CEVA-XM4™ OCEM Reference Guide

Rev 1.1.3.F

June 2016

Documentation Control

History Table

Version	Date	Description	Remarks
1.0.0.F	January 2015	Beta Release	
1.1.0.F	August 2015	Official Release	
1.1.1.0	October 2015	Official Release	
1.1.1.A	January 2016	Official Release	
1.1.1.F	February 2016	Official Release	
1.1.2.F	March 2016	Official Release	
1.1.3.F	June 2016	Official Release	

Disclaimer and Proprietary Information Notice

Support

CEVA® makes great efforts to provide a user-friendly software and hardware development environment. Along with this, CEVA provides comprehensive documentation, enabling users to learn and develop applications on their own. Due to the complexities involved in the development of DSP applications that might be beyond the scope of the documentation, an online Technical Support Service has been established. This service includes useful tips and provides fast and efficient help, assisting users to quickly resolve development problems.

How to Get Technical Support:

- **FAQs**: Visit our website http://www.ceva-dsp.com or your company's protected page on the CEVA website for the latest answers to frequently asked questions.
- **Application Notes**: Visit our website http://www.ceva-dsp.com or your company's protected page on the CEVA website for the latest application notes.
- **Email**: Use the CEVA central support email address ceva-support@ceva-dsp.com. Your email will be forwarded automatically to the relevant support engineers and tools developers who will provide you with the most professional support to help you resolve any problem.
- **License Keys**: Refer any license key requests or problems to sdtkeys@ceva-dsp.com. For SDT license keys installation information, see the *SDT Installation and Licensing Scheme Guide*.

Email: ceva-support@ceva-dsp.com
Visit us at: www.ceva-dsp.com

List of Sales and Support Centers

Israel	USA	Ireland	Sweden
2 Maskit Street P.O. Box 2068 Herzelia 46120 Israel **Tel:** +972 9 961 3700 **Fax:** +972 9 961 3800	1174 Castro Street Suite 210 Mountain View, CA 94040 USA **Tel**: +1-650-417-7923 **Fax**: +1-650-417-7924	Segrave House 19/20 Earlsfort Terrace 3rd Floor Dublin 2 Ireland **Tel**: +353 1 237 3900 **Fax**: +353 1 237 3923	Klarabergsviadukten 70 Box 70396 107 24 Stockholm, Sweden **Tel**: +46(0)8 506 362 24 **Fax**: +46(0)8 506 362 20
China (Shanghai)	**China (Beijing)**	**China (Shenzhen)**	**Hong Kong**
Unit 1203, Building E Chamtime Plaza Office Lane 2889, Jinke Road Pudong New District Shanghai, 201203 China **Tel**: +86-21-20577000 **Fax**: +86-21-20577111	Rm 503, Tower C Raycom InfoTech Park No.2, Kexueyuan South Road Haidian District Beijing 100190 China **Tel**: +86-10 5982 2285 **Fax**: +86-10 5982 2284	Rm 709, Tower A SCC Financial Centre No. 88 First Haide Avenue Nanshan District Shenzhen 518064 China **Tel**: +86-755-8435 6038 **Fax**: +86-755-8435 6077	Level 43, AIA Tower 183 Electric Road North Point Hong Kong **Tel**: +852-39751264
South Korea	**Taiwan**	**Japan**	**France**
#478, Hyundai Arion 147, Gumgok-Dong Bundang-Gu Sungnam-Si Kyunggi-Do, 463-853 South Korea **Tel**: +82-31-704-4471 **Fax**:+82-31-704-4479	Room 621 No.1, Industry E, 2nd Rd Hsinchu, Science Park Hsinchu 300 Taiwan R.O.C **Tel**: +886 3 5798750 **Fax**: +886 3 5798750	1-6-5 Shibuya SK Aoyama Bldg. 3F Shibuya-ku, Tokyo 150-0002 Japan **Tel**: +81 3 5774 8250	RivieraWaves S.A.S 400, avenue Roumanille Les Bureaux Green Side 5, Bât 6 06410 Biot - Sophia Antipolis France **Tel**: +33 4 83 76 06 00 **Fax**: +33 4 83 76 06 01

Table of Contents

List of Figures

List of Tables

1 Introduction

The CEVA-XM4™ emulation and debugger interface is managed by the On-Chip Emulation Module (OCEM). The OCEM supports various debugging capabilities. The OCEM offers a glueless approach using a scan-chain methodology, eliminating the usage of a mailbox and a monitor programs. All communication with the host is carried out via an APB Slave or standard JTAG ports.

The CEVA-XM4 enters debug mode either when a breakpoint occurs or when the debugger issues a stop request. A debug session is then initiated and the debugger gains access to the internal core registers and memories.

During a debug session, the debugger can make the core execute any instruction by feeding its opcode to the core via a dedicated scan chain that is inserted into the core. Once in debug mode the core's clock is controlled by the debugger. For example, in order to read data memory a load instruction is injected, the loaded data is then read by the debugger from the destination register. Any memory location (program and data) can be read from or written to using dedicated scan chains.

1.1 OCEM Features

The OCEM includes the following features:

- Scalar and VPU register access
- Program and data memory access
- Program counter (PC) profiling
- Two program address breakpoints with 16-bit counter
- Data address breakpoint with 16-bit counter
- Two external breakpoint requests
- DMA address breakpoint
- File I/O support
- Data value match breakpoints with 16-bit counter
- Combined address and data breakpoints
- Profiler

1.2 OCEM Registers and Chain Configuration

The OCEM debug mechanism can be operated using one of the two methods using the JTAG scan chain or using the AMBA3 APB3 slave port, selecting the method to use is done by the JT_AP signal.

1.2.1 JTAG

The following chapter describes the JTAG mechanism.

1.2.1.1 Test Access Port Controller

The standard Test Access Port (TAP) controller state machine is implemented in the OCEM. Figure 1-1 illustrates the state machine states and transitions:

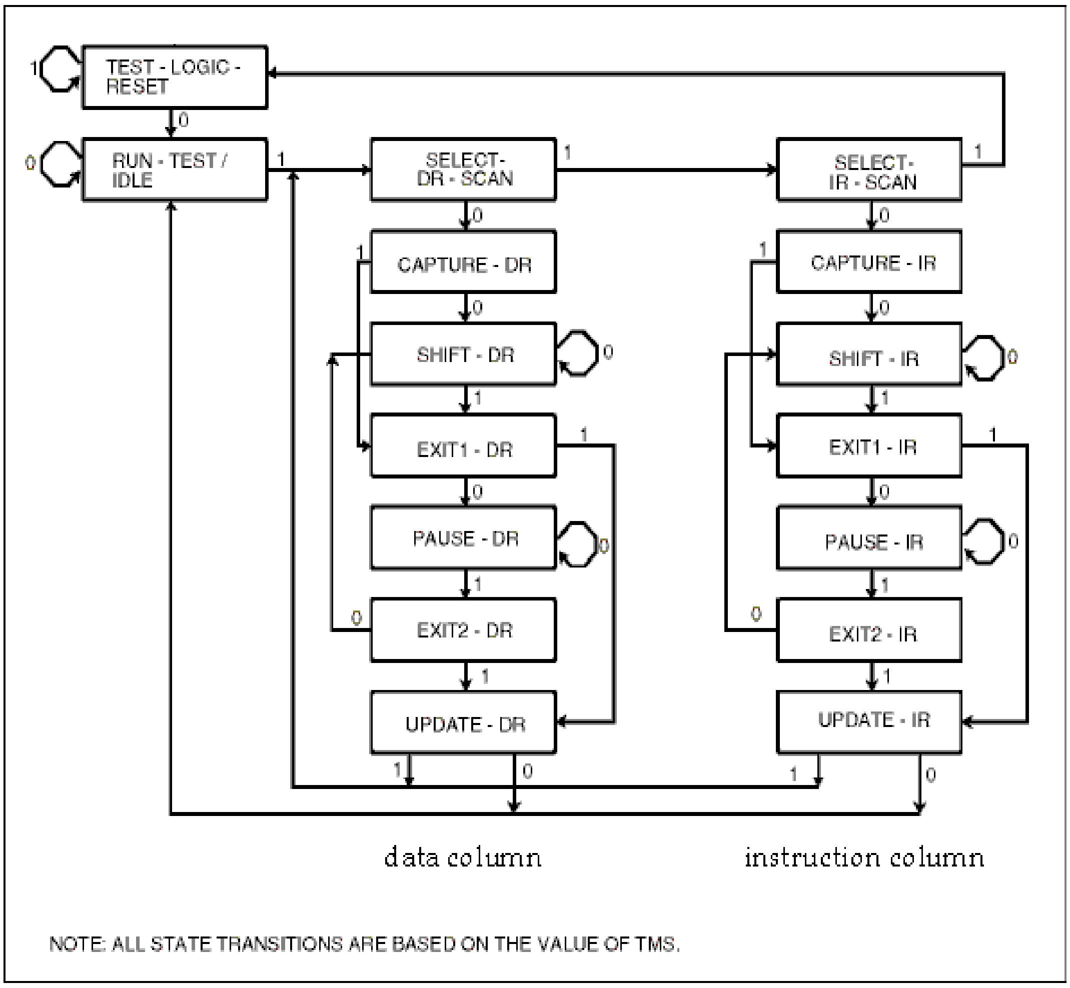


Figure 1-1. TAP Controller State Diagram

1.2.1.2 JTAG Instruction Register

The OCEM includes an instruction register (IR) that controls which data register or scan chain is accessed by the debugger. The instruction register is 32-bits wide. The control codes of the data registers and scan chains are detailed in Section 1.4, OCEM Scan Chains.

The reset value of the instruction register is 0xA0, which is the code of the IDCODE instruction, as required by the JTAG standard.

The instruction register supports also the standard JTAG instructions. Refer to Section 1.2.1.3, Standard JTAG Support for the list of instructions and their access codes.

1.2.1.3 Standard JTAG Support

The OCEM supports the boundary scan register, and the standard JTAG mandatory instructions. This support includes the following signals:

- bs_reg_tdo – output of the boundary scan register, input to the OCEM
- ocm_jtag_state_r – TAP controller state.

The ocm_jtag_state_r signal is an output from the OCEM. It can be used outside the OCEM, to control the boundary scan register of the chip. (To implement the capture, shift and update operations, for the instructions that operate the boundary scan register.)

The following tables define the encoding of ocm_jtag_state_r:

Table 1-1. ocm_jtag_state_r Encoding

Code	State
0x0	EXIT2_DR
0x1	EXIT1_DR
0x2	SHIFT_DR
0x3	PAUSE_DR
0x4	SELECT_IR
0x5	UPDATE_DR
0x6	CAPTURE_DR
0x7	SELECT_DR
0x8	EXIT2_IR
0x9	EXIT1_IR
0xB	SHIFT_IR
0xA	PAUSE_IR
0xC	RUN-TEST_IDLE
0xD	UPDATE_IR
0xE	CAPTURE_IR
0xF	RESET

1.2.1.4 JTAG Standard Identification Register

The JTAG standard defines an identification register. This register is implemented in the JTAG module, according to the standard. Table 1-2 details its fields:

Table 1-2. Identification Register

Field	Bits	Value
Version	31:28	0x0
Part number	27:12	Core Version (for example 0x4210)
Manufacturer ID	11:1	11'b01001010010
LSB	0	1'b1

The full core version register can also be read by the host, using the JTAG scan chain 0x72.

1.2.1.5 TDO Enable Signal

The tdo_oen is a supplement to the JTAG interface signals. It is active low, and denotes that the TAP state machine is either in SHIFT_IR or SHIFT_DR state. This signal is an output of the OCEM. This signal is also asserted during scan tests, when the testmodep signal is asserted.

1.3 CoreSight

1.3.1 Visible Component Architecture

The visible component architecture specifies aspects of components that are visible to the programming interface and to tools that access the device. The visible component architecture specifies the programmer's model and requirements for topology detection.

The programmer's model specifies various registers for the identification and control of the component.

The topology detection registers provide the means for the process of topology detection in the CoreSight system.

1.3.2 Accessing the OCEM Chain

The OCEM Scan Chains are accessed using the SCCO and SCDA registers where the Scan Chain code is written to the SCCO.SCC. The direction (read/write) is written to the SCCO.DIREC field, the number of data registers is written to SCCO.SIZE field and the data are read or written using the SCDA.SCD field.

The chain is activated by writing the code, direction and number of registers, and then writing or reading the data in consecutive requests.

The number of registers to be written is the chain length divided by 32.

For example:

If the chain length is 164, the SCCO.SIZE will be 6, and the number of times the SCDA is written is 6, when the 32 LSBs of the chain are written first, and the last register uses only 4 bits that need to be located at the LSBs.

1.3.3 CoreSight Programmer’s Model

This chapter defines the standard set of registers that you must implement on all CoreSight components see Section 1.3.3, CoreSight Programmer’s Model, in addition to the control registers specific to components. Some registers are optional and must read as zero when they are not implemented.

1.4 OCEM Scan Chains

The debugger uses CoreSight or JTAG serial interface to access the OCEM scan chains.

Some of the scan chains consist of several buses and control signals. Some scan chains are a subset of other, longer chains (master).

The scan chains are grouped in accordance with the hardware function access used.

- CPM Registers – Access to a section of the core programing model registers used for debug and emulation. CPM Registers can be read/written to using the scan chains, APB3 Slave and the core IO interface
- Instruction Insertion – For pushing instruction fetch lines (or parts of them) into the DSP core.
- PMSS Access – Program memory access. Enables read and write access to internal and external program memory.
- DMSS Access – Data memory access. Enables read and write access to L1 data cache and external data memory.
- Vector access – Enables direct read and write access to the vector registers of the VPUs.
- Core Access – Core data read register. Moving 32 bit data from the core to the host.
- Real-Time Trace – Control registers of the CEVA-XM4 real-time trace.

The following table details the structure of the chains, the control codes of the OCEM scan chains, their size and access type.

A chain that starts with m_ is a master chain.

A chain that starts with s_ is a subset of a master chain.

