XBurst®1 CPU Core

Programming Manual

Release Date: Dec. 8, 2014

北京君正集成电路股份有限公司
Ingenic Semiconductor Co.,Ltd.

XBurst®1 CPU Core

Programming Manual

Disclaimer

This documentation is provided for use with Ingenic products. No license to Ingenic property rights is granted. Ingenic assumes no liability, provides no warranty either expressed or implied relating to the usage, or intellectual property right infringement except as provided for by Ingenic Terms and Conditions of Sale.

Ingenic products are not designed for and should not be used in any medical or life sustaining or supporting equipment.

All information in this document should be treated as preliminary. Ingenic may make changes to this document without notice. Anyone relying on this documentation should contact Ingenic for the current documentation and errata.

Ingenic Semiconductor Co., Ltd.

Ingenic Headquarters, East Bldg. 14, Courtyard #10
Xibeiwang East Road, Haidian District, Beijing, China,
Tel: 86-10-56345000
Fax:86-10-56345001
Http: //www.ingenic.com

CONTENTS

1 Overview

XBurst®1 CPU core is a high performance and low power implementation of XBurst® instruction set architecture. XBurst®1 deploys an innovative 9-stage pipeline micro-architecture and provides superior performance, die size and power consumption comparing with existent industry RISC cores.

1.1 Features of XBurst®1 core

Table 1.1 XBurst®1 core's Features

Item	Features
XBurst®1 CPU	● MIPS32 integer and floating point instruction set ● XBurst® SIMD instruction set ● 32 32-bit general purpose registers, ***no shadow GPR*** ● 9-stage pipeline ● Interlocked implementation ● Virtual address space: 4 G-Bytes
Multiple-Divide Unit (MDU)	● not fully pipelined multiplier, 3 cycles latency ● Minimum 2 clock cycles, maximum 34 clock cycles for divide
Branch Target Buffer (BTB)	● Virtually-tagged, Up to 128 entry direct mapped ● 2-bit branch history maintained
Memory Management Unit (MMU)	● 4 G-Bytes of address space ● 32 dual-entry full associative joint TLB ● 4 entry mini-ITLB + 4 entry mini-DTLB ● 7 different page size from 4KB to 16MB supported in any entry ● Support entry lock ● Space identifier ASID: 8 bits ● ***Small (1KB) page not implemented***
Data Cache	● Virtually-indexed, physically-tagged ● 4 way, 8-word line, alterable size: 4K, 8K, 16K bytes ● LRU replacement algorithm ● Write-back, write-through ● 16-word depth write buffer ● ***Cache line lock not implemented***
Instruction Cache	● Virtually-indexed, physically-tagged ● 4 way, 8-word line, alterable size: 4K, 8K, 16K bytes ● LRU replacement algorithm ● ***Cache line lock not implemented***
Debug&JTAG	● JTAG interface to host machine ● ACC mode to accelerate JTAG memory access ● Two instruction and one data breakpoint
Internal Timer	● ***N/A***
Bus Interface	● Compliance with AHB protocol

2 CP0

CP0 functions as System Control Coprocessor, which provides the register interface to the processor core and supports memory management, address translation, exception handling, and other privileged operations. This section describes processor core's definition and implementation of CP0 registers.

2.1 CP0 Register Summary

The following two subsections show the CP0 register set grouped by function and grouped by number.

2.1.1 CP0 Registers Grouped by Function

The CP0 registers set are divided into the register groups shown in Table 2.1.

Table 2.1 CP0 Register Grouped by Function

Category	Register Name	Register Number	Register Select	
CPU Configuration and Status	Config	16	0	section2.3.1.1
	Config1	16	1	section2.3.1.2
	Config2	16	2	section2.3.1.3
	Config3	16	3	section2.3.1.4
	Config4	16	4	section2.3.1.5
	Config5	16	5	section2.3.1.6
	Conifg7	16	7	section2.3.1.7
	PRId	15	0	section2.3.1.8
	EBase	15	1	section2.3.1.9
	Status	12	0	section2.3.1.10
	IntCtl	12	1	section2.3.1.11
TLB Management	Index	0	0	section 2.3.2.1
	Random	1	0	section2.3.2.2
	EntryLo0	2	0	section2.3.2.3
	EntryLo1	3	0	section2.3.2.3
	EntryHi	10	0	section2.3.2.4
	Context	4	0	section2.3.2.5
	PageMask	5	0	section2.3.2.6
	PageGrain	5	1	section2.3.2.7
	Wired	6	0	section2.3.2.8
	BadVAddr	8	0	section2.3.2.9
Exception Control	Cause	13	0	section2.3.3.1
	EPC	14	0	section2.3.3.2
	ErrorEPC	30	0	section2.3.3.3
Timer Registers	Count	9	0	section 2.3.4.1
	Compare	11	0	section2.3.4.2
Cache Management	TagLo	28	0	section2.3.5.1

