</td></tr>
</table>

13.10.33 SOCHAPEV_5: SoCHAP Events Register 5

CSR Register Name: SOCHAPEV_5: SoCHAP Events Register 5				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 507C	**Offset End:** 507F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW/P	0	DECOCE	Decrement Occurrence Count Enable: 0: Decrement Duration Count: the counter is decremented for each clock for which the decrement event signal is asserted logic high. 1: Decrement Occurrence Count: the counter is decremented each time a rising edge of the decrement event signal is detected.
30	RW	0	Reserved	SOCHAPEV_0 Reserved
29	RW/P	0	DECEVOVR	**Decrement Event Override: When this bit is set, the decrement event is tied to logical 1 .** This in turn causes the counter to count the number clocks that occurred in the native domain of the signal.
28	RW/P	0	DECBYP	**Decrement Bypass Valid Protocol: In this mode, the VIS valid protocol is not used.** The counter decrements each clock regardless of valid data.
27:25	RW/P	0	DECVAL	**Decrement Counting Value:By setting this, the decrement value used by the counter can be changed.** Note, decrement value = DECVAL+1. 000: Decrement by 1 001: Decrement by 2 010: Decrement by 3 ... 111: Decrement by 8
24:16	RW/P	0	DECE	Decrement Event: This field contains the Event Selection Code (ESC) that the unit will be required to detect before decrementing the associated counter. This field is only applicable for opcodes that put the counter in a state where it is counting. See the Event tables for valid values. This limits ESC codes to the range of 000-1FFh.

CSR Register Name: SOCHAPEV_5: SoCHAP Events Register 5				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 507C **Offset End:** 507F
Bits	**Access**	**Default**	**Label**	**Bit Description**
15	RW/P	0	INCOCE	Increment Occurrence Count Enable: 0: Increment Duration Count: the counter is incremented for each clock for which the increment event signal is asserted logic high. 1: Increment Occurrence Count: the counter is incremented each time a rising edge of the increment event signal is detected.
14	RW	0	Reserved	SOCHAPEV_0 Reserved
13	RW/P	0	INCEVOVR	**Increment Event Override: When this bit is set, the increment event is tied to logical 1 .** This in turn causes the counter to count the number clocks that occurred in the native domain of the signal.
12	RW/P	0	INCBYP	**Increment Bypass Valid Protocol: In this mode, the VIS valid protocol is not used.** The counter increments each clock regardless of valid data.
11:9	RW/P	0	INCVAL	**Increment Counting Value: By setting this, the increment value used by the counter can be changed.** Note, increment value = INCVAL+1. 000: Increment by 1 001: Increment by 2 010: Increment by 3 ... 111: Increment by 8
8:0	RW/P	0	INCE	Increment Event: This field contains the Event Selection Code (ESC) that the unit will be required to detect before incrementing the associated counter. This field is only applicable for opcodes that put the counter in a state where it is counting. See the Event tables for valid values. This limits ESC codes to the range of 000-1FFh

13.10.34 SOCHAPSTAT_5: SoCHAP Status Register 5

<table>
<tr><td colspan="6"><b>CSR Register Name: SOCHAPSTAT_5</b>: SoCHAP Status Register 5</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5080</td><td><b>Offset End:</b> 5083</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>31</td><td>RW/1C/P</td><td>0</td><td>DCI</td><td colspan="2"><b>Decrement Counter Indicator: The decrement event was detected in the absence of the increment event.</b> The counter may or may not have decremented depending on the counter state when the decrement event occurred.</td></tr>
<tr><td>30</td><td>RW/1C/P</td><td>0</td><td>ICI</td><td colspan="2"><b>Increment Counter Indicator: The increment event was detected in the absence of the decrement event.</b> The counter may or may not have incremented depending on the counter state when the increment event occurred.</td></tr>
<tr><td>29</td><td>RO</td><td>0</td><td>CAI</td><td colspan="2">Counter Active Indicator:

0: The associated counter is in a state that does NOT allow it to be incremented or decremented if the appropriate event(s) are detected.

1: The associated counter is in a state that allows it to be incremented or decremented if the appropriate event(s) are detected. This bit is updated every clock.</td></tr>
<tr><td>28</td><td>RW/1C/P</td><td>0</td><td>IU</td><td colspan="2"><b>In Use: Software Mutex bit.</b> After a RESET, a read to this bit returns a 0. After the first read, subsequent reads will return a 1.
A write of a 1 to this bit will reset the next read value to 0.
Writing a 0 to this bit has no effect
Software can poll this bit until it reads a 0, and will then own the usage of the corresponding counter. This bit has no other effect on any SoCHAP Counter registers, and is only used as a semaphore among various independent software threads that may need to utilize this performance counter.
Software that reads this register, but does not intend to claim exclusive access, must write a 1 to this bit (if it reads a 0), in order to allow other software threads to claim it. This bit is particularly useful when supporting multi-threaded environments.</td></tr>
</table>

CSR Register Name: **SOCHAPSTAT_5**: SoCHAP Status Register 5			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 5080	**Offset End:** 5083

Bits	Access	Default	Label	Bit Description
27	RO	0	Reserved	SOCHAPSTAT_0 Reserved
26	RW/1C/P	0	OUI	**Overflow/Underflow Indicator:** **0: The associated 32 bit counter has NOT rolled over since the last time it was cleared.** 1: The associated 32 bit counter HAS rolled over since the last time it was cleared. This bit is set by hardware, and cleared by software writing a 1 to it.
25	RW/1C/P	0	CTI	**Command Trigger Indicator:** **0: NO commands have been triggered since the last time this bit was cleared.** 1: A command WAS triggered since the last time this bit was cleared. Software can use this bit to know that a command that was pending earlier has been triggered. Once a command has been triggered, another command can be triggered to execute. This bit is set by hardware, and cleared by software writing a 1 to it.
24	RW/1C/P	0	THI	**Threshold Indicator:** **0: No threshold event has been generated since the last time this bit was cleared.** 1: This counter generated a threshold event due to a true threshold condition compare since the last time this bit was cleared. This bit is set by hardware, and cleared by software writing a 1 to it.
23:0	RO	0	Reserved	SOCHAPSTAT_0 Reserved

13.10.35 SOCHAPDATA_5: SoCHAP Data Register 5

CSR Register Name: **SOCHAPDATA_5**: SoCHAP Data Register 5			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 5084	**Offset End:** 5087

Bits	Access	Default	Label	Bit Description
31:0	RW/P	0	CNTVAL	Counter Value: Contains either duration (number of clock ticks high) or occurrences (number of rising edges) contained in SoCHAP event counter 0 at time of sampling. The register is programmed to contain the threshold value that will be compared to the value in the event counter when non-0 condition codes have been selected.

13.10.36 SOCHAPCMD_6: SoCHAP Command Register 6

CSR Register Name: **SOCHAPCMD_6**: SoCHAP Command Register 6				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 5088	**Offset End:** 508B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:30	RO	0	Reserved	SOCHAPCMD_0 Reserved
29:28	RW	0	Reserved	SOCHAPCMD_0 Reserved
27	RW/P	0	PE	**PMI Enable:** **0: PMIout is not asserted when the indicators are valid.** 1: The PMIout signal is asserted when the conditions enabled via the indicator enable bits are true.
26	RW/P	0	OUIE	Overflow/Underflow Indicator Enable: 0: No indication provided when a counter overflow or underflow occurs except for setting the Overflow/Underflow Indicator (OUI) status bit. 1: When the overflow/underflow condition is true, the counter will set the status bit and possibly assert the PMIout pin, which depending on SoC implementation, can be fed to another logic block or interrupt source.

<table>
<tr><td colspan="5"><b>CSR Register Name: SOCHAPCMD_6</b>: SoCHAP Command Register 6</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5088</td><td><b>Offset End:</b> 508B</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>25</td><td>RW/P</td><td>0</td><td>CTIE</td><td colspan="2">Command Trigger Indicator Enable:
0: No indication provided when a command is triggered except for setting the Command Trigger Indicator (CTI) status bit.
1: When the command trigger condition is true, the counter will set the status bit and possibly assert the PMIout pin, which depending on SoC implementation, can be fed to another logic block or interrupt source.</td></tr>
<tr><td>24</td><td>RW/P</td><td>0</td><td>THIE</td><td colspan="2">Threshold Indicator Enable:
0: No indication provided when threshold condition is true except for setting the Threshold Indicator (TI) status bit.
1: When the threshold condition is true, the counter will set the status bit and possibly assert the PMIout pin, which depending on SoC implementation, can be fed to another logic block or interrupt source.</td></tr>
</table>

<table>
<tr><td colspan="6"><b>CSR Register Name: SOCHAPCMD_6</b>: SoCHAP Command Register 6</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5088</td><td><b>Offset End:</b> 508B</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>23:21</td><td>RW/P</td><td>0</td><td>CC</td><td colspan="2">Condition Code: This field contains the code that indicates what type of threshold compare is done between the counter and the data register. For all non-0 values of this field, the counter s data register will contain the threshold value. The outcome of this compare will generate a threshold event and potentially an interrupt if that capability is enabled.Bit 23 is for less than (<) Bit 22 is for equal (=) Bit 21 is for greater than (>) Select the proper bit mask for desired threshold condition:
000: False (no threshold compare)
001: Greater Than
010: Equal
011: Greater Than or Equal
100: Less Than
101: Not Equal
110: Less Than or Equal
111: True (always generate threshold event)</td></tr>
</table>

CSR Register Name: **SOCHAPCMD_6**: SoCHAP Command Register 6				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 5088	**Offset End:** 508B
Bits	**Access**	**Default**	**Label**	**Bit Description**
20	RW/P	0	SAC	Select ALL Counters: When this bit is set, the opcode in bits 19:16 is applied to all counters. When cleared, the opcode in bits 19:16 only apply to the current counter. The rest of the SoCHAP command register is not affected by the setting of this bit. 0: The opcode is applied only to the counter associated with this command register. 1: The opcode is applied to ALL counters. This means that every command register is written to with the same value that was written to this particular command register. This is particularly handy for resetting all counters with a single command or starting or stopping all counters simultaneously. This bit is only valid in Command Register 0 and should be reserved in all non-0 command registers. Globally executed commands (by setting this bit in Command Register 0) always override locally executed commands.

<table>
<tr><td colspan="6"><b>CSR Register Name: SOCHAPCMD_6</b>: SoCHAP Command Register 6</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5088</td><td><b>Offset End:</b> 508B</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>19:16</td><td>RW/P</td><td>0</td><td>OPCODE</td><td colspan="2">Opcode:
0000: Stop. The corresponding counter does not count.
0001: Start. The corresponding counter begins counting. A counter increments by ICV+1 if the corresponding increment event occurs or decrements by DCV+1 if the corresponding decrement event occurs. All duration type events toggle every SoCHAP unit clock tick for which the event is true. The desired increment and decrement events must be selected before this command executes.
0010: Sample. The corresponding counter value is latched into the corresponding data register, which can then be accessed by reading the appropriate data register. Counter functionality is unaffected by a Sample Opcode. The counter continues to count without being reset. If the Condition Code is NOT False then the Data Register is not written when a sample takes place. In other words the Data Register will NOT be updated with counter value if the Condition Code is > 0.
0100: Reset. The corresponding counter is reset to 0000 0000h. The 32 bit wide counter allows 4 billion clock ticks or occurrences to be counted between sample commands. When the counter rolls over, the overflow status bit will be set in the corresponding status register.
0101: Restart. The corresponding counter resets, and begins counting again. This is essentially a Reset & Start command. This functionality will facilitate histogram creation by allowing an event to trigger to clear the counter and resume counting with no further intervention.
0110: Sample & Restart. The Sample command happens and is followed immediately by the Restart command.
1111: Preload. The corresponding counter is set to the value that is located in the associated Data Register. This facilitates rollover and overflow validation. The counter remains in the same state when preloaded. If the counter was counting before the preload was executed it will continue to count after the preload. It is software s responsibility to ensure that the counter is in the desired state (example: execute stop command) prior to issuing a preload command.
All others reserved.</td></tr>
</table>

<table>
<tr><td colspan="5"><b>CSR Register Name: SOCHAPCMD_6</b>: SoCHAP Command Register 6</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5088
<b>Offset End:</b> 508B</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th>Bit Description</th></tr>
<tr><td>15</td><td>RO</td><td>0</td><td>Reserved</td><td>SOCHAPCMD_0 Reserved</td></tr>
<tr><td>14:13</td><td>RW</td><td>0</td><td>Reserved</td><td>SOCHAPCMD_0 Reserved</td></tr>
<tr><td>12</td><td>RW/P</td><td>0</td><td>TBV</td><td><b>Trigger Bypass Valid Protocol: In this mode, the VIS valid protocol is not used.</b> The counter counts every clock regardless of valid data. Since signals from VIS may span multiple clocks this mode is not recommended for triggers since multiple back to back triggers will likely occur.</td></tr>
<tr><td>11:10</td><td>RW</td><td>0</td><td>Reserved</td><td>SOCHAPCMD_0 Reserved</td></tr>
<tr><td>9</td><td>RW/P</td><td>0</td><td>TTM</td><td>Threshold Trigger Mode: When this bit is set, if the operation of the trigger includes a sample (Sample, Sample and Restart opcodes), then the data value is only written if the threshold comparison is true. This mode can be used to store maximum and minimum values into the data register.</td></tr>
<tr><td>8:0</td><td>RW/P</td><td>0</td><td>CT</td><td>Command Trigger: This field contains the Event Selection Code (ESC) that the unit will be required to detect before executing the opcode. The previously programmed opcode continues to execute until this command trigger is detected. The ESC of 000h (the default value) is a special case. This special ESC causes the command to be triggered immediately upon a write to the command register. The special command trigger also causes the target command to be executed (triggered) once and only once. All other (non-0) ESCs cause the command to be re-executed every time the trigger is detected. Only one trigger can occur per VIS clock. If the trigger source clock is running faster than the VIS clock and more than one trigger event occurs per VIS clock, these additional triggers will be dropped. See the Event Tables for valid values. ESC values are limited to the range of 000-1FFh</td></tr>
</table>

13.10.37 SOCHAPEV_6: SoCHAP Events Register 6

CSR Register Name: SOCHAPEV_6: SoCHAP Events Register 6				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 508C **Offset End:** 508F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW/P	0	DECOCE	Decrement Occurrence Count Enable: 0: Decrement Duration Count: the counter is decremented for each clock for which the decrement event signal is asserted logic high. 1: Decrement Occurrence Count: the counter is decremented each time a rising edge of the decrement event signal is detected.
30	RW	0	Reserved	SOCHAPEV_0 Reserved
29	RW/P	0	DECEVOVR	**Decrement Event Override: When this bit is set, the decrement event is tied to logical 1 .** This in turn causes the counter to count the number clocks that occurred in the native domain of the signal.
28	RW/P	0	DECBYP	**Decrement Bypass Valid Protocol: In this mode, the VIS valid protocol is not used.** The counter decrements each clock regardless of valid data.
27:25	RW/P	0	DECVAL	**Decrement Counting Value:By setting this, the decrement value used by the counter can be changed.** Note, decrement value = DECVAL+1. 000: Decrement by 1 001: Decrement by 2 010: Decrement by 3 ... 111: Decrement by 8
24:16	RW/P	0	DECE	Decrement Event: This field contains the Event Selection Code (ESC) that the unit will be required to detect before decrementing the associated counter. This field is only applicable for opcodes that put the counter in a state where it is counting. See the Event tables for valid values. This limits ESC codes to the range of 000-1FFh.
15	RW/P	0	INCOCE	Increment Occurrence Count Enable: 0: Increment Duration Count: the counter is incremented for each clock for which the increment event signal is asserted logic high. 1: Increment Occurrence Count: the counter is incremented each time a rising edge of the increment event signal is detected.