Table 1-3. OCEM Scan Chains

Code	Name	Structure	Description	JTAG Access	APB Access	Size
			OCEM Clock Control			
0x70	OCM_CLOCK	{ock[3:0]}	Number of cycles OCEM should apply to the core. The oclk_applied_r status is reflected on the tdo by keeping the TAP control at SHIFT_DR state	W	W	4+

Table 1-3. OCEM Scan Chains (Continued)

Code	Name	Structure	Description	JTAG Access	APB Access	Size
	CPM Registers					
0x04	OCM_DVM	Refer to OCM_DVM register for details on scan chain structure	Refer to OCM_DVM register for description	R/W	R/W	32
0x08	OCM_OFIO7P	Refer to OCM_OFIO7P register for details on scan chain structure	Refer to OCM_OFIO7P register for description	R	R	32
0x0C	OCM_OFIO8P	Refer to OCM_OFIO8P register for details on scan chain structure	Refer to OCM_OFIO8P register for description	R	R	32
0x10	OCM_PADD1	Refer to OCM_PADD1 for register details on scan chain structure	Refer to OCM_PADD1 register for description	R/W	R/W	32
0x14	OCM_PADD2	Refer to OCM_PADD2 register for details on scan chain structure	Refer to OCM_PADD2 register for description	R/W	R/W	32
0x18	OCM_OFIO1	Refer to OCM_OFIO1 register for details on scan chain structure	Refer to OCM_OFIO1 register for description	R/W	R/W	32
0x1C	OCM_OFIO2	Refer to OCM_OFIO2 register for details on scan chain structure	Refer to OCM_OFIO2 register for description	R/W	R/W	32
0x20	OCM_PCOUNT1	Refer to OCM_PCOUNT1 register for details on scan chain structure	Refer to OCM_PCOUNT1 register for description	R/W	R/W	16
0x24	OCM_PCOUNT2	Refer to OCM_PCOUNT2 register for details on scan chain structure	Refer to OCM_PCOUNT2 register for description	R/W	R/W	16
0x28	OCM_OFIO3	Refer to OCM_OFIO3 register for details on scan chain structure	Refer to OCM_OFIO3 register for description	R/W	R/W	32
0x2C	OCM_OFIO4	Refer to OCM_OFIO4 register for details on scan chain structure	Refer to OCM_OFIO4 register for description	R/W	R/W	32
0x30	OCM_DADD_LOW	Refer to OCM_DADD_LOW register for details on scan chain structure	Refer to OCM_DADD_LOW register for description	R/W	R/W	32

Table 1-3. OCEM Scan Chains (Continued)

Code	Name	Structure	Description	JTAG Access	APB Access	Size
0x34	CEVAX_D3_PC	Refer to CEVAX_D3_PC register for details on scan chain structure	Refer to CEVAX_D3_PC register for description	R	R	32
0x38	OCM_OFIO5	Refer to OCM_OFIO5 register for details on scan chain structure	Refer to OCM_OFIO5 register for description	R/W	R/W	32
0x3C	OCM_OFIO6	Refer to OCM_OFIO6 register for details on scan chain structure	Refer to OCM_OFIO6 register for description	R/W	R/W	32
0x40	OCM_DADD_HIGH	Refer to OCM_DADD_HIGH register for details on scan chain structure	Refer to OCM_DADD_HIGH register for description	R/W	R/W	32
0x48	OCM_OFIO7	Refer to OCM_OFIO7 register for details on scan chain structure	Refer to OCM_OFIO7 register for description	R/W	R/W	32
0x4C	OCM_OFIO8	Refer to OCM_OFIO8 register for details on scan chain structure	Refer to OCM_OFIO8 register for description	R/W	R/W	32
0x50	OCM_CONTROL	Refer to OCM_CONTROL register for details on scan chain structure	Refer to OCM_CONTROL register for description	R/W	R/W	32
0x54	OCM_SA_BP_EN	Refer to OCM_SA_BP_EN register for details on scan chain structure	Refer to OCM_SA_BP_EN register for description	R/W	R/W	32
0x5C	MSS_CONFIG	Refer to MSS_CONFIG register for details on scan chain structure	Refer to MSS_CONFIG register for description	R	R	32
0x60	OCM_STATUS	Refer to OCM_STATUS register for details on scan chain structure	Refer to OCM_STATUS register for description	R	R	32
0x64	OCM_SA_BP_ST	Refer to OCM_SA_BP_ST register for details on scan chain structure	Refer to OCM_SA_BP_ST register for description	R	R	32
0x68	AXIM_CONFIG	Refer to AXIM_CONFIG register for details on scan chain structure	Refer to AXIM_CONFIG register for description	R	R	32

Table 1-3. OCEM Scan Chains (Continued)

Code	Name	Structure	Description	JTAG Access	APB Access	Size
0x6C	AXIS_CONFIG	Refer to AXIS_CONFIG register for details on scan chain structure	Refer to AXIS_CONFIG register for description	R	R	32
0x72	CORE_VERSION	Refer to CORE_VERSION register for details on scan chain structure	Refer to CORE_VERSION register for description	R	R	32
0x78	CORE_ID	Refer to CORE_ID register for details on scan chain structure	Refer to CORE_ID register for description	R	R	32
0x7C	CORE_CONFIG	Refer to CORE_CONFIG register for details on scan chain structure	Refer to CORE_CONFIG register for description	R	R	32
0xF0	OCM_ACOUNT	Refer to OCM_ACOUNT register for details on scan chain structure	Refer to OCM_ACOUNT register for description	R/W	R/W	32
0xF4	OCM_DCOUNT	Refer to OCM_DCOUNT register for details on scan chain structure	Refer to OCM_DCOUNT register for description	R/W	R/W	32
Instruction insertion						
0x80	m_inst_scan	inst_scan[255:0]	Instruction scan chain (used to inject instructions to the core)	W	W	256
0x81	s_inst_scan_64msb	inst_scan[255:192]	Instruction scan chain, the 64 MSBs	W	W	64
0x82	s_inst_imm_scan	{inst_scan[249:224], inst_scan[202:197]}	Immediate value scan chain – a subset of the instruction scan chain (used to modify the immediate value in the instruction)	W	W	32
PMSS access						
0x83	m_pmem_int	{tcm_rd, tcm_wr, set_rd, set_wr, way[1:0],int_address[26:0],int_data[255:0]}	Internal program memory (TCM and I$) scan chain	R/W	R/W	289
0x84	s_pmem_int_ctrl	{tcm_rd, tcm_wr, set_rd, set_wr, way[1:0],int_address[26:0]}	Internal program memory (TCM and I$) scan chain, control segment	W	W	33

Table 1-3. OCEM Scan Chains (Continued)

Code	Name	Structure	Description	JTAG Access	APB Access	Size
0x85	s_pmem_int_data	int_data[255:0]	Internal program memory (TCM and I$) scan chain, data bus segment	R/W	R/W	256
0x86	m_pmem_ext_rd	{prd_valid, L2A [3:0], QOS[3:0], ext_address[26:0], ext_data[255:0]}	External program memory scan chain, read access	R/W	R/W	292
0x87	s_pmem_ext_rd_ctrl	{L2A [3:0], QOS[3:0], ext_address[26:0]}	External program memory scan chain, read access, control segment	R/W	R/W	35
0x88	s_pmem_ext_rd_data	{prd_valid, ext_data[255:0]}	External program memory scan chain, read access, data bus segment	R/W	R	257
0x89	m_pmem_ext_wr	{L2A [3:0], QOS[3:0], ext_address[27:0], ext_data[127:0]}	External program memory scan chain, write access	W	W	164
0x8A	s_pmem_ext_wr_ctrl	{L2A [3:0], QOS[3:0], ext_address[27:0]}	External program memory scan chain, write access, control segment	W	W	36
0x8B	s_pmem_ext_wr_data	ext_data[127:0]	External program memory scan chain, write access, data bus segment	W	W	128
0x90	m_pmem_tag_rd	{way[1:0], int_address[9:0]	Tag memory scan chain, read address	W	W	12
0x91	m_pmem_tag_wr	{way[1:0], lock,valid, int_address[26:0]}	Tag memory scan chain, write address	W	W	31
0x92	m_pmem_tag_data	{tag_data[18:0], lock,valid}	Tag memory scan chain, data segment	R	R	21
	Vector Access					
0x93	vec_rd	{v_sel[5:0], vrd_valid, vec_data[255:0]}	vector scan chain, vector read valid, read access	R/W	R/W	263
0x95-0x9F	Reserved					
0x00	JTAG					
0xFF	JTAG					
0xA0-0xA5	JTAG					

Table 1-3. OCEM Scan Chains (Continued)

Code	Name	Structure	Description	JTAG Access	APB Access	Size
0xA6-0xB3	Reserved					
			Real Time Trace			
0xC0	mode	wr_resetwr_en, etmx2, cid_en, no_stall, pr_trace, ls0_trace, ls1_trace	Mode configuration	W	W	32
0xC1	cmp_cfg	br_all, br_fld, cmp2_cfg[1:0],cmp1_cfg[1:0], cmp0_cfg[1:0]	Comparator configuration	W	W	32
0xC2	ls0_cfg	ls1_tr_cfg[2:0], ls1_lg, ls1_cmp_cfg[2:0]	Load store unit-0 configuration	W	W	32
0xC3	ls1_cfg	ls1_tr_cfg[2:0], ls1_lg, ls1_cmp_cfg[2:0]	Load store unit-0 configuration	W	W	32
0xC4	cmp0_start	cmp0_start[31:0]	Data Address Range Comparator #0 start address	W	W	32
0xC5	cmp0_end	cmp0_end [31:0]	Data Address Range Comparator #0 end address	W	W	32
0xC6	cmp1_start	cmp1_start[31:0]	Data Address Range Comparator #1 start address	W	W	32
0xC7	cmp1_end	cmp1_end[31:0]	Data Address Range Comparator #1 end address	W	W	32
0xC8	cmp2_start	cmp2_start [31:0]	Data Address Range Comparator #2 start address	W	W	32
0xC9	cmp2_end	cmp2_end [31:0]	Data Address Range Comparator #2 end address	W	W	32
			DMSS			
0xD0-0xD2	Reserved for D$					
0xD3-0xDF	Reserved for D$					
	Core access					
0xFC	core_drd_rd	{core_drd[31:0],drd_valid}	Core data read register, data read valid	R	R	33

1.5 Debug Mode

The DEBUG_ENABLE field at the OCM_CONTROL register (scan chain 0x50) enables the core to enter debug mode.

If DEBUG_ENABLE is asserted then the following events cause the core to enter debug mode:

- BI or PABP hardware interrupt is accepted (and the BP_DETECT_EN bit at the OCM_CONTROL register is set)
- Execution of trape instruction (and the TRAPE_EN bit at the OCM_CONTROL register is set)
- Setting the STOP_GO bit at the OCM_CONTROL register

When one of the above events is detected, the core automatically jumps to address 0x0000_0020 while saving the return address (the address following the last executed address) at the retregb return register.

During Debug mode, all interrupts are disabled, including NMIs. The permission level during Debug mode is equivalent to the permission level of the Supervisor mode.

Use the retb instruction to exit Debug mode. The core then returns to the operation mode that was in use prior to the Debug mode.

If DEBUG_ENABLE is not asserted then an enabled breakpoint causes the core to branch to address (0x20 + modH) and begin to execute the instructions from that address. The *retb* instruction causes the core to return to the program address before the breakpoint was accepted.

1.5.1 Before Starting MSS Access

After entering debug mode and before emulation commands can accesses MSS peripherals it is necessary to clear the ADLs lines, finish all MSS ongoing transactions, finish all DDMA and PDMA ongoing bursts transactions and wait for QMAN to complete all outstanding transactions.

This is achieved by:

1. Clearing the ADLs by inserting eight nop instructions and releasing the core clock for eight cycles.
2. Wait for the DMSS_IDLE_OCEM bit at the OCM_STATUS Register to be set, indicating that the write buffer, EDP and AXIS ports are in idle mode.
3. Wait for the PMSS_IDLE bit at the OCM_STATUS Register to be set, indicating that the write buffer, EDP and AXIS ports are in idle mode.

Once in debug mode, the QMAN and DDMA automatically stop issuing new memory access, allowing previous outstanding transactions to complete. If it currently access internal to internal it stops immediately.

1.6 Clock Control

The OCEM stops the clock of the core by activating a wait request called ocm_wait_req_r. This wait request is output to the PSU which activates the core wait signal (cedar_wait_r).

The OCEM de-asserts its wait request signal whenever clock cycles should be applied to the core.

During an emulation session, the debugger can apply clock cycles to the core. The debugger controls when cycles are applied, and how many, by writing to the OCM_CLOCK register (using scan chain 0x70).

The maximum number of clock cycles that can be applied in a single transaction to the OCM_CLOCK register is 15; therefore, the debugger loads a four-bit number to the OCM_CLOCK register.

A clock cycle is considered as non-applied when the core wait signal is asserted, regardless of the cause of the wait.

For example:

During an emulation session the debugger applies 10 clock cycles to the core. During these 10 cycles the core wait signal is asserted for one cycle due to IDM contention. The OCEM considers this cycle as non-applied. Therefore the OCEM will de-assert its wait request signal (ocm_wait_req_r) for 11 cycles (instead of 10 cycles when no IDM contention occurs).

When the count is over, the OCEM asserts the wait request. It also asserts a dedicated signal, called oclk_applied_r. This signal indicates that the clock cycles that were requested were indeed applied. The OCEM cannot be stopped before all the clock cycles are applied.

The OCEM provides two methods for reading the status of the oclk_applied_r signal:

After writing to the OCM_CLOCK register with the 4 bit cycle count the debugger continues performing SHIFT_DR cycles (keeping the TAP control at the SHIFT_DR state). After the four initial cycles of the SHIFT_DR (where the debugger writes the number of applied cycles) the OCEM automatically outputs the status of the oclk_applied_r signal to the tdo output.

The debugger checks the value of the tdo and exits the SHIFT_DR state if the tdo is set.

The Debugger can check the status of the oclk_applied_r signal by reading the CLK_APPLIED field (bit 6) of the General status register (scan chain 0x60).

1.7 Single-Stepping

Single-stepping is performed by setting the STEP bit in the OCM_CONTROL Register. Before setting the STEP bit the debugger loads the retb instruction into the queue. When the STEP bit is set the OCEM releases the core from debug mode. The OCEM activates the BI breakpoint request after executing the retb instruction and the instruction at the return address. This enables a single instruction packet at the target of the retb to be executed, before debug mode is re-entered.

1.8 Real-Time Trace

CEVA-XM4 Real-Time Trace (RTT) solution provides the following features:

Program Instruction Flow trace:

- Cycle and non-cycle accurate program address trace
    - Cycle-accurate trace indicates the number of cycles required to execute each instruction under most trace conditions.
    - Non-cycle accurate trace indicates the execution of an instruction without indicating the number of cycles required.
- High compression achievable by tracing change-of-flow only
- Predicate trace to facilitate condition code reconstruction by decompressor
    - CEVA-XM4 Cores support instruction predication whereby an instruction can be executed or not depending on the value of a specified predicate flag.
    - Predicate flags are sent to the ETM in Normal Data and ContextID packets.

For further details on the CEVA-XM4 Real Time Trace please refer to CEVA-XM4 RealTimeTrace Arch Spec document.

1.9 OCEM Registers Chains

Accessing the OCEM configuration registers can be performed using the following methods:

- JTAG scan chains (scan chains 0x04 through 0x6C and 0x72 through 0xC4)
- Using CEVA-XM4 APB3 Slave interface (at offset 0x0000_0000)
- Using CEVA-XM4 core IO interface (at offset 0x0000_0100)
- Using one of CEVA-XM4 external slave ports (EDAP or ASIS ports at offset 0x0040_2100)

The description and field map of the OCEM CPM registers can be found in Section 5.1, High-level Address Mapping of the Programming Model.

1.10 Slave Port Access During Emulation

During emulation all slave ports can access the MSS normally and have no restrictions.

Note: All the slave ports can be disabled during emulation using the EDAP_DS and AXI_S_DS bits at the MSS

1.11 Cache Coherency

1.11.1 Program Cache Coherency

When using direct access to the cache the user can easily lose the coherency between the behavior of the cache and the corresponding arrays on normal operation and on the direct access using the OCEM chain. The debugger warns the user in the following scenarios.