	DataLo	28	1	section2.3.5.2
	ErrCtl	26	0	section2.3.5.3
Shadow Registers	SRStl	12	2	section2.3.6.1
	SRSMap	12	3	section2.3.6.2
Performance Monitoring	PerfCntCtl0	25	0	section2.3.7.1
	PerfCntCnt0	25	1	section2.3.7.2
	PerfCntCtl1	25	2	section2.3.7.3
	PerfCntCnt1	25	3	section2.3.7.4
Debug	Debug	23	0	section2.3.8.1
	Debug2	23	6	section2.3.8.2
	DEPC	24	0	section2.3.8.3
	DESAVE	31	0	section2.3.8.4
	WatchLo	18	0	section2.3.8.5
	WatchHi	19	0	section2.3.8.6
User Mode Support	HWREna	7	0	section2.3.9.1
	LLAddr	17	0	section2.3.9.2
Kernel Mode Support	KScratch1	31	2	section2.3.10.1
	KScratch2	31	3	section2.3.10.2
	KScratch3	31	4	section2.3.10.3
	KScratch4	31	5	section2.3.10.4
	KScratch5	31	6	section2.3.10.5
	KScratch6	31	7	section2.3.10.6
Multiple core registers	CoreCtrl	11	6	section2.3.11.1
	CoreStat	11	7	section2.3.11.2
	CoreIRQM	16	6	section2.3.11.3
	Mailbox0	22	0	section2.3.11.4
	Mailbox1	22	1	section2.3.11.5
	Mailbox2	22	2	section2.3.11.6
	Mailbox3	22	3	section2.3.11.7
	SpinLock	9	6	section2.3.11.8
	SpinAtom	9	7	section2.3.11.9

2.1.2 CP0 Registers Grouped by Number

The following table provides a numerical list of the processor CP0 register.

Table 2.2 CP0 Registers Grouped by Number

Register			Function	Location
Num	Sel	Name		
0	0	Index	Index into the TLB array	section 2.3.2.1
1	0	Random	Randomly generated index into the TLB array	section2.3.2.2
2	0	EntryLo0	Low-order portion of the TLB entry for even-numbered virtual pages.	section2.3.2.3
3	0	EntryLo1	Low-order portion of the TLB entry for odd-numbered virtual pages.	section2.3.2.3
4	0	Context	Pointer to page table entry in memory.	section2.3.2.5
5	0	PageMask	PageMask controls the variable page sizes in TLB entries.	section2.3.2.6
5	1	PageGrain	PageGrain controls the granularity of the page sizes in TLB entries.	section2.3.2.7
6	0	Wired	Controls the number of fixed ("wired") TLB entries.	section2.3.2.8
7	0	HWREna	Enable access via the RDHWR instruction to selected hardware registers in non-privileged mode.	section2.3.9.1
8	0	BadVaddr	Reports the address for the most recent address-related exception.	section2.3.2.9
9	0	Count	Processor cycle count	section 2.3.4.1
9	6	SpinLock	Spinlock register	section2.3.11.8
9	7	SpinAtom		section2.3.11.9
10	0	EntryHi	High-order portion of the TLB entry.	section2.3.2.4
11	0	Compare	Timer interrupt control.	section2.3.4.2
11	6	CoreCtrl	Multiple core control register	section2.3.11.1
11	7	CoreStat	Multiple core status register.	section2.3.11.2
12	0	Status	Processor status and control.	section2.3.1.10
12	1	IntCtl	Setup for interrupt vector and interrupt priority features.	section2.3.1.11
12	2	SRSCtl	Shadow register set control	section2.3.6.1
12	3	SRSMap	Shadow register map	section2.3.6.2
13	0	Cause	Cause of last exception	section2.3.3.1
14	0	EPC	Program counter at last exception.	section2.3.3.2
15	0	RPID	Processor identification and revision	section2.3.1.8
15	1	EBase	Exception base address.	section2.3.1.9
16	0	Config	Configuration register.	section2.3.1.1
16	1	Config1	Configuration for MMU, catches etc.	section2.3.1.2