CSR Register Name: SOCHAPEV_6: SoCHAP Events Register 6				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 508C	**Offset End:** 508F
Bits	**Access**	**Default**	**Label**	**Bit Description**
14	RW	0	Reserved	SOCHAPEV_0 Reserved
13	RW/P	0	INCEVOVR	**Increment Event Override: When this bit is set, the increment event is tied to logical 1 .** This in turn causes the counter to count the number clocks that occurred in the native domain of the signal.
12	RW/P	0	INCBYP	**Increment Bypass Valid Protocol: In this mode, the VIS valid protocol is not used.** The counter increments each clock regardless of valid data.
11:9	RW/P	0	INCVAL	**Increment Counting Value: By setting this, the increment value used by the counter can be changed.** Note, increment value = INCVAL+1. 000: Increment by 1 001: Increment by 2 010: Increment by 3 ... 111: Increment by 8
8:0	RW/P	0	INCE	Increment Event: This field contains the Event Selection Code (ESC) that the unit will be required to detect before incrementing the associated counter. This field is only applicable for opcodes that put the counter in a state where it is counting. See the Event tables for valid values. This limits ESC codes to the range of 000-1FFh

13.10.38 SOCHAPSTAT_6: SoCHAP Status Register 6

CSR Register Name: SOCHAPSTAT_6: SoCHAP Status Register 6				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 5090	**Offset End:** 5093
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW/1C/P	0	DCI	**Decrement Counter Indicator: The decrement event was detected in the absence of the increment event.** The counter may or may not have decremented depending on the counter state when the decrement event occurred.

CSR Register Name: SOCHAPSTAT_6: SoCHAP Status Register 6				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 5090 **Offset End:** 5093
Bits	**Access**	**Default**	**Label**	**Bit Description**
30	RW/1C/P	0	ICI	**Increment Counter Indicator: The increment event was detected in the absence of the decrement event.** The counter may or may not have incremented depending on the counter state when the increment event occurred.
29	RO	0	CAI	Counter Active Indicator: 0: The associated counter is in a state that does NOT allow it to be incremented or decremented if the appropriate event(s) are detected. 1: The associated counter is in a state that allows it to be incremented or decremented if the appropriate event(s) are detected. This bit is updated every clock.
28	RW/1C/P	0	IU	**In Use: Software Mutex bit.** After a RESET, a read to this bit returns a 0. After the first read, subsequent reads will return a 1. A write of a 1 to this bit will reset the next read value to 0. Writing a 0 to this bit has no effect Software can poll this bit until it reads a 0, and will then own the usage of the corresponding counter. This bit has no other effect on any SoCHAP Counter registers, and is only used as a semaphore among various independent software threads that may need to utilize this performance counter. Software that reads this register, but does not intend to claim exclusive access, must write a 1 to this bit (if it reads a 0), in order to allow other software threads to claim it. This bit is particularly useful when supporting multi-threaded environments.
27	RO	0	Reserved	SOCHAPSTAT_0 Reserved
26	RW/1C/P	0	OUI	**Overflow/Underflow Indicator: 0: The associated 32 bit counter has NOT rolled over since the last time it was cleared.** 1: The associated 32 bit counter HAS rolled over since the last time it was cleared. This bit is set by hardware, and cleared by software writing a 1 to it.

<table>
<tr><td colspan="6"><b>CSR Register Name: SOCHAPSTAT_6</b>: SoCHAP Status Register 6</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5090</td><td><b>Offset End:</b> 5093</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>25</td><td>RW/1C/P</td><td>0</td><td>CTI</td><td colspan="2"><b>Command Trigger Indicator: 0: NO commands have been triggered since the last time this bit was cleared.</b> 1: A command WAS triggered since the last time this bit was cleared. Software can use this bit to know that a command that was pending earlier has been triggered. Once a command has been triggered, another command can be triggered to execute. This bit is set by hardware, and cleared by software writing a 1 to it.</td></tr>
<tr><td>24</td><td>RW/1C/P</td><td>0</td><td>THI</td><td colspan="2"><b>Threshold Indicator: 0: No threshold event has been generated since the last time this bit was cleared.</b> 1: This counter generated a threshold event due to a true threshold condition compare since the last time this bit was cleared. This bit is set by hardware, and cleared by software writing a 1 to it.</td></tr>
<tr><td>23:0</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">SOCHAPSTAT_0 Reserved</td></tr>
</table>

13.10.39 SOCHAPDATA_6: SoCHAP Data Register 6

<table>
<tr><td colspan="6"><b>CSR Register Name: SOCHAPDATA_6</b>: SoCHAP Data Register 6</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 5094</td><td><b>Offset End:</b> 5097</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:0</td><td>RW/P</td><td>0</td><td>CNTVAL</td><td colspan="2">Counter Value: Contains either duration (number of clock ticks high) or occurrences (number of rising edges) contained in SoCHAP event counter 0 at time of sampling. The register is programmed to contain the threshold value that will be compared to the value in the event counter when non-0 condition codes have been selected.</td></tr>
</table>

13.10.40 SOCHAPCMD_7: SoCHAP Command Register 7

CSR Register Name: SOCHAPCMD_7: SoCHAP Command Register 7				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 5098	**Offset End:** 509B

Bits	Access	Default	Label	Bit Description
31:30	RO	0	Reserved	SOCHAPCMD_0 Reserved
29:28	RW	0	Reserved	SOCHAPCMD_0 Reserved
27	RW/P	0	PE	**PMI Enable: 0: PMIout is not asserted when the indicators are valid.** 1: The PMIout signal is asserted when the conditions enabled via the indicator enable bits are true.
26	RW/P	0	OUIE	Overflow/Underflow Indicator Enable: 0: No indication provided when a counter overflow or underflow occurs except for setting the Overflow/Underflow Indicator (OUI) status bit. 1: When the overflow/underflow condition is true, the counter will set the status bit and possibly assert the PMIout pin, which depending on SoC implementation, can be fed to another logic block or interrupt source.
25	RW/P	0	CTIE	Command Trigger Indicator Enable: 0: No indication provided when a command is triggered except for setting the Command Trigger Indicator (CTI) status bit. 1: When the command trigger condition is true, the counter will set the status bit and possibly assert the PMIout pin, which depending on SoC implementation, can be fed to another logic block or interrupt source.
24	RW/P	0	THIE	Threshold Indicator Enable: 0: No indication provided when threshold condition is true except for setting the Threshold Indicator (TI) status bit. 1: When the threshold condition is true, the counter will set the status bit and possibly assert the PMIout pin, which depending on SoC implementation, can be fed to another logic block or interrupt source.

CSR Register Name: SOCHAPCMD_7: SoCHAP Command Register 7				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 5098 **Offset End:** 509B
Bits	**Access**	**Default**	**Label**	**Bit Description**
23:21	RW/P	0	CC	Condition Code: This field contains the code that indicates what type of threshold compare is done between the counter and the data register. For all non-0 values of this field, the counter s data register will contain the threshold value. The outcome of this compare will generate a threshold event and potentially an interrupt if that capability is enabled.Bit 23 is for less than (<) Bit 22 is for equal (=) Bit 21 is for greater than (>) Select the proper bit mask for desired threshold condition: 000: False (no threshold compare) 001: Greater Than 010: Equal 011: Greater Than or Equal 100: Less Than 101: Not Equal 110: Less Than or Equal 111: True (always generate threshold event)
20	RW/P	0	SAC	Select ALL Counters: When this bit is set, the opcode in bits 19:16 is applied to all counters. When cleared, the opcode in bits 19:16 only apply to the current counter. The rest of the SoCHAP command register is not affected by the setting of this bit. 0: The opcode is applied only to the counter associated with this command register. 1: The opcode is applied to ALL counters. This means that every command register is written to with the same value that was written to this particular command register. This is particularly handy for resetting all counters with a single command or starting or stopping all counters simultaneously. This bit is only valid in Command Register 0 and should be reserved in all non-0 command registers. Globally executed commands (by setting this bit in Command Register 0) always override locally executed commands.

CSR Register Name: SOCHAPCMD_7: SoCHAP Command Register 7				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 5098	**Offset End:** 509B
Bits	**Access**	**Default**	**Label**	**Bit Description**
19:16	RW/P	0	OPCODE	Opcode: 0000: Stop. The corresponding counter does not count. 0001: Start. The corresponding counter begins counting. A counter increments by ICV+1 if the corresponding increment event occurs or decrements by DCV+1 if the corresponding decrement event occurs. All duration type events toggle every SoCHAP unit clock tick for which the event is true. The desired increment and decrement events must be selected before this command executes. 0010: Sample. The corresponding counter value is latched into the corresponding data register, which can then be accessed by reading the appropriate data register. Counter functionality is unaffected by a Sample Opcode. The counter continues to count without being reset. If the Condition Code is NOT False then the Data Register is not written when a sample takes place. In other words the Data Register will NOT be updated with counter value if the Condition Code is > 0. 0100: Reset. The corresponding counter is reset to 0000 0000h. The 32 bit wide counter allows 4 billion clock ticks or occurrences to be counted between sample commands. When the counter rolls over, the overflow status bit will be set in the corresponding status register. 0101: Restart. The corresponding counter resets, and begins counting again. This is essentially a Reset & Start command. This functionality will facilitate histogram creation by allowing an event to trigger to clear the counter and resume counting with no further intervention. 0110: Sample & Restart. The Sample command happens and is followed immediately by the Restart command. 1111: Preload. The corresponding counter is set to the value that is located in the associated Data Register. This facilitates rollover and overflow validation. The counter remains in the same state when preloaded. If the counter was counting before the preload was executed it will continue to count after the preload. It is software s responsibility to ensure that the counter is in the desired state (example: execute stop command) prior to issuing a preload command. All others reserved.

CSR Register Name: SOCHAPCMD_7: SoCHAP Command Register 7				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 5098 **Offset End:** 509B
Bits	**Access**	**Default**	**Label**	**Bit Description**
15	RO	0	Reserved	SOCHAPCMD_0 Reserved
14:13	RW	0	Reserved	SOCHAPCMD_0 Reserved
12	RW/P	0	TBV	**Trigger Bypass Valid Protocol: In this mode, the VIS valid protocol is not used.** The counter counts every clock regardless of valid data. Since signals from VIS may span multiple clocks this mode is not recommended for triggers since multiple back to back triggers will likely occur.
11:10	RW	0	Reserved	SOCHAPCMD_0 Reserved
9	RW/P	0	TTM	Threshold Trigger Mode: When this bit is set, if the operation of the trigger includes a sample (Sample, Sample and Restart opcodes), then the data value is only written if the threshold comparison is true. This mode can be used to store maximum and minimum values into the data register.
8:0	RW/P	0	CT	Command Trigger: This field contains the Event Selection Code (ESC) that the unit will be required to detect before executing the opcode. The previously programmed opcode continues to execute until this command trigger is detected. The ESC of 000h (the default value) is a special case. This special ESC causes the command to be triggered immediately upon a write to the command register. The special command trigger also causes the target command to be executed (triggered) once and only once. All other (non-0) ESCs cause the command to be re-executed every time the trigger is detected. Only one trigger can occur per VIS clock. If the trigger source clock is running faster than the VIS clock and more than one trigger event occurs per VIS clock, these additional triggers will be dropped. See the Event Tables for valid values. ESC values are limited to the range of 000-1FFh

13.10.41 SOCHAPEV_7: SoCHAP Events Register 7

CSR Register Name: SOCHAPEV_7: SoCHAP Events Register 7				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 509C **Offset End:** 509F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW/P	0	DECOCE	Decrement Occurrence Count Enable: 0: Decrement Duration Count: the counter is decremented for each clock for which the decrement event signal is asserted logic high. 1: Decrement Occurrence Count: the counter is decremented each time a rising edge of the decrement event signal is detected.
30	RW	0	Reserved	SOCHAPEV_0 Reserved
29	RW/P	0	DECEVOVR	**Decrement Event Override: When this bit is set, the decrement event is tied to logical 1 .** This in turn causes the counter to count the number clocks that occurred in the native domain of the signal.
28	RW/P	0	DECBYP	**Decrement Bypass Valid Protocol: In this mode, the VIS valid protocol is not used.** The counter decrements each clock regardless of valid data.
27:25	RW/P	0	DECVAL	**Decrement Counting Value:By setting this, the decrement value used by the counter can be changed.** Note, decrement value = DECVAL+1. 000: Decrement by 1 001: Decrement by 2 010: Decrement by 3 ... 111: Decrement by 8
24:16	RW/P	0	DECE	Decrement Event: This field contains the Event Selection Code (ESC) that the unit will be required to detect before decrementing the associated counter. This field is only applicable for opcodes that put the counter in a state where it is counting. See the Event tables for valid values. This limits ESC codes to the range of 000-1FFh.