Table 1-4. Coherency Issues

Operation	Behavior in normal operation	Coherency problem	Debugger solution
Write to set memory	Data is written to internal memory from external memory	Data is written to internal memory different from the external memory	Propose to write to external memory on any write to internal memory.
Write to TAG array	Relevant data is stored at the set	TAG and set are not Coherency or Tag already exist.	warning on unexpected behavior
Write to external memory	Not possible	External memory and cache coherency problems	Propose to write to internal set memory on any write to external memory.

2 Instruction Insertion Chains

Using scan chains 0x80 the host can directly inject instruction fetch lines to the CEVA-XM4 core. Prior to executing the injected fetch line the core Program Counter must be at an address aligned to the fetch line size to ensure execution of instructions at the start of the fetch line.

When entering a debug mode the core automatically jumps to address 0x0000_0020 (aligned to fetch line size) and if the DEBUG_ENABLE bit at the OCM_CONTROL register is set the OCEM freeze the core. To ensure the Program Counter will remain at an aligned address even if the DEBUG_ENABLE bit is un-set. Address 0x0000_0020 should contain a “br 0x20” instruction.

Once a fetch line is injected to the core the host can apply the required cycles to the core by writing to the OCM_CLOCK scan chain (refer to Clock Control).

The fetch line can contain several instruction packets.

The last executed instruction at an injected fetch line must be a “br 0x20” instruction, this ensures that the core instruction queue is empty (flushed) and the first instruction at the next injected fetch line will be executed. When writing to the OCM_CLOCK scan chain the host must apply sufficient cycles to enable the last instruction reach the M stage.

An indication of the queue status can be read using the General Status Register (scan chain 0x60) and checking the Q_FLUSH bit.

Scan chain 0x81 enables the host to inject only the 64 MSBs of the instruction fetch line. Using this scan chain enables fast injection of 64MSBs while maintaining the 192 LSBs from the previous instruction fetch.

Scan chain 0x82 enables the host to inject only the immediate part of a mov instruction with immediate data (SC.mov #<imm32>, r0). This can be useful when loading large sections of data into the IDM. The host can inject a sequence where an immediate value is moved into “r0” and then stored to a post incremented address.

Using scan chain 0x82 the host is required to only scan chain the data without having to inject the same instructions to the core.

2.1 Program Memory Chains

The program memory scan chains support fast access to program memory, which is very important for efficient code download. Therefore, dedicated features associated with these scan chains are implemented:

- Address auto increment mode
- Access to a subset of the chain

There is one scan chain for the internal program memory, and another two for the external program memory (for read access and write access). Each one of these chains has two subsets that can be accessed separately – a control segment and a data segment. When performing an access to a memory block, the address and read/write signal are loaded once, and the address auto increment mode is activated. The OCEM increments the address automatically, and therefore the debugger has to read/write only the data segment of the chain.

The scan chains associated with the program memory interface have two levels of flip-flops – a serial level, and a parallel level. During the SHIFT_DR stage, data is serially shifted into the serial level. At the UPDATE_DR stage, the data is parallel-loaded from the serial level to the parallel level.

Two levels of flip-flops are required since clock cycles can be applied when data is shifted to/from these scan chains. The parallel level outputs are connected to the program memory interface. This ensures that the memory interface is stable when the memory access is performed.

2.1.1 ECC Support

When debug is accessing the program internal memory or inserting instructions to the core, the OCEM insert the data without any overheads. It is the HW responsibility to add the ECC overhead bits (if enabled).

When reading data from memory:

the OCEM insert the address

the memory data (ECC not included is returned to the OCEM chain

When written data to memory or inserting instructions to the core:

the OCEM insert the address and data

the data is calculated for ECC

the data and the ECC bits written to the target

2.1.2 Internal Program Memory Write Access

The following sequence is used when performing a write access to the internal program memory:

- Write the first line (256 bits) to the internal program memory scan chain data segment (int_data[255:0] at s_pmem_int_data scan chain 0x85)
- Write the first address and the write signal (of bank0 or bank1) to the internal program memory scan chain control segment (tcm_rd=0,tcm_wr=1,set_rd=0,set_wr=0,way=0,int_address[26:0] at s_pmem_int_ctrl scan chain 0x84)

When performing an access to a sequential memory block, the following steps are performed as well:

- Activate the address-auto-increment bit in the OCM_CONTROL Register (bit 7, PADD_INC)
- Write the next lines to the internal program memory scan chain data segment (int_data[255:0] at s_pmem_int_data scan chain 0x85). The OCEM increments the address every time the debugger writes a new line to the data segment.

2.1.3 Internal Program Memory Read Access

The following sequence is used when performing a read access from the internal program memory:

- Write the first address and the read signal (of bank0 or bank1) to the internal program memory scan chain control segment (tcm_rd=1,tcm_wr=0,set_rd=0,set_wr=0,way=0,int_address[26:0] at s_pmem_int_ctrl scan chain 0x84)
- Read the internal program memory scan chain data segment (int_data[255:0] at s_pmem_int_data scan chain 0x85)

When performing an access to a sequential memory block, activate the address-auto-increment bit in the OCM_CONTROL Register (bit 7, PADD_INC) after the first address is written to the control segment. In this case, the OCEM increments the address every time the debugger reads the data segment.

2.1.4 External Memory Access

Access to the external program memory is performed by a dedicated scan chain, which is connected to the external port interface. Dedicated request logic controls the request signals that the OCEM outputs to the external port, as explained in the following sections.

2.1.4.1 External Program Memory Write – Single Mode

In single mode (writing to a single memory location), the debugger loads the address and data to the external memory write scan chain (m_pmem_ext_wr scan chain 0x89). When the TAP state machine reaches the UPDATE_DR stage, the OCEM activates the request signal which is output to the MSS. The OCEM clears the request signal when the MSS issues an address grant signal.

An indication of an active memory transaction (active-transaction signal) is reflected using ACTIVE_TRANS field in the OCM_STATUS Register. The debugger can poll this field in order to determine if the transaction is over.

2.1.4.2 External Program Memory Write – Address Increment Mode

In Address Increment Mode (AIM), the debugger can perform several memory accesses to a memory block, without having to load the address every time. A starting address is loaded at the beginning of the transaction (to s_pmem_ext_wr_ctrl scan chain 0x8A), and afterwards only the data segment of the chain is accessed (s_pmem_ext_wr_data scan chain 0x8B).

The debugger serially loads the data to the serial level of the data segment during the SHIFT_DR stage.

The parallel level of the data segment is loaded at the end of the UPDATE_DR stage. The incremented address is parallel-loaded to the control segment also at the end of this stage. The OCEM activates the request signal at the end of this stage, and clears it when the address grant signal from the MSS is active.

The debugger can then start to load the serial level of the data segment with the data of the next transaction. The indication of an active-transaction is output to the debugger on the tdo output, during the SHIFT_DR stages of the transaction. If the debugger receives an inactive value on the active-transaction indication (in any one of the cycles), it means that the previous transaction was successful. Otherwise, the previous transaction is not over yet.

The OCEM does not load the incremented address to the control segment of the chain, if the previous transaction is not over yet. Moreover, the data of the next transaction is not loaded to the parallel level of the data segment. In this case, the debugger should re-load the data of the next transaction to the data segment of the scan chain.

2.1.4.3 External Program Memory Read – Single Mode

In single mode (reading a single memory location), the debugger loads the address (to the s_pmem_ext_rd_ctrl scan chain 0x87). When the TAP state machine reaches the UPDATE_DR stage, the OCEM activates the request signal which is output to the MSS. The OCEM clears the request signal when the MSS issues an address grant signal.

When the data is ready, the MSS issues a data ready indication to the OCEM. The OCEM then loads a read buffer with the data that was read from memory. The data-ready signal itself is also loaded to a dedicated FF, called PRD_VALID, which is part of the data read chain (s_pmem_ext_rd_data scan chain 0x88).

Then, the debugger performs an access to the data segment of the scan chain (s_pmem_ext_rd_data scan chain 0x88) in order to read the data. The data and the PRD_VALID bit are loaded to the serial level of the scan chain (from the read buffer) at the end of the CAPTURE_DR stage.

The debugger determines whether the transaction was successful according to the value of the PRD_VALID bit (a high value indicates a successful transaction). If the transaction is not successful, the debugger re-reads the data.

2.1.4.4 External Program Memory Read – Address Increment Mode

In Address Increment Mode (AIM), the debugger can perform several memory accesses, without having to load the address every time. A starting address is loaded at the beginning of the transaction (to the s_pmem_ext_rd_ctrl scan chain 0x87), and afterwards only the data segment of the chain is accessed.

The incremented address is parallel-loaded to the control segment at the end of the UPDATE_DR stage, unless the active-transaction signal (PRD_VALID) is active. The OCEM activates the request signal at the end of this stage, and clears it when the MSS address grant signal is set.

When the data is ready, the MSS issues a data ready indication to the OCEM. The OCEM then loads a read buffer with the data that was read from memory. The data-ready signal itself is also loaded to a dedicated FF, called PRD_VALID, which is part of the data read chain (s_pmem_ext_rd_data scan chain 0x88).

Then, the debugger performs an access to the data segment of the scan chain (s_pmem_ext_rd_data scan chain 0x88) in order to read the data. The data and the PRD_VALID bit are loaded to the serial level of the scan chain (from the read buffer) at the end of the CAPTURE_DR stage.

The debugger determines whether the transaction was successful based on the value of the PRD_VALID bit (a high value indicates a successful transaction). If the transaction is not successful, the debugger re-reads the data.

The address of the next transaction (the incremented address) and the data of the next transaction are loaded only if the previous transaction is over – that is: only if the active-transaction indication (PRD_VALID) is not active.

2.1.4.5 Accessing L2 Using Program Port

When the OCEM chain accesses the external program port, the debugger will change the CICR and IACU registers before accessing the external port and return them to the original values before releasing the core.

2.1.5 Tag Memory Access

Using the tag scan chains (chains 0x90,0x91 and 0x92) the host can access the tag memory for reading and writing

2.1.5.1 Tag Read Access

The following sequence is used when performing a read access from the Tag memory:

- Write the way tag and the 10 bit int_addr (cache line index that used to select the tag memory entry) to the tag memory scan chain read address (m_pmem_tag_rd chain 0x90).
- Read the tag address, valid and lock from the tag memory scan chain data segment (m_pmem_tag_data scan chain 0x92).

When performing an access to a sequential tag memory block, activate the address-auto-increment bit in the OCM_CONTROL register (bit 7, PADD_INC) after the first address is written to scan chain 0x90 (m_pmem_tag_rd). In this case, the OCEM increments the address every time the debugger reads the tag data segment.

2.1.5.2 Tag Write Access

The following sequence is used when performing a write access to the tag memory:

- Write the way, 27 int_address bits (10 bits cache line index and 19 tag data), valid and lock bits to the tag memory scan chain write address (m_pmem_tag_wr chain 0x91).

The relevant tag memory (according to the way and cache line index) will be loaded with the 19 MSB of the address, valid and lock written to the m_pmem_tag_wr chain.

2.1.5.3 Writing Valid And Lock.

The following sequence is used when writing the valid and lock bits in the tag entry:

- Read the relevant the Tag memory according to 2.1.5.1 Tag Read Access.
- Write the tag memory scan chain using the required data for valid and lock bits according to 2.1.5.2 Tag Write Access.

2.2 Vector Access Chains

Vector access scan chains (0x93)) enables the direct transfer of vector content from the core to the host.

The host selects the vector using the v_sel.

When reading a vector (chains 0x93) the host selects the vector by pushing the vector number on the tdi chain (v_sel[5:0]).

The host then reads the vector data from the tdo output signal (vec_data[255:0]). The vec_in_rd scan chain (0x93) also contains a valid bit indicating that the vector was changed after the last read of the vector by the host. This valid bit is useful when loading data from the IDM to one of the core vectors and reading the data by the host for further details on reading IDM data by the host using vector access see section 2.4.3.2 Data Memory Read Using Vector Load Command

2.3 Data Read Access Chain

The core_drd_rd scan chain (0xFC) enables the direct transfer of 32 bit data from the core to the host.

The OCORE_DRD register is accessed by the core software using the core IO interface. The host reads the written value using the core_drd_rd scan chain. This scan chain contains the 32 bit of the OCORE_DRD register (core_drd[31:0]) and an additional valid bit (drd_valid) indicating the value of the register was changed after the last read by the host.

The OCORE_DRD register is written whenever the address 0xFC is used regardless of the iopage field in mod2 register.

2.4 Debugger Sequences

This section details the various sequences that the debugger uses, in order to perform the various emulation operations.

The sequences are performed by injecting instructions to the core. The debugger scans the opcodes of the sequence to the instruction scan chain, and releases the appropriate amount of clock cycles to the core.

2.4.1 Reading Internal Registers

2.4.1.1 Reading an AR Internal Register

In order to read an internal register, the debugger injects an out instruction to the core. This instruction loads the contents of the register to core data read register of the OCEM (OCORE_DRD). The debugger then reads this register (via JTAG scan chain 0xFC).

The out instruction supports all the AR registers as the source. Therefore, reading these registers is simple: the debugger injects a fetch line with two instructions:

An out instruction with the AR register as the source, and the core data read register of the OCEM (OCORE_DRD) as destination.

A branch instruction is placed after the out instruction in order to align and flush the queue.

Example 2-1. Read an ARF Internal Register

```
out{cpm} r0.di ,(#0x1FC).di            ; write the value at r0 to the OCORE_DRD register
nop
nop
br 20
nop
nop
nop
nop
```

Eight clock cycles are applied in order to enable the r0 data to reach the OCORE_DRD register.

2.4.1.2 Reading MODV Mode Registers

The modv cannot be read by a single mov instructions. The following sequence is performed in order to read the modv:

Example 2-2. Read modv Mode Registers

```
mov modv0.di ,r0.di                    ; move modv0 to r0
nop
nop
nop
nop
out{cpm} r0.di ,(#0x1FC).di                 ; write the value at r0 to the OCORE_DRD register
nop
nop
br 20
nop
nop
```

Thirteen clock cycles are applied in order to enable the r0 data to reach the OCORE_DRD register.

2.4.2 Writing to Internal Registers and Vectors

2.4.2.1 Writing an Internal Register

In order to write an internal register, the debugger injects a mov #imm instruction to the core, with the register that needs to be written as the destination.

The instruction sequence is as follows:

Example 2-3. Writing an Internal Register

```
br 20 mov #0x0, r0.i   ; Immediately move the 32 bits to r0
nop
nop
nop
br 20
nop
nop
nop
nop
```

Nine clock cycles are applied in order to enable the data to reach r0.

2.4.2.2 Writing to a Vector Unit Register

In order to write a vector part, the debugger injects a vmov instruction to the core, with the VPU vector part that needs to be written as the destination.