16	2	Config2	Configuration for MMU ,caches etc.	section2.3.1.3
16	3	Config3	Interrupt and ASE capabilities.	section2.3.1.4
16	4	Config4	Indicates presence of Config5 register	section2.3.1.5
16	5	Config5	Provides information on EVA and cache error exception vector	section2.3.1.6
16	6	CoreIRQM	Multiple core interrupt request mask register	section2.3.11.3
16	7	Config7	BTB Configuration	section2.3.1.7
17	0	LLAddr	Contains the physical address ready by the most recent Load Linked (LL) instruction.	section2.3.9.2
18	0	WatchLo	Watchpoint address. (low order)	section2.3.8.5
19	0	WatchHi	Watchpoint address (high order) and mask.	section2.3.8.6
22	0	Mailbox0	Mailbox register 0	section2.3.11.4
22	1	Mailbox1	Mailbox register 1	section2.3.11.5
22	2	Mailbox2	Mailbox register 2	section2.3.11.6
22	3	Mailbox3	Mailbox register 3	section2.3.11.7
23	0	Debug	EJTAG Debug register.	section2.3.8.1
23	6	Debug2	EJTAG Debug 2 register.	section2.3.8.2
24	0	DEPC	Restart address from last EJTAG debug exception.	section2.3.8.3
25	0	PerfCntCtl0	Performance counter 0 control	section2.3.7.1
25	1	PerfCntCnt0	Performance counter 0 count	section2.3.7.2
25	2	PerfCntCtl1	Performance counter 1 control	section2.3.7.3
25	3	PerfCntCnt1	Performance counter 1 count	section2.3.7.4
26	0	ErrCtl	Software test enable of way-select and Data RAM arrays for I-Cache and D-Cache.	section2.3.5.3
28	0	TagLo	Cache tag read/write interface for I-Cache and D-Cache.	section2.3.5.1
28	1	DataLo	Low-order data read/write interface for I-Cache and D-Cache	section2.3.5.2
30	0	ErrorEPC	Program counter at last error	section2.3.3.3
31	0	DESAVE	Debug handler scratchpad register.	section2.3.8.4
31	2	KScratch1	Kernel scratch pad register 1.	section2.3.10.1
31	3	KScratch2	Kernel scratch pad register 2.	section2.3.10.2
31	4	KScratch3	Kernel scratch pad register 3.	section2.3.10.3
31	5	KScratch4	Kernel scratch pad register 4.	section2.3.10.4
31	6	KScratch5	Kernel scratch pad register 5.	section2.3.10.5
31	7	KScrathc6	Kernel scratch pad register 5.	section2.3.10.6

2.2 CP0 register Formats

This section contains descriptions of each CP0 register. The register are listed in numerical order, first by register number, then by select field number.

2.2.1 CP0 Register Field Types

For each register described below, field descriptions include the read/write properties of the field , and the reset value of the field.

For each register described below, field descriptions include the read/write access properties of the field and the reset state of the field. The read/write access properties are described in below table.

Table 2.3 CP0 Register Field R/W Access Types

Notation	Hardware Interpretation	Software Interpretation
R/W	A filed in which all bits are readable and writeable by software and potentially by hardware. Hardware updates of this field are visible by software reads. Software updates of this field are visible by hardware reads.	
R	A field that is either static or is updated only by hardware. If the Reset State of this field is either "0" or "1", hardware initializes this field to zero or to the appropriate state, respectively, on poweron.	A field to which the value written by software is ignored by hardware. Software may write any value to this field without affecting hardware behavior, Software read s of this field
W0	Hardware can write 1's or 0's to this field	Software writes will only cause the bit to be cleared. Software can never set this bit.
W1C	Hardware can write 1's or 0's to this field.	Software should write "1" to this bit to clear it. An example is the *I, R* and *W* bit fields in the *WatchHi0* register.
Reserved	A field that hardware does not update, and for which hardware can assume a zero value.	A field to which the value written by software must be zero. Software writes of non-zero values to this field may result in UNDEFINED behavior of the hardware. Software reads of this field return zero as long as all previous software writes are zero.

2.3 CP0 Register Descriptions

The following subsections describe the CP0 registers listed in above.

2.3.1 CPU Configuration and Status Registers

This section contains the following CPU Configuration and Status registers.

2.3.1.1 Device Configuration -Config(CP0 Register16, Select 0)

The *Config* register specifies various configuration and capabilities information. Most of the fields in the *Config* register are initialized by hardware during the reset period. Moreover, K0 field must be initialized in the reset exception handler.

Config Register

31	30 28	27 25	24 21	20 19 18 17 16	15	14 13	12 10	9 7	6 4	3	2 0
M	K23	Ku	Impl		BE	AT	AR	MT	Reserved	VI	K0