CSR Register Name: SOCHAPEV_7: SoCHAP Events Register 7

Bar: CSR_MTB_BAR | **Reset:** npk_rst_b | **Offset Start:** 509C | **Offset End:** 509F

Bits	Access	Default	Label	Bit Description
15	RW/P	0	INCOCE	Increment Occurrence Count Enable: 0: Increment Duration Count: the counter is incremented for each clock for which the increment event signal is asserted logic high. 1: Increment Occurrence Count: the counter is incremented each time a rising edge of the increment event signal is detected.
14	RW	0	Reserved	SOCHAPEV_0 Reserved
13	RW/P	0	INCEVOVR	**Increment Event Override: When this bit is set, the increment event is tied to logical 1 .** This in turn causes the counter to count the number clocks that occurred in the native domain of the signal.
12	RW/P	0	INCBYP	**Increment Bypass Valid Protocol: In this mode, the VIS valid protocol is not used.** The counter increments each clock regardless of valid data.
11:9	RW/P	0	INCVAL	**Increment Counting Value: By setting this, the increment value used by the counter can be changed.** Note, increment value = INCVAL+1. 000: Increment by 1 001: Increment by 2 010: Increment by 3 ... 111: Increment by 8
8:0	RW/P	0	INCE	Increment Event: This field contains the Event Selection Code (ESC) that the unit will be required to detect before incrementing the associated counter. This field is only applicable for opcodes that put the counter in a state where it is counting. See the Event tables for valid values. This limits ESC codes to the range of 000-1FFh

13.10.42 SOCHAPSTAT_7: SoCHAP Status Register 7

CSR Register Name: SOCHAPSTAT_7: SoCHAP Status Register 7				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 50A0	**Offset End:** 50A3
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW/1C/P	0	DCI	**Decrement Counter Indicator: The decrement event was detected in the absence of the increment event.** The counter may or may not have decremented depending on the counter state when the decrement event occurred.
30	RW/1C/P	0	ICI	**Increment Counter Indicator: The increment event was detected in the absence of the decrement event.** The counter may or may not have incremented depending on the counter state when the increment event occurred.
29	RO	0	CAI	Counter Active Indicator: 0: The associated counter is in a state that does NOT allow it to be incremented or decremented if the appropriate event(s) are detected. 1: The associated counter is in a state that allows it to be incremented or decremented if the appropriate event(s) are detected. This bit is updated every clock.
28	RW/1C/P	0	IU	**In Use: Software Mutex bit.** After a RESET, a read to this bit returns a 0. After the first read, subsequent reads will return a 1. A write of a 1 to this bit will reset the next read value to 0. Writing a 0 to this bit has no effect Software can poll this bit until it reads a 0, and will then own the usage of the corresponding counter. This bit has no other effect on any SoCHAP Counter registers, and is only used as a semaphore among various independent software threads that may need to utilize this performance counter. Software that reads this register, but does not intend to claim exclusive access, must write a 1 to this bit (if it reads a 0), in order to allow other software threads to claim it. This bit is particularly useful when supporting multi-threaded environments.

CSR Register Name: SOCHAPSTAT_7: SoCHAP Status Register 7			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 50A0	**Offset End:** 50A3

Bits	Access	Default	Label	Bit Description
27	RO	0	Reserved	SOCHAPSTAT_0 Reserved
26	RW/1C/P	0	OUI	**Overflow/Underflow Indicator:** **0: The associated 32 bit counter has NOT rolled over since the last time it was cleared.** 1: The associated 32 bit counter HAS rolled over since the last time it was cleared. This bit is set by hardware, and cleared by software writing a 1 to it.
25	RW/1C/P	0	CTI	**Command Trigger Indicator:** **0: NO commands have been triggered since the last time this bit was cleared.** 1: A command WAS triggered since the last time this bit was cleared. Software can use this bit to know that a command that was pending earlier has been triggered. Once a command has been triggered, another command can be triggered to execute. This bit is set by hardware, and cleared by software writing a 1 to it.
24	RW/1C/P	0	THI	**Threshold Indicator:** **0: No threshold event has been generated since the last time this bit was cleared.** 1: This counter generated a threshold event due to a true threshold condition compare since the last time this bit was cleared. This bit is set by hardware, and cleared by software writing a 1 to it.
23:0	RO	0	Reserved	SOCHAPSTAT_0 Reserved

13.10.43 SOCHAPDATA_7: SoCHAP Data Register 7

CSR Register Name: SOCHAPDATA_7: SoCHAP Data Register 7			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 50A4	**Offset End:** 50A7

Bits	Access	Default	Label	Bit Description
31:0	RW/P	0	CNTVAL	Counter Value: Contains either duration (number of clock ticks high) or occurrences (number of rising edges) contained in SoCHAP event counter 0 at time of sampling. The register is programmed to contain the threshold value that will be compared to the value in the event counter when non-0 condition codes have been selected.

13.10.44 SOCHAPPKTCTRL: SoCHAP Packet Control Register

CSR Register Name: SOCHAPPKTCTRL: SoCHAP Packet Control Register				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 50A8	**Offset End:** 50AB
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:24	RO	0	RSVD1	SOCHAPPKTCTRL Reserved
23:16	RW/P	0	PKTTYPESEL	**Packet Type Select.** Each bit location corresponds to a SoCHAP counter and specifies the type field value to be written in each of the counter s output packets. The encoding for each bit is as follows: 0: Don't timestamp this counter s packets. 1: Timestamp all packets for this counter
15:8	RO	0	RSVD2	SOCHAPPKTCTRL Reserved
7:0	RW/P	0	SPLCNTREN	**Sample Counter Enable.** Each bit location corresponds to a SoCHAP counter to be sampled when its trigger asserts if the SoCHAP storeen is asserted. 0: Means the corresponding counter will not be packetized and sent to GTH 1: Means the corresponding counter s data register, OUI field, and THI field will be packetized and sent to GTH

13.11 ODLA Register

13.11.1 ODLA Registers Summary

ODLA Registers			
Offset Start	Offset End	Symbol	Register Name/Function
6000	6003	ODLACAP	ODLA Capability Register
6004	6007	OSMC	ODLA Storage Mode Control
6010	6013	TTSS	Timestamp Trigger Snap Shot
6014	6017	OSTAT	ODLA Status Register
6018	601B	ARBGNTCRED	Arbiter Grant Credit Register
602C	602F	TTSSEXT	Timestamp Trigger Snap Shot Extended Register
6030	6033	CTSS	Current Timestamp Snap Shot Register
6034	6037	CTSSEXT	Current Timestamp Snap Shot Extended Register
6080	6083	VCAPCTL	VIS Capture Control Register
6084	6087	VCAPREG_0	VIS Capture Register 0
6088	608B	VCAPREG_1	VIS Capture Register 1
608C	608F	VCAPREG_2	VIS Capture Register 2
6090	6093	VCAPREG_3	VIS Capture Register 3
6100	6103	LNSTORECTRL_0	Lane Storage Control Register 0
6104	6107	LNSTORECTRL_1	Lane Storage Control Register 1
6108	610B	LNSTORECTRL_2	Lane Storage Control Register 2
610C	610F	LNSTORECTRL_3	Lane Storage Control Register 3
6110	6113	LNSTORECTRL_4	Lane Storage Control Register 4
6114	6117	LNSTORECTRL_5	Lane Storage Control Register 5
6118	611B	LNSTORECTRL_6	Lane Storage Control

ODLA Registers			
Offset Start	**Offset End**	**Symbol**	**Register Name/Function**
			Register 6
611C	611F	LNSTORECTRL_7	Lane Storage Control Register 7
6180	6183	LNFTLCTL_0	Lane Filter Control Register 0
6184	6187	LNFTLCTL_1	Lane Filter Control Register 1
6188	618B	LNFTLCTL_2	Lane Filter Control Register 2
618C	618F	LNFTLCTL_3	Lane Filter Control Register 3
6190	6193	LNFTLCTL_4	Lane Filter Control Register 4
6194	6197	LNFTLCTL_5	Lane Filter Control Register 5
6198	619B	LNFTLCTL_6	Lane Filter Control Register 6
619C	619F	LNFTLCTL_7	Lane Filter Control Register 7

13.11.2 ODLACAP: ODLA Capability Register

CSR Register Name: ODLACAP: ODLA Capability Register				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 6000	**Offset End:** 6003
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:28	RO	1	MaterID	ODLA_MASTERID parameter value.
27:26	RO	0	GblBufDepthVal	**This bit field specifies the buffer depth of each global packet resource buffer.** 00: Depth is 4 packets 01: Depth is 5 packets 10: Depth is 6 packets 11: Depth is 7 packets
25:14	RO	0	Reserved	ODLACAP Reserved

<table>
<tr><td colspan="6"><b>CSR Register Name: ODLACAP</b>: ODLA Capability Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 6000</td><td><b>Offset End:</b> 6003</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>13:12</td><td>RO</td><td>2</td><td>LaneBufDepthVal</td><td colspan="2">This bit field identifies the depth of the packetization buffers in the ODLA controller.
00: Depth is 2 bytes
01: Depth is 3 bytes
10: Depth is 4 bytes
11: Reserved.</td></tr>
<tr><td>11:5</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">ODLACAP Reserved</td></tr>
<tr><td>4:0</td><td>RO</td><td>8</td><td>NumOfLanesVal</td><td colspan="2">Identifies the number of lanes that the ODLA controller
[0h -1h]: Not applicable. A minimum of 2 lanes is necessary for ODLA to operate.
[2h]: Number of lanes is equal to 2
[3h]: Number of lanes is equal to 3
[4h]: Number of lanes is equal to 4. A typical implementation number of lanes is 4.

[Fh]: Number of lanes is equal to 15
[10h]: Number of lanes is equal to 16
[11-1Fh]: Reserved, these values are not allowed.</td></tr>
</table>

13.11.3 OSMC: ODLA Storage Mode Control

<table>
<tr><td colspan="6"><b>CSR Register Name: OSMC</b>: ODLA Storage Mode Control</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 6004</td><td><b>Offset End:</b> 6007</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31</td><td>RW/P</td><td>0</td><td>TSCSrcSel</td><td colspan="2"><b>Timestamp counter source select.</b> This field specifies the source for the timestamp value written to timestamp packets.
0: Internal timestamp counter
1: External timestamp counter</td></tr>
</table>

<table>
<tr><td colspan="6"><b>CSR Register Name: OSMC</b>: ODLA Storage Mode Control</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 6004</td><td><b>Offset End:</b> 6007</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>30</td><td>RW/P</td><td>0</td><td>DeepCompDis</td><td colspan="2"><b>Deep Compression Disable.</b> When the compressed count (CC) exceeds the maximum 8-bit value of 255, deep compression may be requested from the global logic. When deep compression is granted, the lane can enter deep compression and discontinue sending lane packets. Setting this bit to 1 will disable deep compression requests for all lanes which results in lanes storing data samples and compression counts in data packets instead of entering deep compression.
0: Deep compression is enabled
1: Deep compression is disabled, CC-sample pairs will continue been stored.</td></tr>
<tr><td>29:28</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">OSMC Reserved</td></tr>
<tr><td>27</td><td>RW/P</td><td>0</td><td>SampleTS</td><td colspan="2"><b>Sample Timestamp.</b> Writing 1b1 causes the current timestamp value to be written to the CTSS and CTSSEXT registers so that it can be read by software. This bit is automatically cleared one clock after being written to remove the need for a separate write to clear it before getting subsequent snapshot updates. Writing zero has no effect.</td></tr>
<tr><td>26</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">OSMC Reserved</td></tr>
</table>

CSR Register Name: OSMC: ODLA Storage Mode Control				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 6004 **Offset End:** 6007
Bits	**Access**	**Default**	**Label**	**Bit Description**
25:23	RW/P	3	TSinjRate	**Timestamp injection rate.** This field specifies the maximum number of ODLA clocks that can occur between global timestamp injections by configuring a periodic timestamp injection counter. When the counter reaches the specified value, a global timestamp injection will occur and the counter will be reset. Global timestamp injections occurring for reasons other than the counter reaching its terminal count value will also reset the counter. This provides a minimum global timestamp injection rate. 000: Do not inject additional global timestamps. Normal ODLA operation will insert timestamps when required. 001: Inject a global timestamp at least every 1k ODLA clocks. 010: Inject a global timestamp at least every 2k ODLA clocks. … 111: Inject a global timestamp at least every 64k ODLA clocks.
22	RW/P	0	ClkOffsetPktEn	**Clock Offset Packet Enable.** This bit allows for the creation of Clock Offset Packets on global timestamp injections. 0: Disable Clock Offset Packets 1: Enable Clock Offset Packets
21	RO	0	Reserved	OSMC Reserved
20	RW/P	0	ClkOffDetDis	**Clock-Off Detection Disable.** This bit disables detection of a missing clock on all the lanes while global storage is enabled. 0: Clock-Off detection enabled. 1: Clock-Off detection disabled.
19	RO	0	Reserved	OSMC Reserved
18	RW/P	0	CompEn	**Compression Enable.** This bit enables store-on-change compression. 0: Compression is disabled 1: Compression is enabled

CSR Register Name: OSMC: ODLA Storage Mode Control				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 6004 **Offset End:** 6007
Bits	**Access**	**Default**	**Label**	**Bit Description**
17:15	RW/P	0	LatAdj	**Latency Adjustment.** The ODLA control block and the North Peak trigger unit must be aligned for the ODLA to be effective, therefore NPK trigger unit may require additional latency states. Maximum number is defined by LATENCY parameter, 0 <LATENCY<=7. LatAdj=0 will mean no addition latency states, data will be capture on the same trigger clock cycle. If LatAdj=1, data one clock earlier than the trigger will be captured, if LatAdj=2, two clocks earlier and so on. 000: Zero additional ODLA latency clocks. 001: One additional ODLA latency clocks. : 111: Seven additional ODLA latency clocks.
14	RW/P	0	SwRst	**Software Reset.** When this is asserted the ODLA control logic is forced into reset. It remains in reset until this bit is cleared. Setting this bit will not reset the configuration registers or the Global Timestamp Counter. Only a hardware reset will reset the Global Timestamp Counter and clear some of the configuration registers.
13	RW/P	0	SwSusp	**Software Suspend.** This bit ends a data capture that has not finished yet. If some data was stored, SW Suspend will allow OSMC.DataReady get set and data could be extracted. 0: Normal ODLA operation. 1: ODLA operation suspended.
12:0	RO	0	Reserved	OSMC Reserved

13.11.4 TTSS: Timestamp Trigger Snap Shot

<table>
<tr><td colspan="6"><b>CSR Register Name:</b> <b>TTSS</b>: Timestamp Trigger Snap Shot</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 6010</td><td><b>Offset End:</b> 6013</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:0</td><td>RW/1C/P/V</td><td>0</td><td>TrigTsVal</td><td colspan="2">Trigger Timestamp Snapshot DWord Value.</td></tr>
</table>

13.11.5 OSTAT: ODLA Status Register

<table>
<tr><td colspan="6"><b>CSR Register Name:</b> <b>OSTAT</b>: ODLA Status Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 6014</td><td><b>Offset End:</b> 6017</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:5</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">OSTAT Reserved</td></tr>
<tr><td>4</td><td>RO/V</td><td>0</td><td>TsOvrFlw</td><td colspan="2"><b>Time stamp Overflow.</b> This bit indicates that an overflow of the entire time stamp counter has occurred.
0: no overflow detected
1: An overflow has occurred.</td></tr>
<tr><td>3</td><td>RO/V</td><td>0</td><td>WeStat</td><td colspan="2"><b>ODLA Write Enable Status.</b> This bit indicates that at least one output write enable assertion has occurred to trace buffer during the current capture cycle.
0: Write Enable has not asserted.
1: Write Enable has asserted at least once. Only the first assertion is captured. There may have been more than one assertion.</td></tr>
<tr><td>2</td><td>RO/V</td><td>0</td><td>Reserved</td><td colspan="2">OSTAT Reserved</td></tr>
<tr><td>1</td><td>RO/V</td><td>0</td><td>TrigStat</td><td colspan="2"><b>ODLA Trigger Status.</b> This bit indicates that at least one trigger assertion has occurred.
0: Trigger has not asserted.
1: Trigger has asserted at least once. There may have been more than one assertion.</td></tr>
</table>

CSR Register Name: **OSTAT**: ODLA Status Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 6014	**Offset End:** 6017

Bits	Access	Default	Label	Bit Description
0	RO/V	0	DataReady	**ODLA Data Ready status.** This bit indicates that the capture cycle is complete and debug data is ready for unload. Should be driven high whenever storeen is low and the content of all packet buffers are empty 0: ODLA disabled or enabled but capture cycle is not completed yet. 1: ODLA capture cycle completed.