The instruction sequence is as follows:

Example 2-4. Writing to a Vector Unit Register

```
br 20 mov #imm,r0.i          ; Immediately move the 32 LSB to r0
nop
vmov r0.ui #imm ,v0.i8         ; Immediately move r0 to the destination vector part
nop
nop
nop
br 20
nop
nop
nop
nop
```

Eleven clock cycles are applied in order to enable the data to reach the vector.

2.4.3 Data Memory Read

Data memory can be read using the core or using a dedicated scan chain.

2.4.3.1 Data Memory Read Using Load Command

In order to read a data memory location, the debugger loads the desired address to r1 and performs the following sequence:

Example 2-5. Data Memory Read using Load Command

```
br 20 ld (r1.ui).di[+pm], r0.di          ; load and post increment data from r1
nop
nop
nop
nop
nop
out{cpm} r0.di ,(#0x1FC).di              ;write the value at r0 to the OCORE_DRD register
br 20
nop
nop
nop
nop
```

Eleven clock cycles are applied in order to enable the data to reach the OCORE_DRD.

When reading a continuous section in memory, the address is loaded only once, and the post-modification mechanism of the core is used in order to increment the address. In this case, the instruction sequence is loaded only once. Afterwards, the debugger has to read only the OCORE_DRD register in the successive transactions.

The OCEM supports automatic clock insertion for this transaction. In order to use this mode, the number of clocks (0xB) is loaded to bits 27:24 (AUTO_CLK_NUM field) of the OCM_CONTROL register, and bit 21 (AUTO_CLK field) of the OCM_CONTROL register should be set. The OCEM releases eleven clocks every time the debugger reads the OCORE_DRD register.

2.4.3.2 Data Memory Read Using Vector Load Command

In order to read a data memory location with the vector load command, some prerequisites are needed

The debugger performs the following sequence:

Example 2-6. Data Memory Read using Vector Load Command

```
vld (r1.ui).di8[+pm], v0.di8          ; load and post increment data from r1 to v0
nop
nop
nop
br 20
nop
nop
nop
nop
```

Nine clock cycles are applied in order to enable the loaded data to reach the via0 vector.

After performing this sequence, the debugger can read via0 using the vec_in_rd JTAG scan chain (0x93).

The VRD_VALID bit in the vex_in_rd chain is set each time that the vector register is written.

Once the vector register is read by the debugger, the VRD_VALID is cleared.

The debugger determines whether the transaction was successful according to the value of the VRD_VALID bit (a high value indicates a successful transaction). If the transaction is not successful, the debugger re-reads the data.

When reading a continuous section in memory, the address is loaded only once, and the post-modification mechanism of the core is used in order to increment the address. In this case, the instruction sequence is loaded only once. Afterwards, the debugger has to read only the vector register and VRD_VALID indication in the successive transactions.

2.4.4 Data Memory Write

Data memory can be write using the core or using a dedicated scan chain.

2.4.4.1 Data Memory Write Using Store Command

In order to write a data memory location, the debugger loads the desired address to r1 pointer and performs the following sequence:

Example 2-7. Data Memory Read using Vector Load Command

```
mov #imm, r0.1                    ; move the required data to r0
nop
st r0.ud, (r1.ui).di[+pm] ; store the value at address written to r1
nop
nop
nop
br 20
nop
nop
nop
nop
```

Ten clock cycles are applied in order to enable the data to reach the IDM.

When writing a continuous section in memory, the address is loaded only once, and the post-modification mechanism of the core is used in order to increment the address. In this case, the instruction sequence is loaded only once. Afterwards, the debugger has to write only the 32-bit immediate in the successive transactions (using the 32-bit immediate scan chain, inst_imm_scan).

The OCEM supports automatic clock insertion for this transaction. In order to use this mode, the number of clocks (0xA) is loaded to bits 27:24 (AUTO_CLK_NUM field) of the OCM_CONTROL register, and bit 21 (AUTO_CLK field) of the OCM_CONTROL register should be set. The OCEM releases ten clocks every time the debugger writes the 32-bit immediate.

2.4.5 Reading IO Memory

The following sequence is performed in order to read the IO memory:

Example 2-8. Read IO Memory

```
in {[cpm]} (r1.ui).di, r0.di   ;load data from cpm to r0
nop
modr (r1.ui).di+#4               ;post increment r1
nop
nop
nop
nop
nop
nop
out{cpm} r0.di ,(#0x1FC).di            ;write data at r0 to OCORE_DRD
br 0x20
nop
nop
nop
nop
```

Thirteen clock cycles are applied in order to enable the data to reach the OCORE_DRD.

2.4.5.1 Writing IO memory

The following sequence is performed in order to write the IO memory:

Example 2-9. Writing IO Memory

```
mov #imm,r0.i                ; load data to r0
nop
nop
nop
out {cpm} r0.di ,(r1.ui).di        ; write data at r0 to address at r1
modr (r1.ui).di+#4               ; post increment r1
br 0x20
nop
nop
nop
nop
```

Ten clock cycles are applied in order to reach the OF3 stage of the br instruction.

2.5 Breakpoint Generation

The OCEM supports the following breakpoints:

- Two program address breakpoints
- Data address breakpoint
- Data value match breakpoint (on memory transactions)
- Combined data address and data match breakpoint
- Data address DMA breakpoint
- Two external breakpoint requests

Control registers in the OCEM determine which breakpoints are enabled – each breakpoint has a dedicated bit in the OCM_SA_BP_EN register (all breakpoints can be enabled at the same time).

If a breakpoint is enabled, it activates a breakpoint request when the corresponding event is detected. When the core receives a breakpoint request, the OCM_SA_BP_EN register reflects the source of the breakpoint.

The following sections elaborate on each type of breakpoint.

2.5.1 Program Address Breakpoints

The OCEM supports two program address breakpoints configured using the OCM_PADD1 register and OCM_PADD2 register. Each breakpoint address is coupled with a 16-bit counter. The counter is incremented every time the address configured at the corresponding program address breakpoint configuration (OCM_PADD1 or OCM_PADD2) is accessed by the core. Once the counter reaches the value configured at OCM_PCOUNT1 register or OCM_PCOUNT2 (according to the address breakpoint configuration used) the counter is reset and a breakpoint request is activated.

Each program address breakpoint can be enabled or disabled using the SA_PA field at the OCM_SA_BP_EN register.

2.5.2 Data Address Breakpoints

This breakpoint is configured by writing to two registers: OCM_DADD_LOW, which defines the lower boundaries of the data address range, and OCM_DADD_HIGH, which defines the upper boundaries of the data address range. A breakpoint is detected if the data address is inside the defined range. A specific address can be defined by loading the desired address into both range registers. The SA_DA_RD and SA_DA_WR fields in the OCM_SA_BP_EN register define whether a match is considered for read, write or both.

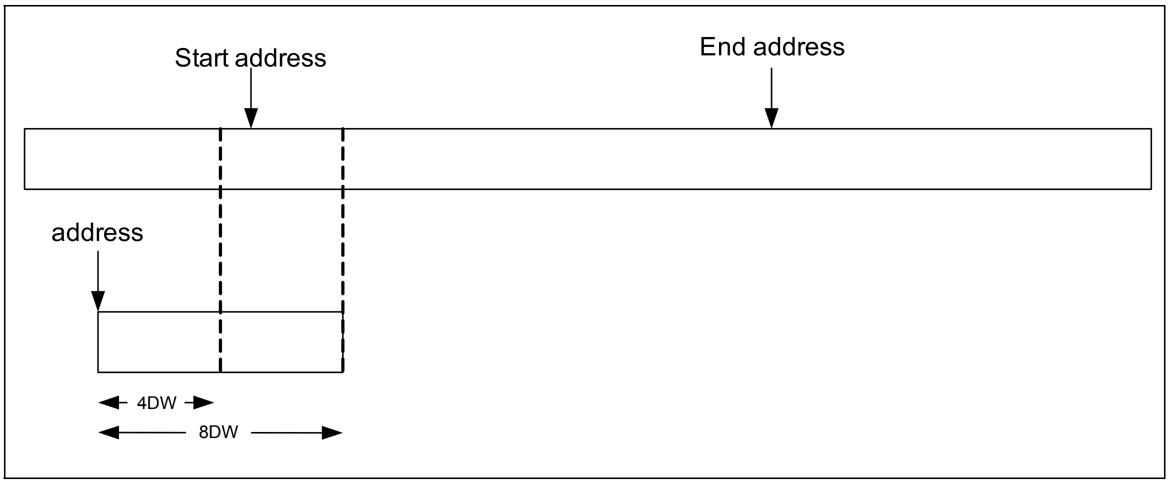
When the CO_A_EN bit in the OCM_SA_BP_EN register is set, this breakpoint is coupled with a 16-bit counter. This breakpoint counter is incremented for every data address match. The breakpoint request is activated only when the internal OCEM counter reaches the value configured at the OCM_ACOUNT register.

The core supports several widths of data transactions (from byte up to 16 DW). When performing a memory access with width greater than byte (for example, DW), consecutive memory locations are accessed. The core issues the base address of the transaction and the access width. The data address breakpoint covers all accessed addresses; therefore, the access width is also taken into consideration when triggering a breakpoint.

This breakpoint is supported for both aligned and unaligned data accesses.

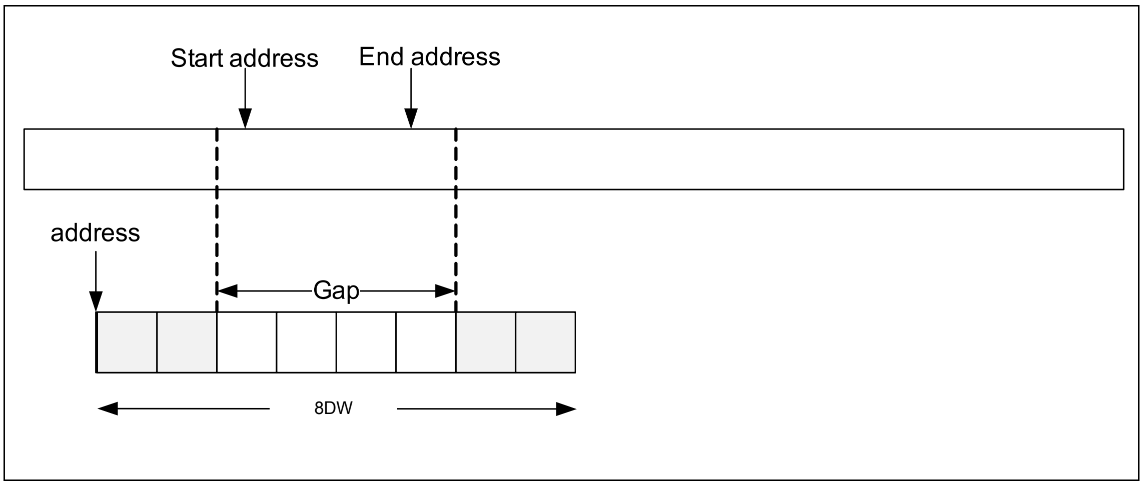
The following example demonstrates two scenarios. Start and end address are loaded into the range registers. In the first scenario, the access width is 4dw. In this case, the breakpoint is not triggered. In the second scenario, the same base address is accessed with an 8dw access width. In this case, the breakpoint is triggered because addresses inside the range were accessed.

Example 2-10. Data Address Breakpoint



The example below shows a vector memory access where stride is 8dw. In this case, the breakpoint range overlaps the vector stride. In this case, the breakpoint will not be triggered, as the breakpoint range is not accessed.

Example 2-11. Data Address Breakpoint with Stride



Note: When performing parallel load or store accesses, both accesses are considered as matches on the data address match counter.

In case of a row stride memory access, the breakpoint will only be triggered by the memory rows and banks that were actually accessed. The gaps between VPU accesses will be ignored.

2.5.3 Data Value Match Breakpoints

The OCEM includes a data value match OCM_DVM register. This 32-bit register contains the data to be matched if the SA_DVM field in the OCM_SA_BP_EN register is set. The data width to be detected (byte, word or dw) can be configured using the DVM_TYPE field in the OCM_CONTROL register. The OCEM compares the write and read data buses of the core to the DVM register. If the CO_D_EN field in the OCM_SA_BP_EN register is set the breakpoint is coupled with an internal 16bit counter. The breakpoint request is activated only when the internal counter in the OCEM reaches the count value configured at OCM_DCOUNT register.

The OCEM looks for the match data from LS0/1, as follows:

- If the DVM register is configured to byte, the match data is in all possible byte locations.
- If the DVM register is configured to word, the match data is in all possible word-aligned locations.
- If the DVM register is configured to DW, the match data is in all possible DW-aligned locations.

When a match is detected, the OCEM activates the breakpoint request.

2.5.4 Combined Address and Data Breakpoints

The OCEM can be configured to generate a breakpoint request for a simultaneous match of both address and value. The SA_COMB_RD and SA_COMB_WR fields at the OCM_SA_BP_EN register are used to define a combined address and data breakpoint for read and write transactions. If the CO_A_EN field at the OCM_SA_BP_EN register is set, this breakpoint is coupled with a 16-bit counter. This breakpoint counter is incremented for every combined match. The breakpoint request is activated only when the internal OCEM counter reaches the value configured at the OCM_ACOUNT register.

2.5.5 DMA Breakpoints

The data DMA breakpoint supports a single data address match on DMA memory access. The address match refers only to the base address; thus, access width is not taken into account. Breakpoints can be enabled for DMA upload, DMA download or both. In addition, the user can specify whether the monitored address is internal or external.

For further information, refer to the DMSS programming model.

2.5.6 External Breakpoints

Two external asynchronous breakpoint requests EXT_BP1_REQ, EXT_BP2_REQ are supported. External breakpoints are enabled by setting the SA_EXT1, SA_EXT2 fields in the OCM_SA_BP_EN register. The external breakpoint request must remain asserted until the OCEM asserts acknowledge (CEVA_OEXT1_ACK, CEVA_OEXT2_ACK) indicating that the breakpoint has been accepted. The external breakpoint status can be read from the SA_EXT_STS field in the OCM_SA_BP_ST register.

2.5.7 Breakpoint Latency

Table 2-1 specifies the cycle latency for the acceptance of each breakpoint type. Each number represents the minimum latency, and the core is assumed to be in an interruptible state. If the core is in a non-interruptible state and a breakpoint is requested, it is accepted as soon as the core exits the non-interruptible state. For example, if the breakpoint request occurs during a multi-cycle instruction or a delay slot, the breakpoint is accepted after the multi-cycle and/or all of the delay slots are executed.

Table 2-1. Breakpoint Acceptance Latency

Breakpoint Type	Latency
Data value match breakpoint (SC write)	9
Data value match breakpoint (VPU write)	10
Data value match breakpoint (SC read)	7
Data value match breakpoint (VPU read)	7
Data address breakpoints	6
Combined breakpoint (SC write)	9
Combined breakpoint (VPU write)	11
Combined breakpoint (SC read)	7
Combined breakpoint (VPU read)	7
Program address breakpoints	0
DMA match breakpoint	1

3 Profiler

The Profiler is an optional block inside the CEVA-XM4. The Profiler is accessible via the I/O protocol, and is responsible for collecting profiling data and statistics used for analyzing software. The profiler counts events that occurred during software routine. These events include: read after write scenarios, data memory block conflicts, program cache misses and more. The profiler counters can be paused, cleared and started via the debugger. The profiler is available only when configuring the OCEM to enhanced debug mode.