Name	Bits	Description	R/W	Reset
M	31	This bit is hardwired to "1" to indicated the presence of the *Config1* register.	R	1
K23	30:28	This field unused in the processor.	R	0
KU	27:25	They should be written as zero only.	R	0
Impl	24:16	This field is reserved for implementation, and is not implement. Must written as zeros, and read as zeros,	R	0
BE	15	Indicates the endian mode in which the processor is running: 0: Little endian 1: Big endian	R	HW config
AT	14:13	Architecture type implemented by the processor. This bits encoding value denotes address and register width The implemented instruction sets are denoted by the ISA register field of *Config3.* This field is hardwired to 2'b00 to indicate architecture type	R	0

		is MIPS32.		
AR	12:10	Architecture revision level. This bit always reads 1 to reflect Release 3 of MIPS32 architecture.	R	1
MT	9:7	MMU type. This field is hardwired to 3’b001 to indicate standard TLB .	R	1
Reserved	6:4	Must be written as zero; returns zero on read	R	0
VI	3	Virtual instruction cache. This field is hardwired to 1'b0 to indicate instruction cache is not virtual.	R	0
K0	2:0	Kseg0 cache attributes. 0~1: Cacheable, noncoherent, write-through, no write allocate 2, 7: Uncacheable 3~6:Cacheable, noncoherent, write-back, write allocate	R/W	2

2.3.1.2 Device Configuration 1-Config 1(CP0 Register16, Select1)

The *Config1* register is an adjunct to the *Config* register and encodes additional capabilities information. All fields in the *Config1* register are read-only.

The Icahce and Dcache configuration parameters include encoding for the number of sets per way, the line size, and the associativity. The total cache size for a cache is therefore:

Cache Size = Associatuvity * Line Size * Set Per Way

Config1 Register

31	30 25	24 22	21 19	18 16	15 13	12 10	9 7	6	5	4	3	2	1	0
M	MMU size	IS	IL	IA	DS	DL	DA	C2	MD	PC	WR	CA	EP	FP

Name	Bits	Description	R/W	Reset
M	31	Denotes that the *Config2* register is implemented at a select field value of 1	R	1
MMU size	30:25	Number of entries in the TLB minus one. The value 0 through 63 in this field correspond to 1 to 64 TLB entries. This field read as 0x1f indicate has 32 entries.	R	0x1F
IS	24:22	L1 instruction cache set per way. In this core, the instruction cache set per way is fixed at 128 .This field is encodes as follow: 000:Reserved 001:128 set per way 010-111: Reseveed.	R	1
IL	21:19	L1 instruction cache line size. In this core, the instruction cache line size is fixed at 32 bytes. As such , this field encoded as follows: 000-011:Reserved 100 :32 byte line size 101-111: Reserved.	R	4
IA	18:16	L1 Instruction cache associativity. In this core, the instruction cache associativity is fixed at 4 ways. As such, this field is encodes as follows: 000-010:Reserved 011 :4 ways 100-111:Reserved	R	3
DS	15:13	L1 Data cache number of sets per way. In this core , the data cache number of sets per way is fixed at 128.This field is encodes as follow: 000:Reserved 001:128 set per way 010-111: Reseveed.	R	1
DL	12:10	L1 data cache line size. In this core, the data cache line size is fixed at 32 bytes. As such , this field encoded as follows: 000-011:Reserved	R	4

		100 :32 byte line size 101-111: Reserved.		
DA	9:7	L1 data cache associativity. In this core, the data cache associativity is fixed at 4 ways. As such, this field is encodes as follows: 000-010:Reserved 011 :4 ways 100-111:Reserved	R	3
C2	6	Coprocessor 2 implemented: 0x1: Coprocessor 2 implements.	R	1
MD	5	MDMX Application Specific Extension (ASE). A logic '0' indicate that the MDMX ASE is not implemented in the floating point unit (FPU) of the core.	R	0
PC	4	Performance Counter enable. There is at least one performance counter implemented in this core. Hence this bit is always logic '1'. Refer to the *PerfCtl0-1* and *PerfCnt0*-1 registers for more information.	R	1
WR	3	Watch registers implemented. This bit always reads 1 because this core always has watch registers. Refer to the *WathcLo/WatchHi* registers.	R	1
CA	2	MIPS16e present. This bit always reads 0 to indicate the MIPS16e compressed-code instruction set is not available.	R	0
EP	1	EJTAG implemented. This bit always reads 1 as the EJTAG debug unit is provided on this core.	R	1
FP	0	FPU implemented. This bit indicates not only that the processor contains support for a floating point unit, but that such a unit is attached. If an FPU is implemented, the capabilities of the FPU can be read from the capability bits in the FIR CP1 register.	R	1

2.3.1.3 Device Configuration 2-Config 2(CP0 Register16, Select2)

The *Config2* register encodes level 2 cache configurations.(level 3 cache not implemented).