13.11.6 ARBGNTCRED: Arbiter Grant Credit Register

CSR Register Name: **ARBGNTCRED**: Arbiter Grant Credit Register			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 6018	**Offset End:** 601B

Bits	Access	Default	Label	Bit Description
31:30	RW/P	0	GntCreditsLn_15	**Grant Credits Lane 15.** GntCreditsLn[0] for details.
29:28	RW/P	0	GntCreditsLn_14	**Grant Credits Lane 14.** GntCreditsLn[0] for details.
27:26	RW/P	0	GntCreditsLn_13	**Grant Credits Lane 13.** GntCreditsLn[0] for details.
25:24	RW/P	0	GntCreditsLn_12	**Grant Credits Lane 12.** GntCreditsLn[0] for details.
23:22	RW/P	0	GntCreditsLn_11	**Grant Credits Lane 11.** GntCreditsLn[0] for details.
21:20	RW/P	0	GntCreditsLn_10	**Grant Credits Lane 10.** GntCreditsLn[0] for details.
19:18	RW/P	0	GntCreditsLn_9	**Grant Credits Lane 9.** GntCreditsLn[0] for details.
17:16	RW/P	0	GntCreditsLn_8	**Grant Credits Lane 8.** GntCreditsLn[0] for details.
15:14	RW/P	0	GntCreditsLn_7	**Grant Credits Lane 7.** GntCreditsLn[0] for details.
13:12	RW/P	0	GntCreditsLn_6	**Grant Credits Lane 6.** GntCreditsLn[0] for details.

CSR Register Name: ARBGNTCRED: Arbiter Grant Credit Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 6018 / **Offset End:** 601B
Bits	**Access**	**Default**	**Label**	**Bit Description**
11:10	RW/P	0	GntCreditsLn_5	**Grant Credits Lane 5.** GntCreditsLn[0] for details.
9:8	RW/P	0	GntCreditsLn_4	**Grant Credits Lane 4.** GntCreditsLn[0] for details.
7:6	RW/P	0	GntCreditsLn_3	**Grant Credits Lane 3.** GntCreditsLn[0] for details.
5:4	RW/P	0	GntCreditsLn_2	**Grant Credits Lane 2.** GntCreditsLn[0] for details.
3:2	RW/P	0	GntCreditsLn_1	**Grant Credits Lane 1.** GntCreditsLn[0] for details.
1:0	RW/P	0	GntCreditsLn_0	**Grant Credits for Lane 0.** 00: No back-to-back grants. The arbiter will grant one request at a time for this lane. 01: One back-to-back grant. The arbiter will grant one request and if there are more packets in the buffer, it will grant one more back-to-back request for this lane. 10: Two back-to-back grants. 11: Three back-to-back grants.

13.11.7 TTSSEXT: Timestamp Trigger Snap Shot Extended Register

CSR Register Name: TTSSEXT: Timestamp Trigger Snap Shot Extended Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 602C / **Offset End:** 602F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:16	RO	0	Reserved	TTSSEXT Reserved
15:8	RW/1C/P/V	0	TrigTsByt6	**Trigger Timestamp Snapshot High DWord Value.** This field contains the value of the upper timestamp at the first trigger assertion. (timestamp[47:40]) If multiple trigger assertions occured, it will contain the value of the upper timestamp at the 1st trigger.

CSR Register Name: TTSSEXT: Timestamp Trigger Snap Shot Extended Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 602C	**Offset End:** 602F
Bits	**Access**	**Default**	**Label**	**Bit Description**
7:0	RW/1C/P/V	0	TrigTsByt5	**Trigger Timestamp Snapshot High DWord Value.** This field contains the value of the upper timestamp at the first trigger assertion (timestamp[39:32]). If multiple trigger assertions occured, it will contain the value of the upper timestamp at the 1st trigger.

13.11.8 CTSS: Current Timestamp Snap Shot Register

CSR Register Name: CTSS: Current Timestamp Snap Shot Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 6030	**Offset End:** 6033
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RW/1C/P	0	CurTsVal	**Current Timestamp Snapshot DWord Value.** Updated with current timestamp value whenever OSMC register bit SampleTs is written with a 1b1

13.11.9 CTSSEXT: Current Timestamp Snap Shot Extended Register

CSR Register Name: CTSSEXT: Current Timestamp Snap Shot Extended Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 6034	**Offset End:** 6037
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:16	RO	0	Reserved	CTSSEXT Reserved
15:0	RW/1C/P	0	CurTsValExt	Upper 2 bytes of Current Timestamp Snapshot Value

13.11.10 VCAPCTL: VIS Capture Control Register

CSR Register Name: VCAPCTL: VIS Capture Control Register				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 6080	**Offset End:** 6083
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:4	RO	0	Reserved	VCAPCTL Reserved
3	RW/P	0	VcapClr	**VIS Clear capture register.** This bit will clear all bits in all byte lanes of the VCAPREG[3:0]. 0: Do not clear 1: Clear all
2:1	RW/P	0	VcapMode	**VIS Capture Mode: This bit determines the mode of operation.** 00: Disable capture 01: Capture on trigger edge. Capture on the rising edge of the trigger source. VcapReg will continue to capture on each subsequent edge. 10: Capture on trigger level. Capture continuously when the trigger is logic 1. VcapReg will continue to capture on each subsequent level. 11: Capture continuously upon entering mode (no trigger). The sticky VCAPREG registers will retain the value until cleared.
0	RW/P	0	VcapEn	**VIS Capture Enable: This bit enables the VIS capture DFV feature.** 0: Disable VIS capture 1: Enable VIS capture

13.11.11 VCAPREG_0: VIS Capture Register 0

CSR Register Name: VCAPREG_0: VIS Capture Register 0			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 6084	**Offset End:** 6087

Bits	Access	Default	Label	Bit Description
31:24	RW/1C/P	0	VcapByte3	**VIS Capture Value Byte3:** description same as Byte0
23:16	RW/1C/P	0	VcapByte2	**VIS Capture Value Byte2:** description same as Byte0
15:8	RW/1C/P	0	VcapByte1	**VIS Capture Value Byte1:** description same as Byte0
7:0	RW/1C/P	0	VcapByte0	**VIS Capture Value Byte0: This field captures one byte on the input VIS bus at the Lakemore top level.** All bytes operate with same mode either trigger based or saturated based. This bit field may be cleared by writing logic1 into the specific bits/byte or with a software clear in the control register.

13.11.12 VCAPREG_1: VIS Capture Register 1

CSR Register Name: VCAPREG_1: VIS Capture Register 1				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 6088	**Offset End:** 608B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:24	RW/1C/P	0	VcapByte7	**VIS Capture Value Byte7:** description same as Byte4
23:16	RW/1C/P	0	VcapByte6	**VIS Capture Value Byte6:** description same as Byte4
15:8	RW/1C/P	0	VcapByte5	**VIS Capture Value Byte5:** description same as Byte4
7:0	RW/1C/P	0	VcapByte4	**VIS Capture Value Byte4: This field captures one byte on the input VIS bus at the Lakemore top level.** All bytes operate with same mode either trigger based or saturated based. This bit field may be cleared by writing logic1 into the specific bits/byte or with a software clear in the control register.

13.11.13 VCAPREG_2: VIS Capture Register 2

CSR Register Name: VCAPREG_2: VIS Capture Register 2			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 608C	**Offset End:** 608F

Bits	Access	Default	Label	Bit Description
31:24	RW/1C/P	0	VcapByte11	VIS Capture Value Byte11: same as Byte7
23:16	RW/1C/P	0	VcapByte10	VIS Capture Value Byte10: same as Byte6
15:8	RW/1C/P	0	VcapByte9	VIS Capture Value Byte9: same as Byte5
7:0	RW/1C/P	0	VcapByte8	**VIS Capture Value Byte8: This field captures one byte on the input VIS bus at the Lakemore top level.** All bytes operate with same mode either trigger based or saturated based. This bit field may be cleared by writing logic1 into the specific bits/byte or with a software clear in the control register.

13.11.14 VCAPREG_3: VIS Capture Register 3

CSR Register Name: VCAPREG_3: VIS Capture Register 3			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 6090	**Offset End:** 6093

Bits	Access	Default	Label	Bit Description
31:24	RW/1C/P	0	VcapByte15	VIS Capture Value Byte15: same as Byte12
23:16	RW/1C/P	0	VcapByte14	VIS Capture Value Byte14: same as Byte12
15:8	RW/1C/P	0	VcapByte13	VIS Capture Value Byte13: same as Byte12
7:0	RW/1C/P	0	VcapByte12	**VIS Capture Value Byte12: This field captures one byte on the input VIS bus at the Lakemore top level.** All bytes operate with same mode either trigger based or saturated based. This bit field may be cleared by writing logic1 into the specific bits/byte or with a software clear in the control register.

13.11.15 LNSTORECTRL_0: Lane Storage Control Register 0

CSR Register Name: LNSTORECTRL_0: Lane Storage Control Register 0			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 6100	**Offset End:** 6103

Bits	Access	Default	Label	Bit Description
31:9	RO	0	Reserved	LNSTORECTRL Reserved
8	RW/P	0	LaneEn	**Lane Enable.** This bit will enable this lane for arbiter control. 0: Disable. This lane will not be observed by the arbitration logic and no data from this lane will be stored by the ODLA controller 1: Enable. This lane will be arbitrated with other enabled lanes and its data consumed by the ODLA controller.
7:0	RW/P	1F	ClkOffComp	**Clock Off Compare.** This bit field contains the 8-bit match value used by the clock off detection compare logic to set the maximum number of ODLA clocks since the last lane valid. When the clock off counter matches the compare value, a global timestamp injection will occur to indicate the time that the lane s source clock turned off. 0: Means the corresponding bit must be logic 0 to match. 1: Means the corresponding bit must be logic 1 to match.

13.11.16 LNSTORECTRL_1: Lane Storage Control Register 1

<table>
<tr><td colspan="6"><b>CSR Register Name: LNSTORECTRL_1</b>: Lane Storage Control Register 1</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 6104</td><td><b>Offset End:</b> 6107</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:9</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">LNSTORECTRL Reserved</td></tr>
<tr><td>8</td><td>RW/P</td><td>0</td><td>LaneEn</td><td colspan="2"><b>Lane Enable.</b> This bit will enable this lane for arbiter control.
0: Disable. This lane will not be observed by the arbitration logic and no data from this lane will be stored by the ODLA controller
1: Enable. This lane will be arbitrated with other enabled lanes and its data consumed by the ODLA controller.</td></tr>
</table>

CSR Register Name: **LNSTORECTRL_1**: Lane Storage Control Register 1				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 6104 / **Offset End:** 6107
Bits	**Access**	**Default**	**Label**	**Bit Description**
7:0	RW/P	1F	ClkOffComp	**Clock Off Compare.** This bit field contains the 8-bit match value used by the clock off detection compare logic to set the maximum number of ODLA clocks since the last lane valid. When the clock off counter matches the compare value, a global timestamp injection will occur to indicate the time that the lane s source clock turned off. 0: Means the corresponding bit must be logic 0 to match. 1: Means the corresponding bit must be logic 1 to match.

13.11.17 LNSTORECTRL_2: Lane Storage Control Register 2

CSR Register Name: **LNSTORECTRL_2**: Lane Storage Control Register 2				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 6108 / **Offset End:** 610B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:9	RO	0	Reserved	LNSTORECTRL Reserved
8	RW/P	0	LaneEn	**Lane Enable.** This bit will enable this lane for arbiter control. 0: Disable. This lane will not be observed by the arbitration logic and no data from this lane will be stored by the ODLA controller 1: Enable. This lane will be arbitrated with other enabled lanes and its data consumed by the ODLA controller.

CSR Register Name: LNSTORECTRL_2: Lane Storage Control Register 2				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 6108	**Offset End:** 610B
Bits	**Access**	**Default**	**Label**	**Bit Description**
7:0	RW/P	1F	ClkOffComp	**Clock Off Compare.** This bit field contains the 8-bit match value used by the clock off detection compare logic to set the maximum number of ODLA clocks since the last lane valid. When the clock off counter matches the compare value, a global timestamp injection will occur to indicate the time that the lane s source clock turned off. 0: Means the corresponding bit must be logic 0 to match. 1: Means the corresponding bit must be logic 1 to match.

13.11.18 LNSTORECTRL_3: Lane Storage Control Register 3

CSR Register Name: LNSTORECTRL_3: Lane Storage Control Register 3				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 610C	**Offset End:** 610F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:9	RO	0	Reserved	LNSTORECTRL Reserved
8	RW/P	0	LaneEn	**Lane Enable.** This bit will enable this lane for arbiter control. 0: Disable. This lane will not be observed by the arbitration logic and no data from this lane will be stored by the ODLA controller 1: Enable. This lane will be arbitrated with other enabled lanes and its data consumed by the ODLA controller.