3.1 Profiler Functionality

The profiler block has a group of nine counters. The first counter is a fixed counter, FRCC, while the remaining eight counters are configurable and can be set to count one event out of the list provided below. The profiler registers in Table 5-58 can be accessed by in/out instructions. The counters are controlled using a set of registers described in detail in the following sections.

3.1.1 Event Descriptions

Table 3-1 describes the profiled events that can be selected for counting in the profilers' counters 0 thru 7 with the exception of FRCC which has a dedicated counter.

Table 3-1. Profiler Event Description

Event Name	Description
FRCC	Free Running Clock Counter. Counts the number of core cycles
WAIT_CNT	Wait counter. Counts the number of wait cycles asserted by any source to the Core.
VPU_STL_CNT	VPU stall counter Counts the number of stall VPU signals due to read after write.
DBLK_CONF_CNT	DMSS block conflict counter. Counts the number of conflicts between the core, write-buffer and the EDAP performed on DMSS.
OSFC	Output stage full counter. Counts the number of wait caused by write buffer full.
TDRC	TCM DMSS read counter. Counts the number of TCM accesses due to any kind of a read transaction (LSU0/1, EDAP and DMA).Aborted reads are not counted.
TDWC	TCM DMSS write counter. Counts the number of TCM accesses due to any kind of a write transaction (LSU0, EDAP and DMA). Aborted writes are not counted.

Table 3-1. Profiler Event Description (Continued)

Event Name	Description
ERWC	EDP read wait counter. Counts the number of cycles for which the core is held in wait due to an external data memory read access.
VPU0_NOP_INST	VPU0 nop instruction. Counts the number of nop instructions in VPU0 unit. If the nop instruction remains additional cycles in the instruction register (due to stall or wait indications), the extra cycles are not counted.
VPU1_NOP_INST	VPU1 nop instruction. Counts the number of nop instructions in VPU1. If the nop instruction remains additional cycles in the instruction register (due to stall or wait indications), the extra cycles are not counted.
PCU_NOP_INST	PCU nop instruction. Counts the number of nop instructions in PCU. If the nop instruction remains additional cycles in the instruction register (due to stall or wait indications), the extra cycles are not counted.
SPU0_NOP_INST	SPU0 nop instruction. Counts the number of nop instructions in SPU0. If the nop instruction remains additional cycles in the instruction register (due to stall or wait indications), the extra cycles are not counted.
SPU1_NOP_INST	SPU1 nop instruction. Counts the number of nop instructions in SPU1. If the nop instruction remains additional cycles in the instruction register (due to stall or wait indications), the extra cycles are not counted.
SPU2_NOP_INST	SPU2 nop instruction. Counts the number of nop instructions in SPU2. If the nop instruction remains additional cycles in the instruction register (due to stall or wait indications), the extra cycles are not counted.
SPU3_NOP_INST	SPU3 nop instruction. Counts the number of nop instructions in SPU3. If the nop instruction remains additional cycles in the instruction register (due to stall or wait indications), the extra cycles are not counted.
LSU0_NOP_INST	LSU0 nop instruction. Counts the number of nop instructions in LSU0. If the nop instruction remains additional cycles in the instruction register (due to stall or wait indications), the extra cycles are not counted.
LSU1_NOP_INST	LSU1 nop instruction. Counts the number of nop instructions in LS1 unit. If the nop instruction remains additional cycles in the instruction register (due to stall or wait indications), the extra cycles are not counted.
PMSS_HIT_CNT	PMSS hit counter. Counts the number of pmss cache hits.

Table 3-1. Profiler Event Description (Continued)

Event Name	Description
PMSS_MIS_CNT	PMSS miss counter. Counts the number of pmss cache misses.
PWCWC	Parallel write contention wait counter. Counts the number of wait cycles caused by parallel write contention.
P_ADD_COUNTX	PC address range counter X Counts the number of times the pc was between P_LOW_ADD and P_HI_ADD * X is 0 to 7

3.1.2 Controlling the Profiles' Counters

The Profiler contains the following control registers:

- PROF_RESET Register (CPM Address 0x300)
- PROF_PAUSE Register (CPM Address 0x304)

By default, all the counters are paused. Each counter operation is enabled when the appropriate pause bit in is cleared and disabled (paused) when the bit is set.

- PROF_CTRL0 Register (CPM Address 0x308)
- PROF_CTRL1 Register (CPM Address 0x30C)

These register control the counters' behavior and event association. The registers contain a field per counter, 0 thru 7, to select the event for counting.

- P_LOW_ADDX Register (CPM Address 0x340-0x35C)
- P_HI_ADDX Register (CPM Address 0x360-0x37C)

These register control the address range to be counted for P_ADD_COUNTX events.

3.2 Profiler Counters under Emulation

While the core is in debug mode (OCEM is in control of the core), all of the Profiler's counters are stopped in order to avoid corruption of profiler results. Although profiler registers are stopped, it is still possible to pause, un-pause or reset any counter in the profiler.

4 OCEM Interfaces

4.1 APB Slave Interface

The CoreSight, OCEM IO registers, RTT registers and OCEM chains can be accessed using a synchronous APB Slave port. The APB slave-port functionality is described in the following sections.When accessing CoreSight registers the APB address is the address offset listed in Table 6.1. When accessing to other registers the APB address is 0xE00+offset[7:0] where offset is the address offset listed in Table 5.1, Table 5-75 and Table 5-100.

Figure 4-1 describe the APB Slave port.

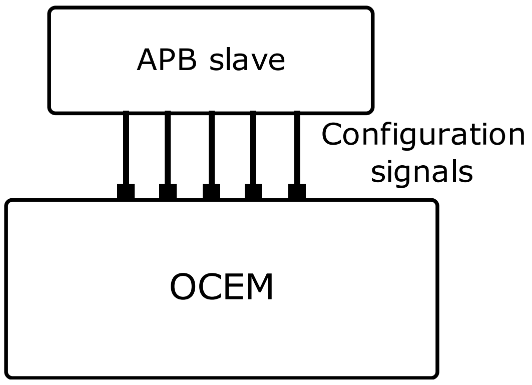


Figure 4-1. APB Slave

4.1.1 APB Slave Inputs

Table 4-1. APB Slave Inputs

Signal Name	Size	Description
System Inputs		
s_pwrite	1	APB Slave pwrite signal
s_psel	1	APB Slave psel signal
s_penable	1	APB Slave penable signal
s_paddr	10	APB Slave paddr signal
s_pwdata	32	APB Slave pwdata signal

4.1.2 APB Slave Outputs

Table 4-2. APB Slave Outputs

Signal Name	Size	Description
System outputs		
s_prdata	32	APB Slave prdata signal
s_pready	1	APB Slave port ready signal
s_pslverr	1	Indicates an error condition on an APB transfer. Error conditions include: Accessing undefined address space.

4.1.3 Accessing the OCEM chains

The dedicated OCEM scan chains are accessed using the SCCO and SCDA registers.

The registers listed in Table 5-1, Table 5-75 and registers OCM_ACOUNT and OCM_DCOUNT have direct access through the APB-Slave interface and are not accessed through the SCCO and SCDA registers.
For more details on accessing the OCEM chain, refer to section 1.3.2 Accessing the OCEM Chain.

4.2 OCEM External Interfaces

Table 4-3. CEVA-XM4 OCEM Input Signals

Signal Name	Size	Description
EXT_BP1_REQ	1	External breakpoint request #1
EXT_BP2_REQ	1	External breakpoint request #2
EXT_PMEM_WR	1	External program memory write indication
BS_REG_TDO	1	Boundary scan register Test Data Out signal
TCK	1	JTAG clock

Table 4-3. CEVA-XM4 OCEM Input Signals (Continued)

Signal Name	Size	Description
TMS	1	JTAG state machine control signal
TDI	1	JTAG protocol test data-in
JT_AP	1	Use JTAG or APB to set the chains 0: use the JTAG interface 1: use the APB slave port

Table 4-4. CEVA-XM4 OCEM Output Signals

Signal Name	Size	Description
CEVA_TDO	1	JTAG protocol test data-out
CEVA_TDO_OEN	1	JTAG protocol test data output enable signal
CEVA_RTCK	1	Return Test Clock (tck which is synchronized to ceva_clk)
CEVA_JTAG_STATE	4	JTAG state
CEVA_GP_OUT	4	OCEM general purpose outputs
CEVA_OEXT1_ACK	1	Acknowledge signal for breakpoint #1
CEVA_OEXT2_ACK	1	Acknowledge signal for breakpoint #2
OCM_CORE_RST	1	The OCEM reset signal to the core, active high
OCM_MSS_RST	1	The OCEM reset signal to the Memory sub-system, active high
CEVA_OCM_DEBUG	1	OCEM debug mode indication

5 Memory Subsystem Programming Model

5.1 High-level Address Mapping of the Programming Model

Programming Model Sections	External Port Slave CPM Address Mapping	Dedicated I/O CPM Address Mapping
OCEM	0x00400104 - 0x0040017c	0x00000104 - 0x0000017c
Wrapper	0x00400180 - 0x004001ac	0x00000180 - 0x000001ac
OCEM_BPCOUNT_DRD	0x004001f0 - 0x004001fc	0x000001f0 - 0x000001fc
PROFILER	0x00400300 - 0x0040037c	0x00000300 - 0x0000037c

5.2 OCEM

Table 5-1: OCEM Programming Model

Name	Offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCM_DVM	0x104	OCM_DVM																															
OCM_OFIO7P	0x108	OCM_OFIO7P																															
OCM_OFIO8P	0x10c	OCM_OFIO8P																															
OCM_PADD1	0x110	OCM_PADD1																															
OCM_PADD2	0x114	OCM_PADD2																															
OCM_OFIO1	0x118	OCM_OFIO1																															
OCM_OFIO2	0x11c	OCM_OFIO2																															
OCM_PCOUNT1	0x120	reserved																OCM_PCOUNT1															
OCM_PCOUNT2	0x124	reserved																OCM_PCOUNT2															
OCM_OFIO3	0x128	OCM_OFIO3																															
OCM_OFIO4	0x12c	OCM_OFIO4																															

Name	**Offset**	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OCM_DADD_LOW	0x130	OCM_DADD_LOW																															
CEVAX_D3_PC	0x134	CEVA_D3_PC																															
OCM_OFIO5	0x138	OCM_OFIO5																															
OCM_OFIO6	0x13c	OCM_OFIO6																															
OCM_DADD_HIGH	0x140	OCM_DADD_HIGH																															
OCM_OFIO7	0x148	OCM_OFIO7																															
OCM_OFIO8	0x14c	OCM_OFIO8																															
OCM_CONTROL	0x150	reserved	AXI_S_DS	EDAP_DS	reserved	AUTO_CLK_NUM				BOOT_MASK	reserved	AUTO_CLK	RST_STATUS	reserved		DVM_TYPE		GP_OUT				reserved				PADD_INC	STEP	STOP_GO	TRAPE_EN	MSS_RST	CORE_RST	BP_DETECT_EN	DEBUG_ENABLE
OCM_SA_BP_EN	0x154	reserved												SA_COMB_WR	SA_COMB_RD	reserved	SA_DVM	reserved	CO_D_EN	CO_A_EN	SA_DA_WR	reserved			SA_DA_RD	reserved		SA_PA		DMA_DA_E	reserved	SA_EXT	
MSS_CONFIG	0x15c	reserved																												NUM_QMAN			

Name	Offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCM_STATUS	0x160	reserved													WAIT	reserved			OCEM_WAIT	DMSS_IDLE_OCEM	reserved			PMSS_IDLE	TRAPE_INST	PACTIVE_TRANS	CLK_APPLIED	Q_FLUSH	reserved	PCACHE_WR	PMEM_WR	RST_DUR_DBG	DBG_MODE
OCM_SA_BP_ST	0x164	reserved												SA_COMB_WR_ST	SA_COMB_RD_ST	reserved	SA_DVM_ST	reserved			SA_DA_WR_ST	reserved			SA_DA_RD_ST	reserved		SA_PA_ST		DMA_DA_ST	reserved	SA_EXT_ST	
AXIM_CONFIG	0x168	reserved																														AXIM_WIDTH	
AXIS_CONFIG	0x16c	reserved																					AXIS2_WIDTH			reserved	AXIS1_WIDTH			reserved	AXIS0_WIDTH		
CORE_VERSION	0x174	DSP_CORE_TYPE																DSP_RTL_VERSION												DSP_RTL_REVISION			
CORE_ID	0x178	CORE_ID																															
CORE_CONFIG	0x17c	reserved									COMPATIBILITY	BUS_ECC	ECC	ETM		PCAC_SZE		DMSS_BLK	DMEM_TCM_SIZE			reserved			NON_LINEAR_FUNC_SUPPORT	PMEM_TCM_SIZE			MEM_PWR	VFLP	FLP	VPU_XTEND	SPU_XTEND

5.2.1 OCM_DVM

Table 5-2: OCM_DVM (CPM Address 0x104)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCM_DVM																															

Table 5-3: OCM_DVM Field Description

Field	Bits/Location	R/W	Description
OCM_DVM	31:0	R/W	Data value match register (Enhanced mode only): Increments breakpoint counters when stored data matches this value. This register is coupled with the OCM_DCOUNT register.

Reset Value: 0x00000000

5.2.2 OCM_OFIO7P

Table 5-4: OCM_OFIO7P (CPM Address 0x108)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCM_OFIO7P																															

Table 5-5: OCM_OFIO7P Field Description

Field	Bits/Location	R/W	Description
OCM_OFIO7P	31:0	RO	File I/O #7 register A shadow register for read only access to the OCM_OFIO7 Register (offset 0x148). This register is mainly used when the host wishes to read the OCM_OFIO7 register through the JTAG scan chain without changing its value (read only).

Reset Value: 0x00000000

5.2.3 OCM_OFIO8P

Table 5-6: OCM_OFIO8P (CPM Address 0x10c)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCM_OFIO8P																															

Table 5-7: OCM_OFIO8P Field Description

Field	Bits/Location	R/W	Description
OCM_OFIO8P	31:0	RO	File I/O #8 register A shadow register for read only access to the OCM_OFIO8 Register (offset 0x14C). This register is mainly used when the host wishes to read the OCM_OFIO8 register through the JTAG scan chain without changing its value (read only).

Reset Value: 0x00000000

5.2.4 OCM_PADD1

Table 5-8: OCM_PADD1 (CPM Address 0x110)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCM_PADD1																															

Table 5-9: OCM_PADD1 Field Description

Field	Bits/Location	R/W	Description
OCM_PADD1	31:0	R/W	Program address breakpoint #1 value: Increments the breakpoint counter when program counter matches this value. This register is coupled with the OCM_PCOUNT1 register.

Reset Value: 0x00000000

5.2.5 OCM_PADD2

Table 5-10: OCM_PADD2 (CPM Address 0x114)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCM_PADD2																															

Table 5-11: OCM_PADD2 Field Description

Field	Bits/Location	R/W	Description
OCM_PADD2	31:0	R/W	Program address breakpoint #2 value: Increments the breakpoint counter when program counter matches this value. This register is coupled with the OCM_PCOUNT2 register.