Config3 Register

31	30 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0
M	TU	TS	TL	TA	SU	SS	SL	SA

Name	Bits	Description	R/W	Reset
M	31	This bit is hardwired to '1' to indicate the presence of the Config3 register.	R	1
TU	30:28	An L3 cache not implemented in this core. As such, the TU, TS,TL,TA bits of this register, which report L3 information, are not used and are all tied to 0.	R	0
TS	27:24		R	0
TL	23:20		R	0
TA	19:16		R	0
SU	15:12	Implementation-specific secondary cache control or status bits. This bit always reads 1 to indicate Secondary cache implemented.	R	1
SS	11:8	L2 cache set per way. In this core, the cache set per way is fixed at 2048 .This field is encodes as follow: 000-100:Reserved 101 :2048 set per way	R	5
SL	7:4	L2 cache line size. In this core, the cache line size is fixed at 32 bytes. As such , this field encoded as follows: 000-011:Reserved 100 :32 byte line size 101-111: Reserved.	R	4
SA	3:0	L2 cache associativity. In this core, the cache associativity is fixed at 8 ways. As such, this field is encodes as follows: 000-110:Reserved 111 :8 ways	R	7

2.3.1.4 Device Configuration 3-Config 3(CP0 Register16, Select3)

Config 3 provides information about the presence of optional extensions to the base MIPS32 architecture in addition to those specified in Config 2. All fields in the Config3 register are read-only.

Config3 Register

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
M	BPG	CMGCR	Reserved	BP	BI	SC	PW	Reserved	IPLW	MMAR	MCU	ISAOnExc	ISA	ULRI	RXI	DSP2P	DSPP	CTXTC	ITL	LPA	VEIC	VInt	SP	CDMM	MT	SM	TL

Name	Bits	Description	R/W	Reset
M	31	Denotes that the *Config4* register is implemented at a select field value of 1	R	1
BPG	30	Big Pages feature is implemented. Always read as 1'b0, indicates Big Pages are not implemented and PageMask register is 32bit wide.	R	0
CMGCR	29	Coherency Manager memory-mapped Global Configuration Register Space is implemented. Always read as 1'b0, indicates CMGCR space is not implemented.	R	0
Reserved	28	Must be written as zero; returns zero on read	R	0
BP	27	*BadInstrP* register implemented. Always read as 1'b0, indicates *BadInstrP* register not implemented.	R	0
BI	26	*BadInstr* register implemented. Always read as 1'b0, indicates *BadInstr* register not implemented.	R	0
SC	25	Segment Control implemented. Always read as 1'b0, indicates Segment Control not implemented.	R	0
PW	24	Hardware Page Table Walk implemented. Always read as 1'b0, indicates Page Table Walking not implemented.	R	0
Reserved	23	Must be written as zero; returns zero on read	R	0
IPLW	22:21	$Config3_{MCU}$ is zero, then MCU ASE is not implemented and this field is not used.	R	0
MMAR	20:18	$Config3_{ISA}$ is zero, then microMIPS32 is not implemented and this field is not used.	R	0
MCU	17	MCU ASE is implemented. Always read as zero, indicates MCU ASE is not implemented.	R	0
ISAOnEx	16	Reflects the Instruction Set Architecture used after vectoring	R	0

c		to an exception. Always read as zero, indicates MIPS32 is used on entrance to an exception vector.		
ISA	15:14	Indicates Instruction Set Availability. These bits are always 0 to indicate MIPS32.	R	0
ULRI	13	UserLocal register implemented. Always read as zero, indicated UserLocal register is not implemented.	R	0
RXI	12	Indicates whether the RIE and XIE bits exist within the PageGrain register. Read always as one, indicates the RIE and XIE bits are implemented within the PageGrain register.	R	1
DSP2P	11	MIPS DSP ASE Revision 2 implemented. Always read as zero, indicates Revision 2 of the MIPS DSP ASE is not implemented.	R	0
DSPP	10	MIPS DSP ASE implemented. Always read as zero, indicates MIPS DSP ASE is not implemented.	R	0
CTXTC	9	*ContextConfig* register is implemented. Always read as zero, indicates *ContextConfig* is not implemented.	R	0
ITL	8	MIPS IFlow Trace mechanism implemented. Always read as zero, indicates MIPS IFlow Trace is not implemented.	R	0
LPA	7	Our processor is MIPS32 processor and this bit returns zero on read.	R	0
VEIC	6	Support for an external interrupt controller. Always read as zero, indicates EIC interrupt mode not support.	R	0
VInt	5	Vectored interrupt implemented. This bit indicates whether vectored interrupts are implemented. On this core, this bit reads 1 to indicate the CPU can handle vectored interrupts.	R	1
SP	4	Small (1KByte) page support is implemented. Always reads 0 to indicate the CPU does not support 1Kbyte TLB pages.	R	0
CDMM	2	Common Device Memory Map implemented. Always read as zero, indicates CDMM is not implemented.	R	0
MT	2	MIPS MT ASE implemented. Always read as zero, indicates MIPS MT ASE is not implemented.	R	0
SM	1	SmartMIPS ASE implemented. Reads 0 to indicate the CPU does not include the	R	0

		instructions of the SmartMIPS ASE.		
TL	0	Trace Logic implemented. Read as 0 to indicate Trace Logic is not implemented.	R	0

2.3.1.5 Device Configuration 4-Config 4(CP0 Register16, Select4)

The *Config4* register encodes additional capabilities such as TLBNV instruction support and the number of kernel scratch registers.