<table>
<tr><td colspan="6"><b>CSR Register Name:</b> <b>LNSTORECTRL_3</b>: Lane Storage Control Register 3</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 610C</td><td><b>Offset End:</b> 610F</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>7:0</td><td>RW/P</td><td>1F</td><td>ClkOffComp</td><td colspan="2"><b>Clock Off Compare.</b> This bit field contains the 8-bit match value used by the clock off detection compare logic to set the maximum number of ODLA clocks since the last lane valid. When the clock off counter matches the compare value, a global timestamp injection will occur to indicate the time that the lane s source clock turned off.
0: Means the corresponding bit must be logic 0 to match.
1: Means the corresponding bit must be logic 1 to match.</td></tr>
</table>

13.11.19 LNSTORECTRL_4: Lane Storage Control Register 4

<table>
<tr><td colspan="6"><b>CSR Register Name:</b> <b>LNSTORECTRL_4</b>: Lane Storage Control Register 4</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 6110</td><td><b>Offset End:</b> 6113</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:9</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">LNSTORECTRL Reserved</td></tr>
<tr><td>8</td><td>RW/P</td><td>0</td><td>LaneEn</td><td colspan="2"><b>Lane Enable.</b> This bit will enable this lane for arbiter control.
0: Disable. This lane will not be observed by the arbitration logic and no data from this lane will be stored by the ODLA controller
1: Enable. This lane will be arbitrated with other enabled lanes and its data consumed by the ODLA controller.</td></tr>
</table>

CSR Register Name: LNSTORECTRL_4: Lane Storage Control Register 4				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 6110	**Offset End:** 6113
Bits	**Access**	**Default**	**Label**	**Bit Description**
7:0	RW/P	1F	ClkOffComp	**Clock Off Compare.** This bit field contains the 8-bit match value used by the clock off detection compare logic to set the maximum number of ODLA clocks since the last lane valid. When the clock off counter matches the compare value, a global timestamp injection will occur to indicate the time that the lane s source clock turned off. 0: Means the corresponding bit must be logic 0 to match. 1: Means the corresponding bit must be logic 1 to match.

13.11.20 LNSTORECTRL_5: Lane Storage Control Register 5

CSR Register Name: LNSTORECTRL_5: Lane Storage Control Register 5				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 6114	**Offset End:** 6117
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:9	RO	0	Reserved	LNSTORECTRL Reserved
8	RW/P	0	LaneEn	**Lane Enable.** This bit will enable this lane for arbiter control. 0: Disable. This lane will not be observed by the arbitration logic and no data from this lane will be stored by the ODLA controller 1: Enable. This lane will be arbitrated with other enabled lanes and its data consumed by the ODLA controller.

<table>
<tr><td colspan="6"><b>CSR Register Name: LNSTORECTRL_5</b>: Lane Storage Control Register 5</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 6114</td><td><b>Offset End:</b> 6117</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>7:0</td><td>RW/P</td><td>1F</td><td>ClkOffComp</td><td colspan="2"><b>Clock Off Compare.</b> This bit field contains the 8-bit match value used by the clock off detection compare logic to set the maximum number of ODLA clocks since the last lane valid. When the clock off counter matches the compare value, a global timestamp injection will occur to indicate the time that the lane s source clock turned off.
0: Means the corresponding bit must be logic 0 to match.
1: Means the corresponding bit must be logic 1 to match.</td></tr>
</table>

13.11.21 LNSTORECTRL_6: Lane Storage Control Register 6

<table>
<tr><td colspan="6"><b>CSR Register Name: LNSTORECTRL_6</b>: Lane Storage Control Register 6</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 6118</td><td><b>Offset End:</b> 611B</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:9</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">LNSTORECTRL Reserved</td></tr>
<tr><td>8</td><td>RW/P</td><td>0</td><td>LaneEn</td><td colspan="2"><b>Lane Enable.</b> This bit will enable this lane for arbiter control.
0: Disable. This lane will not be observed by the arbitration logic and no data from this lane will be stored by the ODLA controller
1: Enable. This lane will be arbitrated with other enabled lanes and its data consumed by the ODLA controller.</td></tr>
</table>

CSR Register Name: LNSTORECTRL_6: Lane Storage Control Register 6				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 6118 **Offset End:** 611B
Bits	**Access**	**Default**	**Label**	**Bit Description**
7:0	RW/P	1F	ClkOffComp	**Clock Off Compare.** This bit field contains the 8-bit match value used by the clock off detection compare logic to set the maximum number of ODLA clocks since the last lane valid. When the clock off counter matches the compare value, a global timestamp injection will occur to indicate the time that the lane s source clock turned off. 0: Means the corresponding bit must be logic 0 to match. 1: Means the corresponding bit must be logic 1 to match.

13.11.22 LNSTORECTRL_7: Lane Storage Control Register 7

CSR Register Name: LNSTORECTRL_7: Lane Storage Control Register 7				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 611C **Offset End:** 611F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:9	RO	0	Reserved	LNSTORECTRL Reserved
8	RW/P	0	LaneEn	**Lane Enable.** This bit will enable this lane for arbiter control. 0: Disable. This lane will not be observed by the arbitration logic and no data from this lane will be stored by the ODLA controller 1: Enable. This lane will be arbitrated with other enabled lanes and its data consumed by the ODLA controller.

CSR Register Name: LNSTORECTRL_7: Lane Storage Control Register 7			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 611C	**Offset End:** 611F

Bits	Access	Default	Label	Bit Description
7:0	RW/P	1F	ClkOffComp	**Clock Off Compare.** This bit field contains the 8-bit match value used by the clock off detection compare logic to set the maximum number of ODLA clocks since the last lane valid. When the clock off counter matches the compare value, a global timestamp injection will occur to indicate the time that the lane s source clock turned off. 0: Means the corresponding bit must be logic 0 to match. 1: Means the corresponding bit must be logic 1 to match.

13.11.23 LNFTLCTL_0: Lane Filter Control Register 0

CSR Register Name: LNFTLCTL_0: Lane Filter Control Register 0			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 6180	**Offset End:** 6183

Bits	Access	Default	Label	Bit Description
31:25	RO	0	Reserved	LNSTORECTRL Reserved
24	RW/P	0	LaneFltInv	**Lane Filter Match Invert.** This bit inverts the resulting match equation. The packet is stored based on a logic 1 from the match equation including 0: Do not invert the resultant match equation. 1: Invert the resultant match equation.
23:16	RW/P	0	LaneFltForce	**Lane Filter Force Value.** This bit field contains the 8-bit force value used by the lane filter logic to force the lane filter bits to zero. Per Bit Force Definition: 0: Means the corresponding bit is passed as-is through to the match/mask filter logic 1: Means the corresponding bit is forced to 0 prior to the match/mask filter logic

<table>
<tr><td colspan="6"><b>CSR Register Name: LNFTLCTL_0</b>: Lane Filter Control Register 0</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 6180</td><td><b>Offset End:</b> 6183</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>15:8</td><td>RW/P</td><td>0</td><td>LaneFltMask</td><td colspan="2"><b>Lane Filter Mask Value.</b> This bit field contains the 8-bit mask value used by the lane filter logic to mask the LaneFltMatch value. Each bit in the mask field corresponds to a match bit in the LaneFltMatch. Per Bit Mask Definition: 0: Means the corresponding bit is not included in the match equation. 1: Means the corresponding bit is included in the match equation. Corner case conditions that affect the match equation output: 0x00 = This byte lane ignores the match value and all debug data is sent to the next stage. The filter logic is disabled. The LaneFltInv must be cleared to zero otherwise it is a programming error. 0xFF = This byte lane must match exactly to the match value to be of use.</td></tr>
<tr><td>7:0</td><td>RW/P</td><td>0</td><td>LaneFltMatch</td><td colspan="2"><b>Lane Filter Match Value.</b> This bit field contains the 8-bit match value used by the lane filter logic to develop an overall match value for all the lanes. Each bit in the match field corresponds to a match bit in the LaneFltMatch. 0: Means the corresponding bit must be logic 0 to match. 1: Means the corresponding bit must be logic 1 to match.</td></tr>
</table>

13.11.24 LNFTLCTL_1: Lane Filter Control Register 1

<table>
<tr><td colspan="6"><b>CSR Register Name: LNFTLCTL_1</b>: Lane Filter Control Register 1</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 6184</td><td><b>Offset End:</b> 6187</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:25</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">LNSTORECTRL Reserved</td></tr>
</table>

<table>
<tr><td colspan="6"><b>CSR Register Name: LNFTLCTL_1</b>: Lane Filter Control Register 1</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 6184</td><td><b>Offset End:</b> 6187</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>24</td><td>RW/P</td><td>0</td><td>LaneFltInv</td><td colspan="2"><b>Lane Filter Match Invert.</b> This bit inverts the resulting match equation. The packet is stored based on a logic 1 from the match equation including
0: Do not invert the resultant match equation.
1: Invert the resultant match equation.</td></tr>
<tr><td>23:16</td><td>RW/P</td><td>0</td><td>LaneFltForce</td><td colspan="2"><b>Lane Filter Force Value.</b> This bit field contains the 8-bit force value used by the lane filter logic to force the lane filter bits to zero. Per Bit Force Definition:
0: Means the corresponding bit is passed as-is through to the match/mask filter logic
1: Means the corresponding bit is forced to 0 prior to the match/mask filter logic</td></tr>
<tr><td>15:8</td><td>RW/P</td><td>0</td><td>LaneFltMask</td><td colspan="2"><b>Lane Filter Mask Value.</b> This bit field contains the 8-bit mask value used by the lane filter logic to mask the LaneFltMatch value. Each bit in the mask field corresponds to a match bit in the LaneFltMatch. Per Bit Mask Definition: 0: Means the corresponding bit is not included in the match equation. 1: Means the corresponding bit is included in the match equation. Corner case conditions that affect the match equation output: 0x00 = This byte lane ignores the match value and all debug data is sent to the next stage. The filter logic is disabled. The LaneFltInv must be cleared to zero otherwise it is a programming error. 0xFF = This byte lane must match exactly to the match value to be of use.</td></tr>
<tr><td>7:0</td><td>RW/P</td><td>0</td><td>LaneFltMatch</td><td colspan="2"><b>Lane Filter Match Value.</b> This bit field contains the 8-bit match value used by the lane filter logic to develop an overall match value for all the lanes. Each bit in the match field corresponds to a match bit in the LaneFltMatch. 0: Means the corresponding bit must be logic 0 to match. 1: Means the corresponding bit must be logic 1 to match.</td></tr>
</table>

13.11.25 LNFTLCTL_2: Lane Filter Control Register 2

CSR Register Name: LNFTLCTL_2: Lane Filter Control Register 2				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 6188	**Offset End:** 618B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:25	RO	0	Reserved	LNSTORECTRL Reserved
24	RW/P	0	LaneFltInv	**Lane Filter Match Invert.** This bit inverts the resulting match equation. The packet is stored based on a logic 1 from the match equation including 0: Do not invert the resultant match equation. 1: Invert the resultant match equation.
23:16	RW/P	0	LaneFltForce	**Lane Filter Force Value.** This bit field contains the 8-bit force value used by the lane filter logic to force the lane filter bits to zero. Per Bit Force Definition: 0: Means the corresponding bit is passed as-is through to the match/mask filter logic 1: Means the corresponding bit is forced to 0 prior to the match/mask filter logic
15:8	RW/P	0	LaneFltMask	**Lane Filter Mask Value.** This bit field contains the 8-bit mask value used by the lane filter logic to mask the LaneFltMatch value. Each bit in the mask field corresponds to a match bit in the LaneFltMatch. Per Bit Mask Definition: 0: Means the corresponding bit is not included in the match equation. 1: Means the corresponding bit is included in the match equation. Corner case conditions that affect the match equation output: 0x00 = This byte lane ignores the match value and all debug data is sent to the next stage. The filter logic is disabled. The LaneFltInv must be cleared to zero otherwise it is a programming error. 0xFF = This byte lane must match exactly to the match value to be of use.

CSR Register Name: **LNFTLCTL_2**: Lane Filter Control Register 2				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 6188	**Offset End:** 618B
Bits	**Access**	**Default**	**Label**	**Bit Description**
7:0	RW/P	0	LaneFltMatch	**Lane Filter Match Value.** This bit field contains the 8-bit match value used by the lane filter logic to develop an overall match value for all the lanes. Each bit in the match field corresponds to a match bit in the LaneFltMatch. 0: Means the corresponding bit must be logic 0 to match. 1: Means the corresponding bit must be logic 1 to match.

13.11.26 LNFTLCTL_3: Lane Filter Control Register 3

CSR Register Name: **LNFTLCTL_3**: Lane Filter Control Register 3				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 618C	**Offset End:** 618F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:25	RO	0	Reserved	LNSTORECTRL Reserved
24	RW/P	0	LaneFltInv	**Lane Filter Match Invert.** This bit inverts the resulting match equation. The packet is stored based on a logic 1 from the match equation including 0: Do not invert the resultant match equation. 1: Invert the resultant match equation.
23:16	RW/P	0	LaneFltForce	**Lane Filter Force Value.** This bit field contains the 8-bit force value used by the lane filter logic to force the lane filter bits to zero. Per Bit Force Definition: 0: Means the corresponding bit is passed as-is through to the match/mask filter logic 1: Means the corresponding bit is forced to 0 prior to the match/mask filter logic

<table>
<tr><td colspan="6"><b>CSR Register Name: LNFTLCTL_3</b>: Lane Filter Control Register 3</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 618C</td><td><b>Offset End:</b> 618F</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>15:8</td><td>RW/P</td><td>0</td><td>LaneFltMask</td><td colspan="2"><b>Lane Filter Mask Value.</b> This bit field contains the 8-bit mask value used by the lane filter logic to mask the LaneFltMatch value. Each bit in the mask field corresponds to a match bit in the LaneFltMatch. Per Bit Mask Definition: 0: Means the corresponding bit is not included in the match equation. 1: Means the corresponding bit is included in the match equation. Corner case conditions that affect the match equation output: 0x00 = This byte lane ignores the match value and all debug data is sent to the next stage. The filter logic is disabled. The LaneFltInv must be cleared to zero otherwise it is a programming error. 0xFF = This byte lane must match exactly to the match value to be of use.</td></tr>
<tr><td>7:0</td><td>RW/P</td><td>0</td><td>LaneFltMatch</td><td colspan="2"><b>Lane Filter Match Value.</b> This bit field contains the 8-bit match value used by the lane filter logic to develop an overall match value for all the lanes. Each bit in the match field corresponds to a match bit in the LaneFltMatch. 0: Means the corresponding bit must be logic 0 to match. 1: Means the corresponding bit must be logic 1 to match.</td></tr>
</table>

13.11.27 LNFTLCTL_4: Lane Filter Control Register 4

<table>
<tr><td colspan="6"><b>CSR Register Name: LNFTLCTL_4</b>: Lane Filter Control Register 4</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 6190</td><td><b>Offset End:</b> 6193</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:25</td><td>RO</td><td>0</td><td>Reserved</td><td colspan="2">LNSTORECTRL Reserved</td></tr>
</table>