Reset Value: 0x00000000

5.2.6 OCM_OFIO1

Table 5-12: OCM_OFIO1 (CPM Address 0x118)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCM_OFIO1																															

Table 5-13: OCM_OFIO1 Field Description

Field	Bits/Location	R/W	Description
OCM_OFIO1	31:0	R/W	File I/O #1 register

Reset Value: 0x00000000

5.2.7 OCM_OFIO2

Table 5-14: OCM_OFIO2 (CPM Address 0x11c)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCM_OFIO2																															

Table 5-15: OCM_OFIO2 Field Description

Field	Bits/Location	R/W	Description
OCM_OFIO2	31:0	R/W	File I/O #2 register

Reset Value: 0x00000000

5.2.8 OCM_PCOUNT1

Table 5-16: OCM_PCOUNT1 (CPM Address 0x120)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																OCM_PCOUNT1															

Table 5-17: OCM_PCOUNT1 Field Description

Field	Bits/Location	R/W	Description
OCM_PCOUNT1	15:0	R/W	Program address breakpoint #1 count value: Triggers a breakpoint and is cleared once the OCM_PADD1 breakpoint counter is equal to this value.
reserved	31:16		

Reset Value: 0x00000001

5.2.9 OCM_PCOUNT2

Table 5-18: OCM_PCOUNT2 (CPM Address 0x124)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																OCM_PCOUNT2															

Table 5-19: OCM_PCOUNT2 Field Description

Field	Bits/Location	R/W	Description
OCM_PCOUNT2	15:0	R/W	Program address breakpoint #2 count value: Triggers a breakpoint and is cleared once the OCM_PADD2 breakpoint counter is equal to this value.
reserved	31:16		

Reset Value: 0x00000001

5.2.10 OCM_OFIO3

Table 5-20: OCM_OFIO3 (CPM Address 0x128)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCM_OFIO3																															

Table 5-21: OCM_OFIO3 Field Description

Field	Bits/Location	R/W	Description
OCM_OFIO3	31:0	R/W	File I/O #3 register

Reset Value: 0x00000000

5.2.11 OCM_OFIO4

Table 5-22: OCM_OFIO4 (CPM Address 0x12c)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCM_OFIO4																															

Table 5-23: OCM_OFIO4 Field Description

Field	Bits/Location	R/W	Description
OCM_OFIO4	31:0	R/W	File I/O #4 register

Reset Value: 0x00000000

5.2.12 OCM_DADD_LOW

Table 5-24: OCM_DADD_LOW (CPM Address 0x130)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCM_DADD_LOW																															

Table 5-25: OCM_DADD_LOW Field Description

Field	Bits/Location	R/W	Description
OCM_DADD_LOW	31:0	R/W	Data address breakpoint lower address: This register represents the lower boundary of the data address range.

Reset Value: 0x00000000

5.2.13 CEVAX_D3_PC

Table 5-26: CEVAX_D3_PC (CPM Address 0x134)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEVA_D3_PC																															

Table 5-27: CEVAX_D3_PC Field Description

Field	Bits/Location	R/W	Description
CEVA_D3_PC	31:0	RO	PC of the D3 stage This register is read only

Reset Value: 0x00000000

5.2.14 OCM_OFIO5

Table 5-28: OCM_OFIO5 (CPM Address 0x138)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCM_OFIO5																															

Table 5-29: OCM_OFIO5 Field Description

Field	Bits/Location	R/W	Description
OCM_OFIO5	31:0	R/W	File I/O #5 register

Reset Value: 0x00000000

5.2.15 OCM_OFIO6

Table 5-30: OCM_OFIO6 (CPM Address 0x13c)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCM_OFIO6																															

Table 5-31: OCM_OFIO6 Field Description

Field	Bits/Location	R/W	Description
OCM_OFIO6	31:0	R/W	File I/O #6 register

Reset Value: 0x00000000

5.2.16 OCM_DADD_HIGH

Table 5-32: OCM_DADD_HIGH (CPM Address 0x140)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCM_DADD_HIGH																															

Table 5-33: OCM_DADD_HIGH Field Description

Field	Bits/Location	R/W	Description
OCM_DADD_HIGH	31:0	R/W	Data address breakpoint higher address: This register represents the upper boundary of the data address range.

Reset Value: 0x00000000

5.2.17 OCM_OFIO7

Table 5-34: OCM_OFIO7 (CPM Address 0x148)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCM_OFIO7																															

Table 5-35: OCM_OFIO7 Field Description

Field	Bits/Location	R/W	Description
OCM_OFIO7	31:0	R/W	File I/O #7 register

Reset Value: 0x00000000

5.2.18 OCM_OFIO8

Table 5-36: OCM_OFIO8 (CPM Address 0x14c)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCM_OFIO8																															

Table 5-37: OCM_OFIO8 Field Description

Field	Bits/Location	R/W	Description
OCM_OFIO8	31:0	R/W	File I/O #8 register

Reset Value: 0x00000000

5.2.19 OCM_CONTROL

Table 5-38: OCM_CONTROL (CPM Address 0x150)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved	AXI_S_DS	EDAP_DS	reserved	AUTO_CLK_NUM				BOOT_MASK	reserved	AUTO_CLK	RST_STATUS	reserved		DVM_TYPE		GP_OUT				reserved				PADD_INC	STEP	STOP_GO	TRAPE_EN	MSS_RST	CORE_RST	BP_DETECT_EN	DEBUG_ENABLE

Table 5-39: OCM_CONTROL Field Description

Field	Bits/Location	R/W	Description
DEBUG_ENABLE	0:0	R/W	When set, enables the OCEM to enter debug mode (i.e., activate the wait signal and the debug indication)
BP_DETECT_EN	1:1	R/W	When set, enables breakpoint detection (enables the FFs that sample the core interface buses)
CORE_RST	2:2	WO	When set, activates a core reset
MSS_RST	3:3	WO	When set, activates a reset to the Memory Sub System
TRAPE_EN	4:4	R/W	When set, enables the OCEM to enter debug mode for trape instructions
STOP_GO	5:5	WO	stop/go: setting this bit causes the OCEM to generate a breakpoint signal to the core. This is effective only during normal operation. When in debug mode, clearing this bit resumes normal operation, i.e. exits debug mode. Writing one to this bit while in debug mode has no effect.
STEP	6:6	WO	Single step: setting this bit activates single stepping

Field	Bits/Location	R/W	Description
PADD_INC	7:7	R/W	Program address increment enable: enables the address auto increment mode for program memory access.
reserved	11:8		
GP_OUT	15:12	R/W	General purpose outputs
DVM_TYPE	17:16	R/W	Width of the data value inserted in the DVM: 2b00: byte 2b01: word 2b10: double word
reserved	19:18		
RST_STATUS	20:20	WO	When written with 1, clears all the status bits
AUTO_CLK	21:21	R/W	Auto clock mode enable bit (when set, enables auto clock mode)
reserved	22:22		
BOOT_MASK	23:23	R/W	Control the boot wake up after OCEM reset the core
AUTO_CLK_NUM	27:24	R/W	The number of clocks to apply during auto-clock mode
reserved	28:28		
EDAP_DS	29:29	R/W	Enable and disable all EDAP port accesses 0: EDAP port accesses are enabled 1: EDAP port accesses are disabled
AXI_S_DS	30:30	R/W	Enable and disable all AXI Slaves access to the core 0: AXI Slave ports accesses are enabled 1: AXI Slave ports accesses are disabled All AXI slave ports are affected by this field.
reserved	31:31		

Reset Value: 0x00000000

5.2.20 OCM_SA_BP_EN

Table 5-40: OCM_SA_BP_EN (CPM Address 0x154)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved												SA_COMB_WR	SA_COMB_RD	reserved	SA_DVM	reserved	CO_D_EN	CO_A_EN	SA_DA_WR	reserved			SA_DA_RD	reserved		SA_PA		DMA_DA_E	reserved	SA_EXT	

Table 5-41: OCM_SA_BP_EN Field Description

Field	Bits/Location	R/W	Description
SA_EXT	1:0	R/W	Enable breakpoint for external request 4-1 respectively: Bit1: SA_EXT2 Bit0: SA_EXT1
reserved	2:2		
DMA_DA_E	3:3	R/W	Enable breakpoint for DMA request

Field	Bits/Location	R/W	Description
SA_PA	5:4	R/W	Enable program address breakpoints 2-1 respectively: Bit5: SA_PA2 Bit4: SA_PA1
reserved	7:6		
SA_DA_RD	8:8	R/W	Enable data address breakpoint for read access
reserved	11:9		
SA_DA_WR	12:12	R/W	Enable data address breakpoint for write access
CO_A_EN	13:13	R/W	Enable address breakpoint counter
CO_D_EN	14:14	R/W	Enable data breakpoint counter
reserved	15:15		
SA_DVM	16:16	R/W	Enable data value match breakpoint (Enhanced mode only)
reserved	17:17		
SA_COMB_RD	18:18	R/W	Enable combined data address and data value match BP, read access (Enhanced mode only)
SA_COMB_WR	19:19	R/W	Enable combined data address and data value match BP, write access (Enhanced mode only)
reserved	31:20		

Reset Value: 0x00000000

Note:

For all breakpoints: 0: breakpoint is disabled, 1: breakpoint enabled.

When enabling read and write breakpoints together the breakpoint counter will increment for both accesses.

5.2.21 MSS_CONFIG

Table 5-42: MSS_CONFIG (CPM Address 0x15c)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																												NUM_QMAN			

Table 5-43: MSS_CONFIG Field Description

Field	Bits/Location	R/W	Description
NUM_QMAN	3:0	RO	Number of available hardware Queue Managers 0100 8 QMANs All other values reserved.
reserved	31:4		

Reset Value: 0x00000008

5.2.22 OCM_STATUS

Table 5-44: OCM_STATUS (CPM Address 0x160)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved													WAIT	reserved			OCEM_WAIT	DMSS_IDLE_OCEM	reserved			PMSS_IDLE	TRAPE_INST	PACTIVE_TRANS	CLK_APPLIED	Q_FLUSH	reserved	PCACHE_WR	PMEM_WR	RST_DUR_DBG	DBG_MODE

Table 5-45: OCM_STATUS Field Description

Field	Bits/Location	R/W	Description
DBG_MODE	0:0	RO	Indicates that the OCEM is in debug mode (a breakpoint was accepted)
RST_DUR_DBG	1:1	RO	Indicates that there was a user reset during the debug session Important: see the note below the table
PMEM_WR	2:2	RO	Indicates that the internal program memory was written while the core was not in debug. OCEM writes will not be reflects in this field.
PCACHE_WR	3:3	RO	Indicates that the internal program cache was written while the core was not in debug. OCEM writes will not be reflects in this field.
reserved	4:4		
Q_FLUSH	5:5	RO	Indicates that the queue is empty (flushed)
CLK_APPLIED	6:6	RO	Indicates that the total number of clock cycles that the debugger requested were applied by the OCEM
PACTIVE_TRANS	7:7	RO	External program memory active-transaction indication
TRAPE_INST	8:8	RO	Debug mode was entered due to a trape instruction
PMSS_IDLE	9:9	RO	The Program external port is not occupied by the PMSS, • EPP has no pending read or write transactions • SWOP has no pending transactions • RAB has no pending transactions • PDMA has no pending transactions
reserved	12:10		
DMSS_IDLE_OCEM	13:13	RO	• Indicates all memory access are complete and memory access can be performed using the OCEM. • This bit is set when: • Write buffer is empty • EDP has no pending read or write transactions • AXIM0 has no pending read or write transactions • AXIM1 has no pending read or write transactions • EPP has no pending read or write transactions
OCEM_WAIT	14:14	RO	Wait initiated by the OCEM after branch to self
reserved	17:15		

Field	Bits/Location	R/W	Description
WAIT	18:18	RO	Indicates that the core is in wait
reserved	31:19		

Reset Value: 0x00000000

Note: All the status bits are cleared when the debugger reads the status register. This functionality enables the debugger to identify multiple occurrences of the events that these status bits reflect, within the same debug session. They are also cleared when the debug session ends.

5.2.23 OCM_SA_BP_ST

Table 5-46: OCM_SA_BP_ST (CPM Address 0x164)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved												SA_COMB_WR_ST	SA_COMB_RD_ST	reserved	SA_DVM_ST	reserved			SA_DA_WR_ST	reserved			SA_DA_RD_ST	reserved		SA_PA_ST		DMA_DA_ST	reserved	SA_EXT_ST	

Table 5-47: OCM_SA_BP_ST Field Description

Field	Bits/Location	R/W	Description
SA_EXT_ST	1:0	RO	Breakpoint status for external request 4-1 respectively: Bit1: SA_EXT2 Bit0: SA_EXT1
reserved	2:2		
DMA_DA_ST	3:3	RO	Status of DMA data address match breakpoint
SA_PA_ST	5:4	RO	Program address breakpoint status: Bit5: SA_PA2 Bit4: SA_PA1
reserved	7:6		
SA_DA_RD_ST	8:8	RO	Status of data address breakpoint for read access
reserved	11:9		
SA_DA_WR_ST	12:12	RO	Status of data address breakpoint for write access
reserved	15:13		
SA_DVM_ST	16:16	RO	Status of data value match breakpoint (Enhanced mode only)
reserved	17:17		
SA_COMB_RD_ST	18:18	RO	Status of combined data address and data value match breakpoint read access (Enhanced mode only)
SA_COMB_WR_ST	19:19	RO	Status of combined data address and data value match breakpoint write access (Enhanced mode only)
reserved	31:20		

Reset Value: 0x00000000

Note:

For all breakpoints: 0: breakpoint not accepted, 1: breakpoint accepted

When enabling read and write breakpoints together, only the last access will set the status bit.