Config4 Register

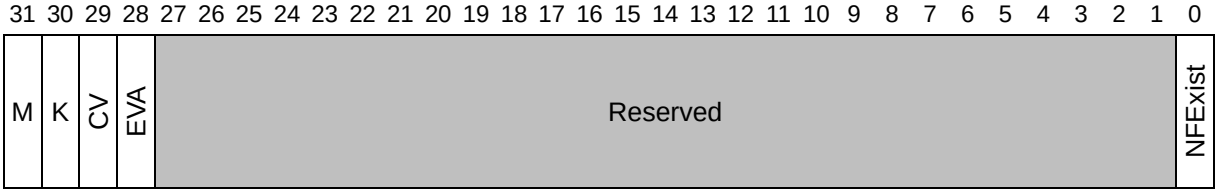
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M	IE		AE	VTLBSizeExt				KscrExist								MMUExDef		Reserved	FTLBPageSize					FTLBWays				FTLBSets			

Name	Bits	Description	R/W	Reset
M	31	Denotes that the *Config5* register is implemented at a select field value of 1	R	1
IE	30:29	TLB invalidate instruction support/configuration. Read as zero, indicates TLBINV, TLBINVF, $EntryHi_{EHINV}$ not support by hardware.	R	0
AE	28	Read as zero, indicates $EntryHI_{ASID}$ is not extened.	R	0
VTLBSizeExt	27:24	VTLB is not implemented and this field read as zero.	R	0
KScrExist	23:16	Indicates how many scratch registers are available to kernel-mode software within COP0 register 31.In this CPU, six kernel scratch registers are included at register 2,3,4,5,6 and 7. Each bit represents a select for Coprocessor0 Register 31. Bit 16 represents Select 0(*DESAVE* register), Bit 23 represents Select 7. If the bit is set, the associated scratch register is implemented and available for kernel-mode software. Therefore, this field contains a value of 0xfd(8'b11111101). This indicates that bits 23-18 are set, corresponding to selects 7, 6,5,4,3 and 2.	R	0xfd
MMUExDef	15:14	MMU Extension Definition. Read as zero, indicates Config4 [12:0] must be written as zero, returns zeros on read.	R	0
Reserved	13	Must be written as zero, returns zeros on read.	R	0
FTLBPageSize	12:8	A FTLB not implemented in this core. As such, FTLBPageSize, FTLBWays, FTLBSets bits of this register, which report FTLB information, are not used and are all tied to 0.	R	0
FTLBWays	7:4		R	0
FTLBSets	3:0		R	0

2.3.1.6 Device Configuration 5-Config 5(CP0 Register16, Select5)

The *Config5* register encodes additional capabilities for the address mode programming and cache error exceptions.

Config5 Register



Name	Bits	Description	R/W	Reset
M	31	Denotes that the *Config6* register is not implemented at a select field value of 0	R	0
K	30	Segmentation Control is not implemented and this bit read as zero.	R	0
CV	29	Segmentation Control is not implemented and this bit read as zero.	R	0
EVA	28	Enhanced Virtual Addressing instructions is not implemented and read as zero.	R	0
Reserved	27:1	Must be written as zero; returns zero on read	R	0
NFExist	0	Indicate that the Nested Fault feature exists. Read as zero, indicates the nested fault feature is not exists.	R	0

2.3.1.7 Device Configuration 7-Config 7(CP0 Register16, Select7)

Config7 Register

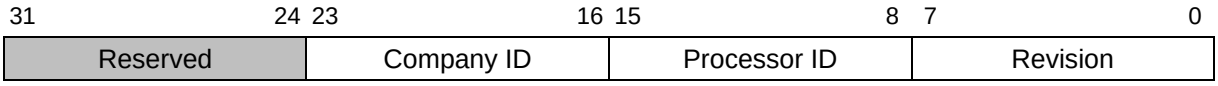
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													ALLOC	BTBV	BTBE

Name	Bits	Description	R/W	Reset
Reserved	31:3	Must be written as zero, returns zeros on read.	R	0
ALLOC	2	Allocate hint for PREF	RW	0
BTBV	1	Writing 1to this bit to invalidate BTB. Read always as zero.	RW	0
BTBE	0	BTB enable.	RW	0

2.3.1.8 Processor ID-PRId(CP0 Register 15, Select 0)

The Processor Identification (*PRId*) register is a 32 bit read only register that contains information identifying manufacturer, manufacturing options ,processor identification and revision level of the processor.