<table>
<tr><td colspan="5"><b>CSR Register Name: LNFTLCTL_4</b>: Lane Filter Control Register 4</td></tr>
<tr><td><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 6190</td><td><b>Offset End:</b> 6193</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th>Bit Description</th></tr>
<tr><td>24</td><td>RW/P</td><td>0</td><td>LaneFltInv</td><td><b>Lane Filter Match Invert.</b> This bit inverts the resulting match equation. The packet is stored based on a logic 1 from the match equation including
0: Do not invert the resultant match equation.
1: Invert the resultant match equation.</td></tr>
<tr><td>23:16</td><td>RW/P</td><td>0</td><td>LaneFltForce</td><td><b>Lane Filter Force Value.</b> This bit field contains the 8-bit force value used by the lane filter logic to force the lane filter bits to zero. Per Bit Force Definition:
0: Means the corresponding bit is passed as-is through to the match/mask filter logic
1: Means the corresponding bit is forced to 0 prior to the match/mask filter logic</td></tr>
<tr><td>15:8</td><td>RW/P</td><td>0</td><td>LaneFltMask</td><td><b>Lane Filter Mask Value.</b> This bit field contains the 8-bit mask value used by the lane filter logic to mask the LaneFltMatch value. Each bit in the mask field corresponds to a match bit in the LaneFltMatch. Per Bit Mask Definition: 0: Means the corresponding bit is not included in the match equation. 1: Means the corresponding bit is included in the match equation. Corner case conditions that affect the match equation output: 0x00 = This byte lane ignores the match value and all debug data is sent to the next stage. The filter logic is disabled. The LaneFltInv must be cleared to zero otherwise it is a programming error. 0xFF = This byte lane must match exactly to the match value to be of use.</td></tr>
<tr><td>7:0</td><td>RW/P</td><td>0</td><td>LaneFltMatch</td><td><b>Lane Filter Match Value.</b> This bit field contains the 8-bit match value used by the lane filter logic to develop an overall match value for all the lanes. Each bit in the match field corresponds to a match bit in the LaneFltMatch. 0: Means the corresponding bit must be logic 0 to match. 1: Means the corresponding bit must be logic 1 to match.</td></tr>
</table>

13.11.28 LNFTLCTL_5: Lane Filter Control Register 5

CSR Register Name: LNFTLCTL_5: Lane Filter Control Register 5				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 6194	**Offset End:** 6197
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:25	RO	0	Reserved	LNSTORECTRL Reserved
24	RW/P	0	LaneFltInv	**Lane Filter Match Invert.** This bit inverts the resulting match equation. The packet is stored based on a logic 1 from the match equation including 0: Do not invert the resultant match equation. 1: Invert the resultant match equation.
23:16	RW/P	0	LaneFltForce	**Lane Filter Force Value.** This bit field contains the 8-bit force value used by the lane filter logic to force the lane filter bits to zero. Per Bit Force Definition: 0: Means the corresponding bit is passed as-is through to the match/mask filter logic 1: Means the corresponding bit is forced to 0 prior to the match/mask filter logic
15:8	RW/P	0	LaneFltMask	**Lane Filter Mask Value.** This bit field contains the 8-bit mask value used by the lane filter logic to mask the LaneFltMatch value. Each bit in the mask field corresponds to a match bit in the LaneFltMatch. Per Bit Mask Definition: 0: Means the corresponding bit is not included in the match equation. 1: Means the corresponding bit is included in the match equation. Corner case conditions that affect the match equation output: 0x00 = This byte lane ignores the match value and all debug data is sent to the next stage. The filter logic is disabled. The LaneFltInv must be cleared to zero otherwise it is a programming error. 0xFF = This byte lane must match exactly to the match value to be of use.

CSR Register Name: LNFTLCTL_5: Lane Filter Control Register 5				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 6194	**Offset End:** 6197
Bits	**Access**	**Default**	**Label**	**Bit Description**
7:0	RW/P	0	LaneFltMatch	**Lane Filter Match Value.** This bit field contains the 8-bit match value used by the lane filter logic to develop an overall match value for all the lanes. Each bit in the match field corresponds to a match bit in the LaneFltMatch. 0: Means the corresponding bit must be logic 0 to match. 1: Means the corresponding bit must be logic 1 to match.

13.11.29 LNFTLCTL_6: Lane Filter Control Register 6

CSR Register Name: LNFTLCTL_6: Lane Filter Control Register 6				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 6198	**Offset End:** 619B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:25	RO	0	Reserved	LNSTORECTRL Reserved
24	RW/P	0	LaneFltInv	**Lane Filter Match Invert.** This bit inverts the resulting match equation. The packet is stored based on a logic 1 from the match equation including 0: Do not invert the resultant match equation. 1: Invert the resultant match equation.
23:16	RW/P	0	LaneFltForce	**Lane Filter Force Value.** This bit field contains the 8-bit force value used by the lane filter logic to force the lane filter bits to zero. Per Bit Force Definition: 0: Means the corresponding bit is passed as-is through to the match/mask filter logic 1: Means the corresponding bit is forced to 0 prior to the match/mask filter logic

CSR Register Name: LNFTLCTL_6: Lane Filter Control Register 6				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 6198 **Offset End:** 619B
Bits	**Access**	**Default**	**Label**	**Bit Description**
15:8	RW/P	0	LaneFltMask	**Lane Filter Mask Value.** This bit field contains the 8-bit mask value used by the lane filter logic to mask the LaneFltMatch value. Each bit in the mask field corresponds to a match bit in the LaneFltMatch. Per Bit Mask Definition: 0: Means the corresponding bit is not included in the match equation. 1: Means the corresponding bit is included in the match equation. Corner case conditions that affect the match equation output: 0x00 = This byte lane ignores the match value and all debug data is sent to the next stage. The filter logic is disabled. The LaneFltInv must be cleared to zero otherwise it is a programming error. 0xFF = This byte lane must match exactly to the match value to be of use.
7:0	RW/P	0	LaneFltMatch	**Lane Filter Match Value.** This bit field contains the 8-bit match value used by the lane filter logic to develop an overall match value for all the lanes. Each bit in the match field corresponds to a match bit in the LaneFltMatch. 0: Means the corresponding bit must be logic 0 to match. 1: Means the corresponding bit must be logic 1 to match.

13.11.30 LNFTLCTL_7: Lane Filter Control Register 7

CSR Register Name: LNFTLCTL_7: Lane Filter Control Register 7				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 619C **Offset End:** 619F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:25	RO	0	Reserved	LNSTORECTRL Reserved

CSR Register Name: LNFTLCTL_7: Lane Filter Control Register 7				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 619C **Offset End:** 619F
Bits	**Access**	**Default**	**Label**	**Bit Description**
24	RW/P	0	LaneFltInv	**Lane Filter Match Invert.** This bit inverts the resulting match equation. The packet is stored based on a logic 1 from the match equation including 0: Do not invert the resultant match equation. 1: Invert the resultant match equation.
23:16	RW/P	0	LaneFltForce	**Lane Filter Force Value.** This bit field contains the 8-bit force value used by the lane filter logic to force the lane filter bits to zero. Per Bit Force Definition: 0: Means the corresponding bit is passed as-is through to the match/mask filter logic 1: Means the corresponding bit is forced to 0 prior to the match/mask filter logic
15:8	RW/P	0	LaneFltMask	**Lane Filter Mask Value.** This bit field contains the 8-bit mask value used by the lane filter logic to mask the LaneFltMatch value. Each bit in the mask field corresponds to a match bit in the LaneFltMatch. Per Bit Mask Definition: 0: Means the corresponding bit is not included in the match equation. 1: Means the corresponding bit is included in the match equation. Corner case conditions that affect the match equation output: 0x00 = This byte lane ignores the match value and all debug data is sent to the next stage. The filter logic is disabled. The LaneFltInv must be cleared to zero otherwise it is a programming error. 0xFF = This byte lane must match exactly to the match value to be of use.
7:0	RW/P	0	LaneFltMatch	**Lane Filter Match Value.** This bit field contains the 8-bit match value used by the lane filter logic to develop an overall match value for all the lanes. Each bit in the match field corresponds to a match bit in the LaneFltMatch. 0: Means the corresponding bit must be logic 0 to match. 1: Means the corresponding bit must be logic 1 to match.

13.12 TAP Registers

13.12.1 TAP Registers Summary

TAP Registers			
Offset Start	**Offset End**	**Symbol**	**Register Name/Function**
C	C	SLVIDCODE	Slave TAP Identification Register
40	40	REQMSG	Request Message Register
41	41	RSPMSG	Response Message Register
42	42	CLKOVR	Clock Override Register
43	43	RSTOVR	Reset Override Register

13.12.2 SLVIDCODE: Slave TAP Identification Register

CSR Register Name: **SLVIDCODE**: Slave TAP Identification Register				
Bar: n/a		**Reset:** trst_b		**Offset Start:** C **Offset End:** C
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RO	strap	SLVIDCODE	Slave ID Code

13.12.3 REQMSG: Request Message Register

CSR Register Name: **REQMSG**: Request Message Register				
Bar: n/a		**Reset:** trst_b		**Offset Start:** 40 **Offset End:** 40
Bits	**Access**	**Default**	**Label**	**Bit Description**
113:112	WO	0	BAR	Base Address Register number (0, 1, 2, or 3) for register read/write requests
111:104	WO	0	BE	Request data byte enables
103:40	WO	0	REQDATA	**Request Data (write data).** If accessing a 32-bit CSR, only the least significant 32 bits will be written

<table>
<tr><td colspan="6"><b>CSR Register Name: REQMSG</b>: Request Message Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> n/a</td><td colspan="2"><b>Reset:</b> trst_b</td><td><b>Offset Start:</b> 40</td><td><b>Offset End:</b> 40</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>28</td><td>WO</td><td>0</td><td>REQTYPE</td><td colspan="2"><b>Request type:</b>
0001: Memory (Register) Read
0010: Memory (Register) Write
0100: PCI Configuration Space Read
1000: PCI Configuration Space Write</td></tr>
<tr><td>17:4</td><td>WO</td><td>0</td><td>ADDR</td><td colspan="2"><b>Request address:</b> the offset within the BAR</td></tr>
<tr><td>3:0</td><td>WO</td><td>0</td><td>RSVD</td><td colspan="2">Reserved for future use.</td></tr>
</table>

13.12.4 RSPMSG: Response Message Register

<table>
<tr><td colspan="6"><b>CSR Register Name: RSPMSG</b>: Response Message Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> n/a</td><td colspan="2"><b>Reset:</b> trst_b</td><td><b>Offset Start:</b> 41</td><td><b>Offset End:</b> 41</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>65:2</td><td>RO</td><td>0</td><td>RSPDATA</td><td colspan="2"><b>Response Data (read data) which is only valid if STAT is equal to 10b.</b> If reading a 32-bit CSR, the most significant 32 bits of this field have to be ignored</td></tr>
<tr><td>65:2</td><td>RO</td><td>0</td><td>RSPDATA</td><td colspan="2"><b>Response Data (read data) which is only valid if STAT is equal to 10b.</b> If reading a 32-bit CSR, the most significant 32 bits of this field have to be ignored</td></tr>
</table>

13.12.5 CLKOVR: Clock Override Register

<table>
<tr><td colspan="6"><b>CSR Register Name: CLKOVR</b>: Clock Override Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> n/a</td><td colspan="2"><b>Reset:</b> trst_b</td><td><b>Offset Start:</b> 42</td><td><b>Offset End:</b> 42</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>3:0</td><td>RW</td><td>0</td><td>CLKSEL</td><td colspan="2">Clock select to select the clock used to initiate requests and return responses with the following encoding:
0000: IOSF-Primary clock (IPCLK)
0101: TAP Clock (TCK)
All others are reserved for future use</td></tr>
</table>

13.12.6 RSTOVR: Reset Override Register

CSR Register Name: **RSTOVR**: Reset Override Register				
Bar: n/a		**Reset:** trst_b		**Offset Start:** 43 / **Offset End:** 43
Bits	**Access**	**Default**	**Label**	**Bit Description**
5	RW	0	HSTRST	If cleared, the host_rst_b will be asserted until the bit is set again.

13.13 VIS Event Recognizer Registers

13.13.1 VIS Event Recognizer Registers Summary

VIS Event Recognizer Registers			
Offset Start	**Offset End**	**Symbol**	**Register Name/Function**
07000	07003	VERCAP	Event Recognizer Capabilities Register
07004	07007	VERCTL	Event Recognizer Control Register
07008	0700B	VERSTAT	Event Recognizer Status Register
07100	07103	EVRCTL0	Event Recognizer Control Register 0
07104	07107	EVRCTL1	Event Recognizer Control Register 1
07108	0710B	EVRCTL2	Event Recognizer Control Register 2
0710C	0710F	EVRCTL3	Event Recognizer Control Register 3
07110	07113	EVRCTL4	Event Recognizer Control Register 4
07114	07117	EVRCTL5	Event Recognizer Control Register 5
07118	0711B	EVRCTL6	Event Recognizer Control Register 6
0711C	0711F	EVRCTL7	Event Recognizer Control Register 7
07140	07143	EVRMCH0	Event Recognizer Match Register 0

VIS Event Recognizer Registers			
Offset Start	Offset End	Symbol	Register Name / Function
07144	07147	EVRMCH1	Event Recognizer Match Register 1
07148	0714B	EVRMCH2	Event Recognizer Match Register 2
0714C	0714F	EVRMCH3	Event Recognizer Match Register 3
07150	07153	EVRMCH4	Event Recognizer Match Register 4
07154	07157	EVRMCH5	Event Recognizer Match Register 5
07158	0715B	EVRMCH6	Event Recognizer Match Register 6
0715C	0715F	EVRMCH7	Event Recognizer Match Register 7
07180	07183	EVRMSK0	Event Recognizer Mask Register 0
07184	07187	EVRMSK1	Event Recognizer Mask Register 1
07188	0718B	EVRMSK2	Event Recognizer Mask Register 2
0718C	0718F	EVRMSK3	Event Recognizer Mask Register 3
07190	07193	EVRMSK4	Event Recognizer Mask Register 4
07194	07197	EVRMSK5	Event Recognizer Mask Register 5
07198	0719B	EVRMSK6	Event Recognizer Mask Register 6
0719C	0719F	EVRMSK7	Event Recognizer Mask Register 7
071C0	071C3	EVRFBMCH0	Event Recognizer Feedback Match Register 0
071C4	071C7	EVRFBMCH1	Event Recognizer Feedback Match Register 1
071C8	071CB	EVRFBMCH2	Event Recognizer Feedback Match Register 2
071CC	071CF	EVRFBMCH3	Event Recognizer Feedback Match Register 3