5.2.24 AXIM_CONFIG

Table 5-48: AXIM_CONFIG (CPM Address 0x168)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																														AXIM_WIDTH	

Table 5-49: AXIM_CONFIG Field Description

Field	Bits/Location	R/W	Description
AXIM_WIDTH	1:0	RO	Width of AXI slave port 0 000 – ports are disabled 001 - Reserved 010 – 128bit 011 – 256 bit 100 – Reserved 101 – Reserved 110 – reserved 111 - reserved
reserved	31:2		

Reset Value: 0x00000002

5.2.25 AXIS_CONFIG

Table 5-50: AXIS_CONFIG (CPM Address 0x16c)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																					AXIS2_WIDTH			reserved	AXIS1_WIDTH			reserved	AXIS0_WIDTH		

Table 5-51: AXIS_CONFIG Field Description

Field	Bits/Location	R/W	Description
AXIS0_WIDTH	2:0	RO	Width of AXI slave port 0 000 – port is disabled 001 - Reserved 010 – 128bit 011 – 256 bit 100 – Reserved 101 – Reserved 110 – reserved 111 - reserved
reserved	3:3		

Field	Bits/Location	R/W	Description
AXIS1_WIDTH	6:4	RO	Width of AXI slave port 1 000 – port is disabled 001 - Reserved 010 – 128bit 011 – 256 bit 100 – Reserved 101 – Reserved 110 – reserved 111 - reserved
reserved	7:7		
AXIS2_WIDTH	10:8	RO	Width of AXI slave port 2 000 – port is disabled 001 - Reserved 010 – 128bit 011 – 256 bit 100 – Reserved 101 – Reserved 110 – reserved 111 - reserved
reserved	31:11		

Reset Value: 0x00000222

5.2.26 CORE_VERSION

Table 5-52: CORE_VERSION (CPM Address 0x174)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSP_CORE_TYPE																DSP_RTL_VERSION												DSP_RTL_REVISION			

Table 5-53: CORE_VERSION Field Description

Field	Bits/Location	R/W	Description
DSP_RTL_REVISION	3:0	RO	Bits [3:0]: DSP RTL revision Reset value: RTL revision Preliminary revisions contain ascending values – preliminary revision 0 contains the value 0x0, preliminary revision 1 contains the value 0x1, preliminary revision 2 contains the value 0x2, etc. The final revision contains the value 0xF.
DSP_RTL_VERSION	15:4	RO	Bits [15:4]: DSP RTL version Reset value: RTL version For example for version v2.0.0 this field contains 0x200.
DSP_CORE_TYPE	31:16	RO	Bits [31:16]: DSP core type Reset value: Core Version. For example for CEVA-XM4 this field contains 0x4210.

Reset Value: 0x0004111f

Note:

This register is read only.

This register Can be read via the OCEM chain 0x72.

For EDAP or AXISR ports the CORE_VERSION register can be accessed using address 0x400174.

5.2.27 CORE_ID

Table 5-54: CORE_ID (CPM Address 0x178)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CORE_ID																															

Table 5-55: CORE_ID Field Description

Field	Bits/Location	R/W	Description
CORE_ID	31:0	RO	CORE ID The value of this register is sampled at reset from the core_id input bus. This register is read only. For EDAP or AXISR ports the CORE_ID register can be accessed using address 0x40_0178 .

Reset Value: 0x00000000

5.2.28 CORE_CONFIG

Table 5-56: CORE_CONFIG (CPM Address 0x17c)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved									COMPATIBILITY	BUS_ECC	ECC	ETM		PCAC_SZE		DMSS_BLK	DMEM_TCM_SIZE			reserved			NON_LINEAR_FUNC_SUPPORT	PMEM_TCM_SIZE			MEM_PWR	VFLP	FLP	VPU_XTEND	SPU_XTEND

Table 5-57: CORE_CONFIG Field Description

Field	Bits/Location	R/W	Description
SPU_XTEND	0:0	RO	SPU Xtend configuration: 1 = Xtend included 0 = Xtend not included
VPU_XTEND	1:1	RO	VPU Xtend configuration: 1 = Xtend included 0 = Xtend not included
FLP	2:2	RO	Scalar Floating Point configuration: 0 = Scalar Floating Point not included 1 = Four floating point per SPU

Field	Bits/Location	R/W	Description
VFLP	3:3	RO	Vector Floating Point configuration: 0 = Vector Floating Point not included 1 = 16 floating point (Eight floating point per VPU)
MEM_PWR	4:4	RO	Use memories with power gating 1 = used 0 = not used
PMEM_TCM_SIZE	7:5	RO	Program memory TCM size 3b000 = 32kB 3b001 = 64 kB 3b010 = 128 kB 3b011 = 256 kB
NON_LINEAR_FUNC_SUPPORT	8:8	RO	0 = No Non-linear Functions Support 1 = 32 operations (16 operations per VPU)
reserved	11:9		
DMEM_TCM_SIZE	14:12	RO	Data memory TCM size 3b000 = Reserved 3b001 = 128 kB 3b010 = Reserved 3b011 = 256 kB 3b100 = 512 kB 3b101 = 1024 kB
DMSS_BLK	15:15	RO	Number of DMSS blocks 0 = 4 blocks 1 = Reserved
PCAC_SZE	17:16	RO	Instruction cache memory size: 00: 32KB cache 01: 64KB cache 10: 128KB cache 11: Reserved
ETM	19:18	RO	ETM configuration trace 2b00 = No ETM 2b01 = Internal ETM 2b10 = External ETM
ECC	20:20	RO	Error Correction Code (ECC) configuration: 0 = disabled 1 = enabled with 32-bit ECC
BUS_ECC	21:21	RO	AXI Bus Error Correction Code (ECC) configuration: 0 = disabled 1 = enabled with 32-bit ECC
COMPATIBILITY	22:22	RO	COMPATIBILITY Configuration: 1 = support for MM3101 Vec-C compatibility 0 = No support for MM3101 Vec-C compatibility
reserved	31:23		

Reset Value: 0x0074417b

5.3 PROFILER

Table 5-58: PROFILER Programming Model

Name	Offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PROF_RESET	0x300	reserved																							PROF_CNT7_RST	PROF_CNT6_RST	PROF_CNT5_RST	PROF_CNT4_RST	PROF_CNT3_RST	PROF_CNT2_RST	PROF_CNT1_RST	PROF_CNT0_RST	FRCC_RST
PROF_PAUSE	0x304	reserved																							PROF_CNT7_PAUSE	PROF_CNT6_PAUSE	PROF_CNT5_PAUSE	PROF_CNT4_PAUSE	PROF_CNT3_PAUSE	PROF_CNT2_PAUSE	PROF_CNT1_PAUSE	PROF_CNT0_PAUSE	FRCC_PAUSE
PROF_CTRL0	0x308	reserved			PROF_CNT3_SEL					reserved			PROF_CNT2_SEL					reserved			PROF_CNT1_SEL					reserved			PROF_CNT0_SEL				
PROF_CTRL1	0x30c	reserved			PROF_CNT7_SEL					reserved			PROF_CNT6_SEL					reserved			PROF_CNT5_SEL					reserved			PROF_CNT4_SEL				
FRCC	0x310	FRCC																															
PROF_CNTx	0x320	PROF_CNTx																															
P_LOW_ADDx	0x340	P_LOW_ADD																															
P_HI_ADDx	0x360	P_HI_ADDx																															

5.3.1 PROF_RESET

Table 5-59: PROF_RESET (CPM Address 0x300)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																							PROF_CNT7_RST	PROF_CNT6_RST	PROF_CNT5_RST	PROF_CNT4_RST	PROF_CNT3_RST	PROF_CNT2_RST	PROF_CNT1_RST	PROF_CNT0_RST	FRCC_RST

Table 5-60: PROF_RESET Field Description

Field	Bits/Location	R/W	Description
FRCC_RST	0:0	WO	Free Running Clock Counter reset
PROF_CNT0_RST	1:1	WO	Profiler counter 0 reset
PROF_CNT1_RST	2:2	WO	Profiler counter 1 reset
PROF_CNT2_RST	3:3	WO	Profiler counter 2 reset
PROF_CNT3_RST	4:4	WO	Profiler counter 3 reset
PROF_CNT4_RST	5:5	WO	Profiler counter 4 reset
PROF_CNT5_RST	6:6	WO	Profiler counter 5 reset
PROF_CNT6_RST	7:7	WO	Profiler counter 6 reset
PROF_CNT7_RST	8:8	WO	Profiler counter 7 reset
reserved	31:9		

Reset Value: 0x00000000

5.3.2 PROF_PAUSE

Table 5-61: PROF_PAUSE (CPM Address 0x304)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																							PROF_CNT7_PAUSE	PROF_CNT6_PAUSE	PROF_CNT5_PAUSE	PROF_CNT4_PAUSE	PROF_CNT3_PAUSE	PROF_CNT2_PAUSE	PROF_CNT1_PAUSE	PROF_CNT0_PAUSE	FRCC_PAUSE

Table 5-62: PROF_PAUSE Field Description

Field	Bits/Location	R/W	Description
FRCC_PAUSE	0:0	R/W	Free Running Clock Counter pause
PROF_CNT0_PAUSE	1:1	R/W	Profiler counter 0 pause
PROF_CNT1_PAUSE	2:2	R/W	Profiler counter 1 pause
PROF_CNT2_PAUSE	3:3	R/W	Profiler counter 2 pause
PROF_CNT3_PAUSE	4:4	R/W	Profiler counter 3 reset

Field	Bits/Location	R/W	Description
PROF_CNT4_PAUSE	5:5	R/W	Profiler counter 4 pause
PROF_CNT5_PAUSE	6:6	R/W	Profiler counter 5 pause
PROF_CNT6_PAUSE	7:7	R/W	Profiler counter 6 pause
PROF_CNT7_PAUSE	8:8	R/W	Profiler counter 7 pause
reserved	31:9		

Reset Value: 0x000001FF

5.3.3 PROF_CTRL0

Table 5-63: PROF_CTRL0 (CPM Address 0x308)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved			PROF_CNT3_SEL					reserved			PROF_CNT2_SEL					reserved			PROF_CNT1_SEL					reserved			PROF_CNT0_SEL				

Table 5-64: PROF_CTRL0 Field Description

Field	Bits/Location	R/W	Description
PROF_CNT0_SEL	4:0	R/W	Select the event that this counter reflects. For event description see Table 3 1: Profiler Event Description. 0 WAIT_CNT 1 VPU_STL_CNT 2 DBLK_CONF_CNT 3 OSFC 4 TDRC 5 TDWC 6 ERWC 7 VPU0_NOP_INST 8 VPU1_NOP_INST 9 PCU_NOP_INST 10 SPU0_NOP_INST 11 SPU 1_NOP_INST 12 SPU 2_NOP_INST 13 SPU 3_NOP_INST 14 LSU0_NOP_INST 15 LSU1_NOP_INST 16 PMSS_HIT_CNT 17 PMSS_MIS_CNT 18 PWCWC 19 P_ADD_COUNT0 20 P_ADD_COUNT1 21 P_ADD_COUNT2 22 P_ADD_COUNT3 23 P_ADD_COUNT4 24 P_ADD_COUNT5 25 P_ADD_COUNT6 26 P_ADD_COUNT7 27 Reserved 28 Reserved 29 Reserved 30 Reserved 31 Reserved
reserved	7:5		
PROF_CNT1_SEL	12:8	R/W	Same as PROF_CNT0_SEL
reserved	15:13		
PROF_CNT2_SEL	20:16	R/W	Same as PROF_CNT0_SEL
reserved	23:21		
PROF_CNT3_SEL	28:24	R/W	Same as PROF_CNT0_SEL
reserved	31:29		

Reset Value: 0x00000000

5.3.4 PROF_CTRL1

Table 5-65: PROF_CTRL1 (CPM Address 0x30c)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved			PROF_CNT7_SEL					reserved			PROF_CNT6_SEL					reserved			PROF_CNT5_SEL					reserved			PROF_CNT4_SEL				

Table 5-66: PROF_CTRL1 Field Description

Field	Bits/Location	R/W	Description
PROF_CNT4_SEL	4:0	R/W	Same as PROF_CNT0_SEL
reserved	7:5		
PROF_CNT5_SEL	12:8	R/W	Same as PROF_CNT0_SEL
reserved	15:13		
PROF_CNT6_SEL	20:16	R/W	Same as PROF_CNT0_SEL
reserved	23:21		
PROF_CNT7_SEL	28:24	R/W	Same as PROF_CNT0_SEL
reserved	31:29		

Reset Value: 0x00000000

5.3.5 FRCC

Table 5-67: FRCC (CPM Address 0x310)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRCC																															

Table 5-68: FRCC Field Description

Field	Bits/Location	R/W	Description
FRCC	31:0	RO	Free Running Clock Counter Counts the number of core cycles

Reset Value: 0x00000000

5.3.6 PROF_CNTx

Table 5-69: PROF_CNTx (CPM Address 0x320)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PROF_CNTx																															

Table 5-70: PROF_CNTx Field Description

Field	Bits/Location	R/W	Description
PROF_CNTx	31:0	RO	Profiler counter X X = 0 thru 7

Reset Value: 0x00000000

5.3.7 P_LOW_ADDx

Table 5-71: P_LOW_ADDx (CPM Address 0x340)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P_LOW_ADD																															

Table 5-72: P_LOW_ADDx Field Description

Field	Bits/Location	R/W	Description
P_LOW_ADD	31:0	R/W	Low address for pc address range counter X * X is 0 to 7

Reset Value: 0x00000000

5.3.8 P_HI_ADDx

Table 5-73: P_HI_ADDx (CPM Address 0x360)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P_HI_ADDx																															

Table 5-74: P_HI_ADDx Field Description

Field	Bits/Location	R/W	Description
P_HI_ADDx	31:0	R/W	Hi address for PC address range counter X User should configure P_HI_ADDx > P_LOW_ADDx * X is 0 to 7.

Reset Value: 0x00000000

5.4 Wrapper

Table 5-75: Wrapper Programming Model

Name	Offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE	0x180	reserved																					BR_ALL	B2I_LC0	B2I_FLD	WR_RESET	WR_EN	ETMX2	CID_EN	NO_STALL	PR_TRACE	LS0_TRACE	LS1_TRACE
CMP_CFG	0x184	reserved																								BR_ALL	BR_FLD	CMP2_CFG		CMP1_CFG		CMP0_CFG	
LS0_CFG	0x188	reserved																								LS0_TR_CFG			LS0_LG		LS0_CMP_CFG		
LS1_CFG	0x18c	reserved																								LS1_TR_CFG			LS1_LG		LS1_CMP_CFG		
CMP0_START	0x190	COMP0_START																															
CMP0_END	0x194	COMP0_END																															
CMP1_START	0x198	COMP1_START																															
CMP1_END	0x19c	COMP1_END																															
CMP2_START	0x1a0	COMP2_START																															
CMP2_END	0x1a4	COMP2_END																															
PRED_ADDR	0x1a8	PRED_ADDR																															

Name	Offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CID_CNT	0x1ac	REV																												CID_CNT			

5.4.1 MODE

Table 5-76: MODE (CPM Address 0x180)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																					BR_ALL	B2I_LC0	B2I_FLD	WR_RESET	WR_EN	ETMX2	CID_EN	NO_STALL	PR_TRACE	LS0_TRACE	LS1_TRACE

Table 5-77: MODE Field Description

Field	Bits/Location	R/W	Description
LS1_TRACE	0:0	RO	1'b0
LS0_TRACE	1:1	RO	1'b0
PR_TRACE	2:2	RO	1'b0
NO_STALL	3:3	RO	1'b0
CID_EN	4:4	R/W	Active high. Predicate flags can be output as ContextID data. If negated, predicate data is transferred to the ETM as normal data only.
ETMX2	5:5	RO	1'b0
WR_EN	6:6	R/W	Active high. When asserted, the wrapper functionality is enabled.
WR_RESET	7:7	WO	Active high. Single shot. When asserted, the Pipeline Align, Control and Sequencer and ETM Interface are reset.
B2I_FLD	8:8	R/W	Active high. Trace optimization for bkrep2i loop count=1
B2I_LC0	9:9	R/W	Active high. Trace optimization for bkrep2i loop count=0
BR_ALL	10:10	R/W	Active high. Output all branches as indirect branches (excluding bkrep fold indications).
reserved	31:11		