PRId Register



Name	Bits	Description	R/W	Reset
Reserved	31:24	Must be written as zero; returns zero on read	R	0
Company ID	23:16	Company ID. Identifies the company that designed or manufactured the processor.	R	Preset
Processor ID	15:8	Processor ID. Identifies the uA type of processor. This field allows software to distinguish between the various types of processors. 1) Generation(15~13): The number of the architecture design. 0 - XBurst1; 1 - XBurst2; others, reserved 2) uA(12~8): micro architecture version. please reference to SOC document.	R	Preset
Revision	7:0	The revision number of mass production. 1) Process(7~4): encode of IC process technology. 0 - SMIC40LP; 1 - TSMC40LP; others-reserved 2) version(3~0):internal version of different implementation. Please refer to SOC document.	R	preset

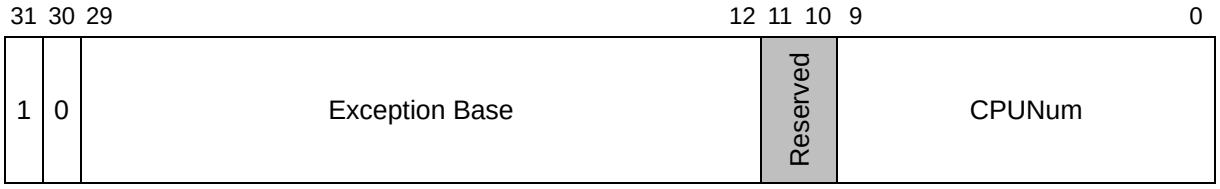
2.3.1.9 Exception Base Address-EBase(CP0 Register 15, Select 1)

The *EBase* register is a read/write register containing the base address of the exception vectors used when *StatusBEV* equals 0, and a read-only CPU number value that may be used by software to distinguish different processor in a multi-processor system.

Bit 31:30 of the *EBase* Register are fixed with the value 2'b10, and the addition of the base address and the exception offset is done inhibiting a carry between bit 29 and bit 30 of the final exception address. The combination of these two restriction forces the final exception address to be in the keseg0 or kseg1 unmapped virtual address segments. For cache error exceptions, bit 29 ir forced to a 1 in the ultimate exception base address segment.

If the value of the exception base register is to be changed, this must be done with *StatusBEV* equal to 1. The operation of the processor is UNDEFINED if the exception base field is written with a different value when *StatusBEV* is 0.

Ebase Register

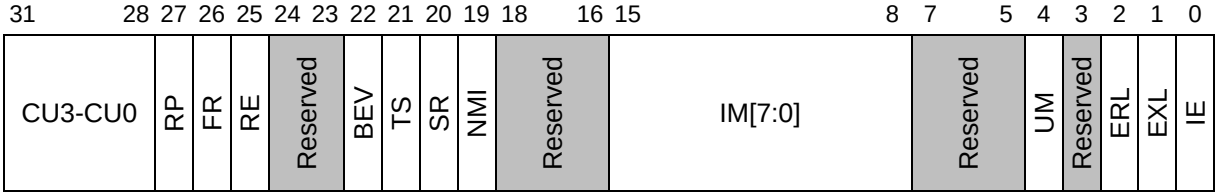


Name	Bits	Description	R/W	Reset
1	31	This bit is ignored on write and returns one on read	R	1
0	30	This bit is ignored on write and returns zero on read.	R	0
Exception Base	29:12	In conjunction with bits 31...30, this field specifies the base address of the exception vectors when $Status_{BEV}$ is zero.	RW	0
Reserved	11,10	Must be written as zero, returns zero on read.	R	0
CPUNum	9:0	This field specifies the number of the CPU in a multi-processor system and can be used by software to distinguish a particular processor from the others. The value in this field is set by inputs to the processor hardware when the processor is implemented in the system environment. In a single processor system, this value should be set to zero. This field can also be read via RDHWR register 0.	R	0

2.3.1.10 Status (CP0 Register 12, Select 0)

The Status register is a read/write register that contains the operating mode, interrupt enabling and the diagnostic states of the processor.