VIS Event Recognizer Registers

Offset Start	Offset End	Symbol	Register Name/Function
071D0	071D3	EVRFBMCH4	Event Recognizer Feedback Match Register 4
071D4	071D7	EVRFBMCH5	Event Recognizer Feedback Match Register 5
071D8	071DB	EVRFBMCH6	Event Recognizer Feedback Match Register 6
071DC	071DF	EVRFBMCH7	Event Recognizer Feedback Match Register 7
07200	07203	EVRFBMSK0	Event Recognizer Feedback Mask Register 0
07204	07207	EVRFBMSK1	Event Recognizer Feedback Mask Register 1
07208	0720B	EVRFBMSK2	Event Recognizer Feedback Mask Register 2
0720C	0720F	EVRFBMSK3	Event Recognizer Feedback Mask Register 3
07210	07213	EVRFBMSK4	Event Recognizer Feedback Mask Register 4
07214	07217	EVRFBMSK5	Event Recognizer Feedback Mask Register 5
07218	0721B	EVRFBMSK6	Event Recognizer Feedback Mask Register 6
0721C	0721F	EVRFBMSK7	Event Recognizer Feedback Mask Register 7

13.13.2 VERCAP: Event Recognizer Capabilities Register

<table>
<tr><td colspan="6"><b>CSR Register Name: VERCAP</b>: Event Recognizer Capabilities Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b>
07000</td><td><b>Offset End:</b>
07003</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:16</td><td>RO</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>15:8</td><td>RO</td><td>10</td><td></td><td colspan="2"></td></tr>
<tr><td>7:5</td><td>RO</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>4:0</td><td>RO</td><td>8</td><td></td><td colspan="2"></td></tr>
</table>

13.13.3 VERCTL: Event Recognizer Control Register

<table>
<tr><td colspan="6"><b>CSR Register Name: VERCTL</b>: Event Recognizer Control Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b>
07004</td><td><b>Offset End:</b>
07007</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:2</td><td>RO</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>1</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>0</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
</table>

13.13.4 VERSTAT: Event Recognizer Status Register

<table>
<tr><td colspan="6"><b>CSR Register Name: VERSTAT</b>: Event Recognizer Status Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b>
07008</td><td><b>Offset End:</b>
0700B</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:16</td><td>RO</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>15:0</td><td>RW/1C/P</td><td>0</td><td></td><td colspan="2"></td></tr>
</table>

13.13.5 EVRCTL0: Event Recognizer Control Register 0

CSR Register Name: **EVRCTL0**: Event Recognizer Control Register 0					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 07100	**Offset End:** 07103
Bits	**Access**	**Default**	**Label**	**Bit Description**	
31:9	RO	0			
8	RW/P	0			
7:3	RW/P	0			
2	RW/P	0			
1	RW/P	0			
0	RW/P	0			

13.13.6 EVRCTL1: Event Recognizer Control Register 1

CSR Register Name: **EVRCTL1**: Event Recognizer Control Register 1					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 07104	**Offset End:** 07107
Bits	**Access**	**Default**	**Label**	**Bit Description**	
31:9	RO	0			
8	RW/P	0			
7:3	RW/P	0			
2	RW/P	0			
1	RW/P	0			
0	RW/P	0			

13.13.7 EVRCTL2: Event Recognizer Control Register 2

CSR Register Name: **EVRCTL2**: Event Recognizer Control Register 2					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 07108	**Offset End:** 0710B
Bits	**Access**	**Default**	**Label**	**Bit Description**	
31:9	RO	0			
8	RW/P	0			
7:3	RW/P	0			

<table>
<tr><td colspan="6"><b>CSR Register Name: EVRCTL2</b>: Event Recognizer Control Register 2</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 07108</td><td><b>Offset End:</b> 0710B</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>2</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>1</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>0</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
</table>

13.13.8 EVRCTL3: Event Recognizer Control Register 3

<table>
<tr><td colspan="6"><b>CSR Register Name: EVRCTL3</b>: Event Recognizer Control Register 3</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 0710C</td><td><b>Offset End:</b> 0710F</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:9</td><td>RO</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>8</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>7:3</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>2</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>1</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>0</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
</table>

13.13.9 EVRCTL4: Event Recognizer Control Register 4

<table>
<tr><td colspan="6"><b>CSR Register Name: EVRCTL4</b>: Event Recognizer Control Register 4</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 07110</td><td><b>Offset End:</b> 07113</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:9</td><td>RO</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>8</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>7:3</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>2</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>1</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>0</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
</table>

13.13.10 EVRCTL5: Event Recognizer Control Register 5

<table>
<tr><td colspan="6"><b>CSR Register Name:</b> <b>EVRCTL5</b>: Event Recognizer Control Register 5</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 07114</td><td><b>Offset End:</b> 07117</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:9</td><td>RO</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>8</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>7:3</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>2</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>1</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>0</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
</table>

13.13.11 EVRCTL6: Event Recognizer Control Register 6

<table>
<tr><td colspan="6"><b>CSR Register Name:</b> <b>EVRCTL6</b>: Event Recognizer Control Register 6</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 07118</td><td><b>Offset End:</b> 0711B</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:9</td><td>RO</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>8</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>7:3</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>2</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>1</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>0</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
</table>

13.13.12 EVRCTL7: Event Recognizer Control Register 7

<table>
<tr><td colspan="6"><b>CSR Register Name:</b> <b>EVRCTL7</b>: Event Recognizer Control Register 7</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 0711C</td><td><b>Offset End:</b> 0711F</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:9</td><td>RO</td><td>0</td><td></td><td colspan="2"></td></tr>
</table>

CSR Register Name: **EVRCTL7**: Event Recognizer Control Register 7			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 0711C	**Offset End:** 0711F

Bits	Access	Default	Label	Bit Description
8	RW/P	0		
7:3	RW/P	0		
2	RW/P	0		
1	RW/P	0		
0	RW/P	0		

13.13.13 EVRMCH0: Event Recognizer Match Register 0

CSR Register Name: **EVRMCH0**: Event Recognizer Match Register 0			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 07140	**Offset End:** 07143

Bits	Access	Default	Label	Bit Description
31:16	RO	0		
15:0	RW/P	0		

13.13.14 EVRMCH1: Event Recognizer Match Register 1

CSR Register Name: **EVRMCH1**: Event Recognizer Match Register 1			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 07144	**Offset End:** 07147

Bits	Access	Default	Label	Bit Description
31:16	RO	0		
15:0	RW/P	0		

13.13.15 EVRMCH2: Event Recognizer Match Register 2

CSR Register Name: **EVRMCH2**: Event Recognizer Match Register 2			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 07148	**Offset End:** 0714B

Bits	Access	Default	Label	Bit Description
31:16	RO	0		
15:0	RW/P	0		

13.13.16 EVRMCH3: Event Recognizer Match Register 3

<table>
<tr><td colspan="6"><b>CSR Register Name: EVRMCH3</b>: Event Recognizer Match Register 3</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 0714C</td><td><b>Offset End:</b> 0714F</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:16</td><td>RO</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>15:0</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
</table>

13.13.17 EVRMCH4: Event Recognizer Match Register 4

<table>
<tr><td colspan="6"><b>CSR Register Name: EVRMCH4</b>: Event Recognizer Match Register 4</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 07150</td><td><b>Offset End:</b> 07153</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:16</td><td>RO</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>15:0</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
</table>

13.13.18 EVRMCH5: Event Recognizer Match Register 5

<table>
<tr><td colspan="6"><b>CSR Register Name: EVRMCH5</b>: Event Recognizer Match Register 5</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b> 07154</td><td><b>Offset End:</b> 07157</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:16</td><td>RO</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>15:0</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
</table>

13.13.19 EVRMCH6: Event Recognizer Match Register 6

CSR Register Name: EVRMCH6: Event Recognizer Match Register 6			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 07158	**Offset End:** 0715B

Bits	Access	Default	Label	Bit Description
31:16	RO	0		
15:0	RW/P	0		

13.13.20 EVRMCH7: Event Recognizer Match Register 7

CSR Register Name: EVRMCH7: Event Recognizer Match Register 7			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 0715C	**Offset End:** 0715F

Bits	Access	Default	Label	Bit Description
31:16	RO	0		
15:0	RW/P	0		

13.13.21 EVRMSK0: Event Recognizer Mask Register 0

CSR Register Name: EVRMSK0: Event Recognizer Mask Register 0			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 07180	**Offset End:** 07183

Bits	Access	Default	Label	Bit Description
31:16	RO	0		
15:0	RW/P	0		

13.13.22 EVRMSK1: Event Recognizer Mask Register 1

CSR Register Name: EVRMSK1: Event Recognizer Mask Register 1			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 07184	**Offset End:** 07187

Bits	Access	Default	Label	Bit Description
31:16	RO	0		

CSR Register Name: **EVRMSK1**: Event Recognizer Mask Register 1				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 07184 **Offset End:** 07187
Bits	**Access**	**Default**	**Label**	**Bit Description**
15:0	RW/P	0		

13.13.23 EVRMSK2: Event Recognizer Mask Register 2

CSR Register Name: **EVRMSK2**: Event Recognizer Mask Register 2				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 07188 **Offset End:** 0718B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:16	RO	0		
15:0	RW/P	0		

13.13.24 EVRMSK3: Event Recognizer Mask Register 3

CSR Register Name: **EVRMSK3**: Event Recognizer Mask Register 3				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 0718C **Offset End:** 0718F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:16	RO	0		
15:0	RW/P	0		

13.13.25 EVRMSK4: Event Recognizer Mask Register 4

CSR Register Name: **EVRMSK4**: Event Recognizer Mask Register 4				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 07190 **Offset End:** 07193
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:16	RO	0		
15:0	RW/P	0		

13.13.26 EVRMSK5: Event Recognizer Mask Register 5

CSR Register Name: EVRMSK5: Event Recognizer Mask Register 5				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 07194	**Offset End:** 07197
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:16	RO	0		
15:0	RW/P	0		

13.13.27 EVRMSK6: Event Recognizer Mask Register 6

CSR Register Name: EVRMSK6: Event Recognizer Mask Register 6				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 07198	**Offset End:** 0719B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:16	RO	0		
15:0	RW/P	0		

13.13.28 EVRMSK7: Event Recognizer Mask Register 7

CSR Register Name: EVRMSK7: Event Recognizer Mask Register 7				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 0719C	**Offset End:** 0719F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:16	RO	0		
15:0	RW/P	0		

13.13.29 EVRFBMCH0: Event Recognizer Feedback Match Register 0

CSR Register Name: EVRFBMCH0: Event Recognizer Feedback Match Register 0			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 071C0	**Offset End:** 071C3

Bits	Access	Default	Label	Bit Description
31:8	RO	0		
7:0	RW/P	0		

13.13.30 EVRFBMCH1: Event Recognizer Feedback Match Register 1

<table>
<tr><td colspan="6"><b>CSR Register Name: EVRFBMCH1</b>: Event Recognizer Feedback Match Register 1</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b>
071C4</td><td><b>Offset End:</b>
071C7</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:8</td><td>RO</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>7:0</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
</table>

13.13.31 EVRFBMCH2: Event Recognizer Feedback Match Register 2

<table>
<tr><td colspan="6"><b>CSR Register Name: EVRFBMCH2</b>: Event Recognizer Feedback Match Register 2</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b>
071C8</td><td><b>Offset End:</b>
071CB</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:8</td><td>RO</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>7:0</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
</table>

13.13.32 EVRFBMCH3: Event Recognizer Feedback Match Register 3

<table>
<tr><td colspan="6"><b>CSR Register Name: EVRFBMCH3</b>: Event Recognizer Feedback Match Register 3</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> npk_rst_b</td><td><b>Offset Start:</b>
071CC</td><td><b>Offset End:</b>
071CF</td></tr>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Label</th><th colspan="2">Bit Description</th></tr>
<tr><td>31:8</td><td>RO</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>7:0</td><td>RW/P</td><td>0</td><td></td><td colspan="2"></td></tr>
</table>

13.13.33 EVRFBMCH4: Event Recognizer Feedback Match Register 4

CSR Register Name: EVRFBMCH4: Event Recognizer Feedback Match Register 4					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 071D0	**Offset End:** 071D3
Bits	**Access**	**Default**	**Label**	**Bit Description**	
31:8	RO	0			
7:0	RW/P	0			

13.13.34 EVRFBMCH5: Event Recognizer Feedback Match Register 5

CSR Register Name: EVRFBMCH5: Event Recognizer Feedback Match Register 5					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 071D4	**Offset End:** 071D7
Bits	**Access**	**Default**	**Label**	**Bit Description**	
31:8	RO	0			
7:0	RW/P	0			

13.13.35 EVRFBMCH6: Event Recognizer Feedback Match Register 6

CSR Register Name: EVRFBMCH6: Event Recognizer Feedback Match Register 6					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 071D8	**Offset End:** 071DB
Bits	**Access**	**Default**	**Label**	**Bit Description**	
31:8	RO	0			
7:0	RW/P	0			

13.13.36 EVRFBMCH7: Event Recognizer Feedback Match Register 7

CSR Register Name: EVRFBMCH7: Event Recognizer Feedback Match Register 7					
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 071DC	**Offset End:** 071DF
Bits	**Access**	**Default**	**Label**	**Bit Description**	
31:8	RO	0			

CSR Register Name: EVRFBMCH7: Event Recognizer Feedback Match Register 7				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 071DC **Offset End:** 071DF
Bits	**Access**	**Default**	**Label**	**Bit Description**
7:0	RW/P	0		

13.13.37 EVRFBMSK0: Event Recognizer Feedback Mask Register 0

CSR Register Name: EVRFBMSK0: Event Recognizer Feedback Mask Register 0				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 07200 **Offset End:** 07203
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:8	RO	0		
7:0	RW/P	0		

13.13.38 EVRFBMSK1: Event Recognizer Feedback Mask Register 1

CSR Register Name: EVRFBMSK1: Event Recognizer Feedback Mask Register 1				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 07204 **Offset End:** 07207
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:8	RO	0		
7:0	RW/P	0		

13.13.39 EVRFBMSK2: Event Recognizer Feedback Mask Register 2

CSR Register Name: EVRFBMSK2: Event Recognizer Feedback Mask Register 2				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b		**Offset Start:** 07208 **Offset End:** 0720B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:8	RO	0		
7:0	RW/P	0		

13.13.40 EVRFBMSK3: Event Recognizer Feedback Mask Register 3

CSR Register Name: EVRFBMSK3: Event Recognizer Feedback Mask Register 3				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 0720C	**Offset End:** 0720F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:8	RO	0		
7:0	RW/P	0		

13.13.41 EVRFBMSK4: Event Recognizer Feedback Mask Register 4

CSR Register Name: EVRFBMSK4: Event Recognizer Feedback Mask Register 4				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 07210	**Offset End:** 07213
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:8	RO	0		
7:0	RW/P	0		

13.13.42 EVRFBMSK5: Event Recognizer Feedback Mask Register 5

CSR Register Name: EVRFBMSK5: Event Recognizer Feedback Mask Register 5				
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b		**Offset Start:** 07214	**Offset End:** 07217
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:8	RO	0		
7:0	RW/P	0		