Reset Value: 0x0000000C

5.4.2 CMP_CFG

Table 5-78: CMP_CFG (CPM Address 0x184)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																								BR_ALL	BR_FLD	CMP2_CFG		CMP1_CFG		CMP0_CFG	

Table 5-79: CMP_CFG Field Description

Field	Bits/Location	R/W	Description
CMP0_CFG	1:0	RO	Address selection for range comparator #0 00=Program Address 01=LS0 Address 10=LS1 Address 11=Reserved
CMP1_CFG	3:2	RO	Address selection for range comparator #1 00=Program Address 01=LS0 Address 10=LS1 Address 11=Reserved
CMP2_CFG	5:4	RO	Address selection for range comparator #2 00=Program Address 01=LS0 Address 10=LS1 Address 11=Reserved
BR_FLD	6:6	R/W	Specify mapping of bkrep foldback branch indication to ETM. Treat the last instruction of the last iteration of a loop as a change of flow. NB: this bit should be set prior to starting a trace run.
BR_ALL	7:7	R/W	Test mode: when asserted remap all loop foldback indications as indirect branches.
reserved	31:8		

Reset Value: 0x00000000

5.4.3 LS0_CFG

Table 5-80: LS0_CFG (CPM Address 0x188)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																								LS0_TR_CFG			LS0_LG		LS0_CMP_CFG		

Table 5-81: LS0_CFG Field Description

Field	Bits/Location	R/W	Description

LS0_CMP_CFG	2:0	RO	Select which comparator in-range indications to apply to LS0 address filtering
			ls0_cmp_cfg[0]=1 Filter LS0 accesses on comparator #0 in-range indication cr0
			ls0_cmp_cfg[1]=1 Filter LS0 accesses on comparator #1 in-range indication cr1
			ls0_cmp_cfg[2]=1 Filter LS0 accesses on comparator #2 in-range indication cr2
			Where more than 1 comparator is selected the in-range indications are combined according to the logical operation specified by ls0_lg

Field	Bits/Location	R/W	Description
LS0_LG	4:3	RO	Select logical operators to combine comparator in-range indications 00= comparators (#0 AND #1) AND #2 in-range indications 01= comparators (#0 AND #1) OR #2 in-range indications 10=comparators (#0 OR #1) AND #2 in-range indications 11=comparators (#0 OR #1) OR #2 in-range indications

LS0_TR_CFG	7:5	RO	Select which ext_trigger signals to apply to LS0 address filtering
			000=No filtering on external triggers
			ls0_tr_cfg[5]=1 Filter on ext_trigger[0]
			ls0_tr_cfg[6]=1 Filter on ext_trigger[1]
			ls0_tr_cfg[7]=1 Filter on ext_trigger[2]
			Selected external triggers are logically ORed with the selected range comparators

reserved	31:8		

Reset Value: 0x00000000

5.4.4 LS1_CFG

Table 5-82: LS1_CFG (CPM Address 0x18c)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																								LS1_TR_CFG			LS1_LG		LS1_CMP_CFG		

Table 5-83: LS1_CFG Field Description

Field	Bits/Location	R/W	Description

LS1_CMP_CFG	2:0	RO	Select which comparator in-range indications to apply to LS1 address filtering
			ls1_cmp_cfg[0]=1 Filter LS1 accesses on comparator #0 in-range indication cr0
			ls1_cmp_cfg[1]=1 Filter LS1 accesses on comparator #1 in-range indication cr1
			ls1_cmp_cfg[2]=1 Filter LS1 accesses on comparator #2 in-range indication cr2
			Where more than 1 comparator is selected the in-range indications are combined according to the logical operation specified by ls1_lg

LS1_LG	4:3	RO	Select logical operators to combine comparator in-range indications 00= comparators (#0 AND #1) AND #2 in-range indications 01= comparators (#0 AND #1) OR #2 in-range indications 10=comparators (#0 OR #1) AND #2 in-range indications 11=comparators (#0 OR #1) OR #2 in-range indications

Field	Bits/Location	R/W	Description

LS1_TR_CFG	7:5	RO	Select which ext_trigger signals to apply to LS1 address filtering
			000=No filtering on external triggers
			ls1_tr_cfg[5]=1 Filter on ext_trigger[0]
			ls1_tr_cfg[6]=1 Filter on ext_trigger[1]
			ls1_tr_cfg[7]=1 Filter on ext_trigger[2]
			Selected external triggers are logically ORed with the selected range comparators

reserved	31:8		

Reset Value: 0x00000000

5.4.5 CMP0_START

Table 5-84: CMP0_START (CPM Address 0x190)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP0_START																															

Table 5-85: CMP0_START Field Description

Field	Bits/Location	R/W	Description
COMP0_START	31:0	RO	32-bit Data Address Range Comparator #0 start address

Reset Value: 0x00000000

5.4.6 CMP0_END

Table 5-86: CMP0_END (CPM Address 0x194)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP0_END																															

Table 5-87: CMP0_END Field Description

Field	Bits/Location	R/W	Description
COMP0_END	31:0	RO	32-bit Data Address Range Comparator #0 end address

Reset Value: 0xFFFFFFFF

5.4.7 CMP1_START

Table 5-88: CMP1_START (CPM Address 0x198)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP1_START																															

Table 5-89: CMP1_START Field Description

Field	Bits/Location	R/W	Description
COMP1_START	31:0	RO	32-bit Data Address Range Comparator #1 start address

Reset Value: 0x00000000

5.4.8 CMP1_END

Table 5-90: CMP1_END (CPM Address 0x19c)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP1_END																															

Table 5-91: CMP1_END Field Description

Field	Bits/Location	R/W	Description
COMP1_END	31:0	RO	32-bit Data Address Range Comparator #1 end address

Reset Value: 0xFFFFFFFF

5.4.9 CMP2_START

Table 5-92: CMP2_START (CPM Address 0x1a0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP2_START																															

Table 5-93: CMP2_START Field Description

Field	Bits/Location	R/W	Description
COMP2_START	31:0	RO	32-bit Data Address Range Comparator #2 start address

Reset Value: 0x00000000

5.4.10 CMP2_END

Table 5-94: CMP2_END (CPM Address 0x1a4)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP2_END																															

Table 5-95: CMP2_END Field Description

Field	Bits/Location	R/W	Description
COMP2_END	31:0	RO	32-bit Data Address Range Comparator #2 end address

Reset Value: 0xFFFFFFFF

5.4.11 PRED_ADDR

Table 5-96: PRED_ADDR (CPM Address 0x1a8)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRED_ADDR																															

Table 5-97: PRED_ADDR Field Description

Field	Bits/Location	R/W	Description
PRED_ADDR	31:0	R/W	32-bit Data Address with which predicate flags are output to the ETM

Reset Value: 0x00000000

5.4.12 CID_CNT

Table 5-98: CID_CNT (CPM Address 0x1ac)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REV																												CID_CNT			

Table 5-99: CID_CNT Field Description

Field	Bits/Location	R/W	Description
CID_CNT	3:0	R/W	32-bit Data Address with which predicate flags are output to the ETM
REV	31:4	RO	Read only. Revision code.

Reset Value: 0x0000010F

5.5 OCEM_BPCOUNT_DRD

Table 5-100: OCEM_BPCOUNT_DRD Programming Model

Name	Offset	31–16	15–0
OCM_ACOUNT	0x1f0	reserved	OCM_ACOUNT
OCM_DCOUNT	0x1f4	reserved	OCM_DCOUNT
OCORE_DRD	0x1fc	OCORE_DRD	

5.5.1 OCM_ACOUNT

Table 5-101: OCM_ACOUNT (CPM Address 0x1f0)

31–16	15–0
reserved	OCM_ACOUNT

Table 5-102: OCM_ACOUNT Field Description

Field	Bits/Location	R/W	Description
OCM_ACOUNT	15:0	R/W	Data address breakpoint count value: Triggers a breakpoint once the data address breakpoint counter is equal to this value. After breakpoint is triggered the counter is cleared. In enhanced mode this count value is used for combined address and data breakpoints.
reserved	31:16		

Reset Value: 0x00000000

5.5.2 OCM_DCOUNT

Table 5-103: OCM_DCOUNT (CPM Address 0x1f4)

31–16	15–0
reserved	OCM_DCOUNT

Table 5-104: OCM_DCOUNT Field Description

Field	Bits/Location	R/W	Description
OCM_DCOUNT	15:0	R/W	Data value breakpoint count value (Enhanced mode only): Triggers a breakpoint once the data value breakpoint counter is equal to this value. After breakpoint is triggered the counter is cleared.
reserved	31:16		

Reset Value: 0x00000000

5.5.3 OCORE_DRD

Table 5-105: OCORE_DRD (CPM Address 0x1fc)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCORE_DRD																															

Table 5-106: OCORE_DRD Field Description

Field	Bits/Location	R/W	Description
OCORE_DRD	31:0	R/W	Output core data read register The OCORE_DRD register is used by the core software to pass values to the debugger.

Reset Value: 0x00000000

Note:

The OCORE_DRD register is written whenever the address 0xFC is used regardless of the iopage field in mod2 register

This register is write-only. The host can read the value written to this register by reading scan chain 0x9F.

6 CoreSight Programming Model

These registers can be accessed only using the APB slave address space.

6.1 High-Level Address Mapping of the Programming Model

Programming Model Sections	Address Mapping
CoreSight OCEM	0xd00 - 0xd04
Class 0x9 CoreSight Component	0xf00 - 0xfcc
CoreSight ID	0xfd0 - 0xffc

6.2 CoreSight OCEM

Table 6-1: CoreSight OCEM Programming Model

Name	Offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCCO	0xd00	SIZE								reserved							DIREC	SCC															
SCDA	0xd04	SCD																															

6.2.1 SCCO

Table 6-2: SCCO (CPM Address 0xd00)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIZE								reserved							DIREC	SCC															

Table 6-3: SCCO Field Description

Field	Bits/Location	R/W	Description
SCC	15:0	R/W	Scan chain code according to Table ?1 3: OCEM Scan Chains
DIREC	16:16	R/W	OCEM operation is read or write 0: read 1: write
reserved	23:17		
SIZE	31:24	R/W	Number of SCDA registers to read or write

Reset Value: 0x00000000

6.2.2 SCDA

Table 6-4: SCDA (CPM Address 0xd04)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCD																															

Table 6-5: SCDA Field Description

Field	Bits/Location	R/W	Description
SCD	31:0	R/W	Scan chain data according to Table ?1 3: OCEM Scan Chains

Reset Value: 0x00000000

6.3 CoreSight ID

Table 6-6: CoreSight ID Programming Model

Name	Offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P_ID4	0xfd0	reserved																								FOURKB_C				JEP106_CC			
P_ID0	0xfe0	reserved																								PN_L							
P_ID1	0xfe4	reserved																								JEP106_ICL				PN_M			
P_ID2	0xfe8	reserved																								REV				JEDEC_A	reserved	JEP106_ICH	
P_ID3	0xfec	reserved																								RE_AN				CU_MO			
C_ID0	0xff0	reserved																								ID0_VAL							
C_ID1	0xff4	reserved																								CCLA				ID1_VAL			
C_ID2	0xff8	reserved																								ID2_VAL							
C_ID3	0xffc	reserved																								ID3_VAL							

6.3.1 P_ID4

Table 6-7: P_ID4 (CPM Address 0xfd0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																								FOURKB_C				JEP106_CC			

Table 6-8: P_ID4 Field Description

Field	Bits/Location	R/W	Description
JEP106_CC	3:0	RO	JEP106 continuation code
FOURKB_C	7:4	RO	4KB count
reserved	31:8		

Reset Value: 0x00000004

6.3.2 P_ID0

Table 6-9: P_ID0 (CPM Address 0xfe0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																								PN_L							

Table 6-10: P_ID0 Field Description

Field	Bits/Location	R/W	Description
PN_L	7:0	RO	Part Number [7:0] (LSB)
reserved	31:8		

Reset Value: 0x00000021

6.3.3 P_ID1

Table 6-11: P_ID1 (CPM Address 0xfe4)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																								JEP106_ICL				PN_M			

Table 6-12: P_ID1 Field Description

Field	Bits/Location	R/W	Description
PN_M	3:0	RO	Part Number [11:8] (MSB)
JEP106_ICL	7:4	RO	JEP106 identity code [3:0] (LSB)
reserved	31:8		

Reset Value: 0x00000024

6.3.4 P_ID2

Table 6-13: P_ID2 (CPM Address 0xfe8)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																								REV				JEDEC_A	reserved	JEP106_ICH	

Table 6-14: P_ID2 Field Description

Field	Bits/Location	R/W	Description
JEP106_ICH	1:0	RO	JEP106 identity code [6:4] (MSB)
reserved	2:2		
JEDEC_A	3:3	RO	Always set. Indicates that a JEDEC assigned value is used

Field	Bits/Location	R/W	Description
REV	7:4	RO	Revision
reserved	31:8		

Reset Value: 0x0000000D

6.3.5 P_ID3

Table 6-15: P_ID3 (CPM Address 0xfec)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																								RE_AN				CU_MO			

Table 6-16: P_ID3 Field Description

Field	Bits/Location	R/W	Description
CU_MO	3:0	RO	Customer Modified
RE_AN	7:4	RO	RevAnd
reserved	31:8		

Reset Value: 0x00000000

6.3.6 C_ID0

Table 6-17: C_ID0 (CPM Address 0xff0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																								ID0_VAL							

Table 6-18: C_ID0 Field Description

Field	Bits/Location	R/W	Description
ID0_VAL	7:0	RO	Preamble
reserved	31:8		

Reset Value: 0x0000000D

6.3.7 C_ID1

Table 6-19: C_ID1 (CPM Address 0xff4)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																								CCLA				ID1_VAL			

Table 6-20: C_ID1 Field Description

Field	Bits/Location	R/W	Description
ID1_VAL	3:0	RO	Preamble
CCLA	7:4	RO	Component class
reserved	31:8		

Reset Value: 0x00000090

6.3.8 C_ID2

Table 6-21: C_ID2 (CPM Address 0xff8)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																								ID2_VAL							

Table 6-22: C_ID2 Field Description

Field	Bits/Location	R/W	Description
ID2_VAL	7:0	RO	Preamble
reserved	31:8		

Reset Value: 0x00000005

6.3.9 C_ID3

Table 6-23: C_ID3 (CPM Address 0xffc)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																								ID3_VAL							

Table 6-24: C_ID3 Field Description

Field	Bits/Location	R/W	Description
ID3_VAL	7:0	RO	Preamble
reserved	31:8		

Reset Value: 0x000000b1

6.4 Class 0x9 CoreSight Component

Table 6-25: Class 0x9 CoreSight Component Programming Model

Name	Offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTIR	0xfcc	reserved																								SUB_T				MAJ_T			

6.4.1 DTIR

Table 6-26: DTIR (CPM Address 0xfcc)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																								SUB_T				MAJ_T			

Table 6-27: DTIR Field Description

Field	Bits/Location	R/W	Description
MAJ_T	3:0	RO	Major Type and Class
SUB_T	7:4	RO	Sub Type
reserved	31:8		

Reset Value: 0x00000025