13.13.43 EVRFBMSK6: Event Recognizer Feedback Mask Register 6

CSR Register Name: EVRFBMSK6: Event Recognizer Feedback Mask Register 6			
Bar: CSR_MTB_BAR	**Reset:** npk_rst_b	**Offset Start:** 07218	**Offset End:** 0721B

Bits	Access	Default	Label	Bit Description
31:8	RO	0		
7:0	RW/P	0		

13.13.44 EVRFBMSK7: Event Recognizer Feedback Mask Register 7

CSR Register Name: EVRFBMSK7: Event Recognizer Feedback Mask Register 7				
Bar: CSR_MTB_BAR		**Reset:** npk_rst_b	**Offset Start:** 0721C	**Offset End:** 0721F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:8	RO	0		
7:0	RW/P	0		

13.14 VIS Register Controller Registers

13.14.1 VIS Register Controller Registers Summary

VIS Register Controller Registers			
Offset Start	**Offset End**	**Symbol**	**Register Name/Function**
20000	20003	VISACTLADDR	VIS Controller Address/Index Register
20004	20007	VISACTLDATA	VIS Controller Data Register
20008	2000B	VISARPLYSEQ0	VIS Controller Replay Sequence Control Register 0
2000C	2000F	VISARPLYSEQ1	VIS Controller Replay Sequence Control Register 1
20010	20013	VISARPLYSEQ2	VIS Controller Replay Sequence Control Register 2
20014	20017	VISARPLYSEQ3	VIS Controller Replay Sequence Control Register 3
20040	20043	VISACTLCAP	VIS Controller Capabilities Register

VIS Register Controller Registers			
Offset Start	**Offset End**	**Symbol**	**Register Name/Function**
20080	20083	VISACTLCLM0	VISCTLCLM0
20084	20087	VISACTLCLM1	VISCTLCLM1
20088	2008B	VISACTLCLM2	VISCTLCLM2
2008C	2008F	VISACTLCLM3	VISCTLCLM3
20090	20093	VISACTLCLM4	VISCTLCLM4
20094	20097	VISACTLCLM5	VISCTLCLM5
20098	2009B	VISACTLCLM6	VISCTLCLM6
2009C	2009F	VISACTLCLM7	VISCTLCLM7
200A0	200A3	VISACTLCLM8	VISCTLCLM8
200A4	200A7	VISACTLCLM9	VISCTLCLM9
22814	22817	VISACTLULM0	VISCTLULM0
22818	2281B	VISACTLULM1	VISCTLULM1
2281C	2281F	VISACTLULM2	VISCTLULM2
22820	22823	VISACTLULM3	VISCTLULM3
22824	22827	VISACTLULM4	VISCTLULM4
22828	2282B	VISACTLULM5	VISCTLULM5
2282C	2282F	VISACTLULM6	VISCTLULM6
22830	22833	VISACTLULM7	VISCTLULM7
22834	22837	VISACTLULM8	VISCTLULM8
22838	2283B	VISACTLULM9	VISCTLULM9

13.14.2 VISACTLADDR: VIS Controller Address/Index Register

CSR Register Name: **VISACTLADDR**: VIS Controller Address/Index Register				
Bar: CSR_MTB_BAR		**Reset:** vrc_rst_b	**Offset Start:** 20000	**Offset End:** 20003
Bits	**Access**	**Default**	**Label**	**Bit Description**
31	RW	0		
30	RW	0		
29:28	RW	0		
27:26	RO	0		
25	RW	0		

<table>
<tr><td colspan="6"><b>CSR Register Name: VISACTLADDR</b>: VIS Controller Address/Index Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> vrc_rst_b</td><td><b>Offset Start:</b>
20000</td><td><b>Offset End:</b>
20003</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>24</td><td>RO/V</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>23:16</td><td>RW</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>15</td><td>RW</td><td>6</td><td></td><td colspan="2"></td></tr>
<tr><td>14</td><td>RW</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>13:5</td><td>RW</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>4:0</td><td>RW</td><td>0</td><td></td><td colspan="2"></td></tr>
</table>

13.14.3 VISACTLDATA: VIS Controller Data Register

<table>
<tr><td colspan="6"><b>CSR Register Name: VISACTLDATA</b>: VIS Controller Data Register</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> vrc_rst_b</td><td><b>Offset Start:</b>
20004</td><td><b>Offset End:</b>
20007</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>31:0</td><td>RW</td><td>0</td><td></td><td colspan="2"></td></tr>
</table>

13.14.4 VISARPLYSEQ0: VIS Controller Replay Sequence Control Register 0

<table>
<tr><td colspan="6"><b>CSR Register Name: VISARPLYSEQ0</b>: VIS Controller Replay Sequence Control Register 0</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> vrc_rst_b</td><td><b>Offset Start:</b>
20008</td><td><b>Offset End:</b>
2000B</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>31:28</td><td>RO</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>27</td><td>RW</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>26</td><td>RW</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>25</td><td>RO/V</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>24</td><td>RO/V</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>23:16</td><td>RW</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>15:8</td><td>RW</td><td>0</td><td></td><td colspan="2"></td></tr>
<tr><td>7:0</td><td>RW</td><td>0</td><td></td><td colspan="2"></td></tr>
</table>

13.14.5 VISARPLYSEQ1: VIS Controller Replay Sequence Control Register 1

CSR Register Name: VISARPLYSEQ1: VIS Controller Replay Sequence Control Register 1				
Bar: CSR_MTB_BAR		**Reset:** vrc_rst_b	**Offset Start:** 2000C	**Offset End:** 2000F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:28	RO	0		
27	RW	0		
26	RW	0		
25	RO/V	0		
24	RO/V	0		
23:16	RW	0		
15:8	RW	0		
7:0	RW	0		

13.14.6 VISARPLYSEQ2: VIS Controller Replay Sequence Control Register 2

CSR Register Name: VISARPLYSEQ2: VIS Controller Replay Sequence Control Register 2				
Bar: CSR_MTB_BAR		**Reset:** vrc_rst_b	**Offset Start:** 20010	**Offset End:** 20013
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:28	RO	0		
27	RW	0		
26	RW	0		
25	RO/V	0		
24	RO/V	0		
23:16	RW	0		
15:8	RW	0		
7:0	RW	0		

13.14.7 VISARPLYSEQ3: VIS Controller Replay Sequence Control Register 3

CSR Register Name: VISARPLYSEQ3: VIS Controller Replay Sequence Control Register 3			
Bar: CSR_MTB_BAR	**Reset:** vrc_rst_b	**Offset Start:** 20014	**Offset End:** 20017

Bits	Access	Default	Label	Bit Description
31:28	RO	0		
27	RW	0		
26	RW	0		
25	RO/V	0		
24	RO/V	0		
23:16	RW	0		
15:8	RW	0		
7:0	RW	0		

13.14.8 VISACTLCAP: VIS Controller Capabilities Register

CSR Register Name: VISACTLCAP: VIS Controller Capabilities Register			
Bar: CSR_MTB_BAR	**Reset:** vrc_rst_b	**Offset Start:** 20040	**Offset End:** 20043

Bits	Access	Default	Label	Bit Description
31	RW	0		
30:24	RO	0		
23:16	RO	4		
15:8	RO	40		
7:4	RO	4		
3:0	RO	2		

13.14.9 VISACTLCLM0: VISCTLCLM0

CSR Register Name: VISACTLCLM0: VISCTLCLM0			
Bar: CSR_MTB_BAR	**Reset:** vrc_rst_b	**Offset Start:** 20080	**Offset End:** 20083

Bits	Access	Default	Label	Bit Description
31:0	RW	0		

13.14.10 VISACTLCLM1: VISCTLCLM1

CSR Register Name: VISACTLCLM1: VISCTLCLM1				
Bar: CSR_MTB_BAR		**Reset:** vrc_rst_b		**Offset Start:** 20084 | **Offset End:** 20087
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RW	0		

13.14.11 VISACTLCLM2: VISCTLCLM2

CSR Register Name: VISACTLCLM2: VISCTLCLM2				
Bar: CSR_MTB_BAR		**Reset:** vrc_rst_b		**Offset Start:** 20088 | **Offset End:** 2008B
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RW	0		

13.14.12 VISACTLCLM3: VISCTLCLM3

CSR Register Name: VISACTLCLM3: VISCTLCLM3				
Bar: CSR_MTB_BAR		**Reset:** vrc_rst_b		**Offset Start:** 2008C | **Offset End:** 2008F
Bits	**Access**	**Default**	**Label**	**Bit Description**
31:0	RW	0		

13.14.13 VISACTLCLM4: VISCTLCLM4

CSR Register Name: VISACTLCLM4: VISCTLCLM4			
Bar: CSR_MTB_BAR	**Reset:** vrc_rst_b	**Offset Start:** 20090	**Offset End:** 20093

Bits	Access	Default	Label	Bit Description
31:0	RW	0		

13.14.14 VISACTLCLM5: VISCTLCLM5

CSR Register Name: VISACTLCLM5: VISCTLCLM5			
Bar: CSR_MTB_BAR	**Reset:** vrc_rst_b	**Offset Start:** 20094	**Offset End:** 20097

Bits	Access	Default	Label	Bit Description
31:0	RW	0		

13.14.15 VISACTLCLM6: VISCTLCLM6

CSR Register Name: VISACTLCLM6: VISCTLCLM6			
Bar: CSR_MTB_BAR	**Reset:** vrc_rst_b	**Offset Start:** 20098	**Offset End:** 2009B

Bits	Access	Default	Label	Bit Description
31:0	RW	0		

13.14.16 VISACTLCLM7: VISCTLCLM7

CSR Register Name: VISACTLCLM7: VISCTLCLM7			
Bar: CSR_MTB_BAR	**Reset:** vrc_rst_b	**Offset Start:** 2009C	**Offset End:** 2009F

Bits	Access	Default	Label	Bit Description
31:0	RW	0		

13.14.17 VISACTLCLM8: VISCTLCLM8

CSR Register Name: VISACTLCLM8: VISCTLCLM8			
Bar: CSR_MTB_BAR	**Reset:** vrc_rst_b	**Offset Start:** 200A0	**Offset End:** 200A3

Bits	Access	Default	Label	Bit Description
31:0	RW	0		

13.14.18 VISACTLCLM9: VISCTLCLM9

<table>
<tr><td colspan="6"><b>CSR Register Name: VISACTLCLM9</b>: VISCTLCLM9</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> vrc_rst_b</td><td><b>Offset Start:</b>
200A4</td><td><b>Offset End:</b>
200A7</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>31:0</td><td>RW</td><td>0</td><td></td><td colspan="2"></td></tr>
</table>

13.14.19 VISACTLULM0: VISCTLULM0

<table>
<tr><td colspan="6"><b>CSR Register Name: VISACTLULM0</b>: VISCTLULM0</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> vrc_rst_b</td><td><b>Offset Start:</b>
22814</td><td><b>Offset End:</b>
22817</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>31:0</td><td>RW</td><td>0</td><td></td><td colspan="2"></td></tr>
</table>

13.14.20 VISACTLULM1: VISCTLULM1

<table>
<tr><td colspan="6"><b>CSR Register Name: VISACTLULM1</b>: VISCTLULM1</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> vrc_rst_b</td><td><b>Offset Start:</b>
22818</td><td><b>Offset End:</b>
2281B</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>31:0</td><td>RW</td><td>0</td><td></td><td colspan="2"></td></tr>
</table>

13.14.21 VISACTLULM2: VISCTLULM2

<table>
<tr><td colspan="6"><b>CSR Register Name: VISACTLULM2</b>: VISCTLULM2</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> vrc_rst_b</td><td><b>Offset Start:</b>
2281C</td><td><b>Offset End:</b> 2281F</td></tr>
</table>

Bits	Access	Default	Label	Bit Description
31:0	RW	0		

13.14.22 VISACTLULM3: VISCTLULM3

CSR Register Name: VISACTLULM3: VISCTLULM3					
Bar: CSR_MTB_BAR		**Reset:** vrc_rst_b		**Offset Start:** 22820	**Offset End:** 22823
Bits	**Access**	**Default**	**Label**	**Bit Description**	
31:0	RW	0			

13.14.23 VISACTLULM4: VISCTLULM4

CSR Register Name: VISACTLULM4: VISCTLULM4					
Bar: CSR_MTB_BAR		**Reset:** vrc_rst_b		**Offset Start:** 22824	**Offset End:** 22827
Bits	**Access**	**Default**	**Label**	**Bit Description**	
31:0	RW	0			

13.14.24 VISACTLULM5: VISCTLULM5

CSR Register Name: VISACTLULM5: VISCTLULM5					
Bar: CSR_MTB_BAR		**Reset:** vrc_rst_b		**Offset Start:** 22828	**Offset End:** 2282B
Bits	**Access**	**Default**	**Label**	**Bit Description**	
31:0	RW	0			

13.14.25 VISACTLULM6: VISCTLULM6

CSR Register Name: VISACTLULM6: VISCTLULM6					
Bar: CSR_MTB_BAR		**Reset:** vrc_rst_b		**Offset Start:** 2282C	**Offset End:** 2282F

Bits	Access	Default	Label	Bit Description
31:0	RW	0		

13.14.26 VISACTLULM7: VISCTLULM7

<table>
<tr><td colspan="6"><b>CSR Register Name: VISACTLULM7</b>: VISCTLULM7</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> vrc_rst_b</td><td><b>Offset Start:</b>
22830</td><td><b>Offset End:</b>
22833</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>31:0</td><td>RW</td><td>0</td><td></td><td colspan="2"></td></tr>
</table>

13.14.27 VISACTLULM8: VISCTLULM8

<table>
<tr><td colspan="6"><b>CSR Register Name: VISACTLULM8</b>: VISCTLULM8</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> vrc_rst_b</td><td><b>Offset Start:</b>
22834</td><td><b>Offset End:</b>
22837</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>31:0</td><td>RW</td><td>0</td><td></td><td colspan="2"></td></tr>
</table>

13.14.28 VISACTLULM9: VISCTLULM9

<table>
<tr><td colspan="6"><b>CSR Register Name: VISACTLULM9</b>: VISCTLULM9</td></tr>
<tr><td colspan="2"><b>Bar:</b> CSR_MTB_BAR</td><td colspan="2"><b>Reset:</b> vrc_rst_b</td><td><b>Offset Start:</b>
22838</td><td><b>Offset End:</b>
2283B</td></tr>
<tr><td><b>Bits</b></td><td><b>Access</b></td><td><b>Default</b></td><td><b>Label</b></td><td colspan="2"><b>Bit Description</b></td></tr>
<tr><td>31:0</td><td>RW</td><td>0</td><td></td><td colspan="2"></td></tr>
</table>

§