Status Register



Name	Bits	Description	R/W	Reset
CU3	31	Coprocessor 3 Usable. Because the CPU does not support a coprocessor 3, $Status_{CU3}$ is hardwired to zero.	RW	0
CU2	30	Coprocessor 2 Usable. Controls access to coprocessor 2. 0: Access not allowed. 1: Access allowed. In this core, CU2 is used for MXU unit.	RW	0
CU1	29	Coprocessor 1 Usable. Controls access to coprocessor 1. 0: Access not allowed. 1: Access allowed. CU1 is used for a floating-point unit.	RW	0
CU0	28	Coprocessor 0 is always usable when the processor is running in Kernel Mode or Debug Mode, independent of the state of the CU0 bit.	RW	0
RP	27	Enables reduced power mode implemented. Always read as zero, indicate reduced power mode is not implemented.	R	0
FR	26	Floating Register. This bit is used to control the floating-point register mode for 64-bit point unit. This bit always read as zero, indicate the floating-point register can contain any 32-bit data byte, 64-bit data types are store in even-odd pairs of registers.	R	0
RE	25	Reverse Endian. Enables Reverse endianness for instructions that execute in User mode. This bit is always 0 as this feature is not support in the CPU.	R	0
MX	24	MIPS DSP extension. This bit is always 0 in the CPU, indicate MIPS DSP is not implemented.	R	0
Reserved	23	Must be written as zero; returns zero on read	R	0
BEV	22	Controls the location of exception vectors. 0: Normal 1: Bootstrap	RW	1
TS	21	Indicates that the TLB has detected a match on multiple	RW	0

		entries. It is implementation dependent whether this detection occurs at all, on a write to the TLB, or an access to the TLB. When such a detection occurs, the processor initiates a machine check exception and sets this bit. If the condition can be corrected, this bit should be cleared by software before resuming normal operation.		
SR	20	Soft Reset. The CPU only supports a full external reset, so this bit is not used and always reads zero.	R	0
NMI	19	The CPU not implemented an NMI reset, so this bit is not used and always reads zero.	R	0
ASE	18	The CPU not supports ASE, so this bit is not used and always reads zero.	R	0
Reserved	17:16	Must be written as zero; returns zero on read	R	0
IM[7:0]	15:10	Interrupt Mask. Controls the enabling of each the hardware interrupts. 0: Interrupt request disabled 1: Interrupt request enabled Note:$Config3_{VEIC}$ is hardwired to '0', indicate External Interrupt Controller mode is not supported, so IM[7:0] is only used for interrupt mask.	R/W	0x0
IM1...IM0	9:8	Interrupt Mask: Controls the enabling of each of the software interrupts. 0: Interrupt request disabled. 1: Interrupt request enabled.	R/W	0x0
Reserved	7:5	Must be written as zero; returns zero on read	R	0
UM	4	This bit denotes the base operating mode of the processor .The encoding for this bit is : 0: Base mode is Kernel Mode. 1: Base mode is User Mode.	RW	0
R0	3	The Supervisor Mode is not implemented, this bit is reserved. This bit must be ignored on write and read as zero.	R	0
ERL	2	Error Level. Set by the processor when a reset exception is taken. 0: normal level; 1: error level. When EXL is set: ● The processor is running in kernel mode ● Hardware and software interrupts are disabled ● The ERET instruction will use the return address held in Error EPC instead of EPC. ● Segment kuseg is treated as an unmapped and uncached region. This allows main memory to be	RW	1

		accessed in the presence of cache errors. The operation of the processor is UNDEFINED if the ERL bit is set while the processor is executing instructions from kuseg.		
EXL	1	Exception Level. Set by the processor when any exception other than a Reset exception is taken. 0: normal level; 1: exception level. When EXL is set: ● The processor is running in kernel mode ● Hardware and software interrupts are disabled ● TLB Refill exceptions use the general exception vector instead of the TLB Refill vector. ● EPC Cause$_{BD}$ and SRSCtl will not be updated if another exception is taken	RW	0
IE	0	Interrupt Enable. Acts as the master enable for software and hardware interrupts: 0: disable interrupts 1: enable interrupts	RW	0

2.3.1.11 Interrupt Control - IntCtl (CP0 Register 12, Select 1)

The *IntCtl* register controls the interrupt capabilities of the core, including vectored interrupt and support for an external interrupt controller. $Config3_{VEIC}$ $Config3_{VInt}$

IntCtl Register

31 29	28 26	25 23	22 10	9 5	4 0
IPTI	IPPCI	IPFDC	Reserved	VS	Reserved

Name	Bits	Description	R/W	Reset
IPTI	31:29	This field specifies the IP number to which the Timer Interrupt request is merged. The bits field must be ignored on write and read as 0x7.	R	0x7
IPPCI	28:26	This field specifies the IP number to which the Performance Counter Interrupt request is merged. The bits field must be ignored on write and read as 0x6.	R	0x6
IPFDC	25:23	This field specifies the IP number to which the Fast Debug Channel Interrupt request is merged. The bits field must be ignored on write and read as 0x5.	R	0x5
Reserved	22:10	Must be written as zero; returns zero on read	R	0
VS	9:5	Vector spacing. If vectored interrupts are implemented(as denoted by $Config3_{VEIC}$ or $Config3_{VInt)}$, this field specifies the spacing between vectored interrupts.	RW	0
Reserved	4:0	Must be written as zero; returns zero on read	R	0

VS Field Encoding	Spacing Between Vectors(hex)	Spacing Between Vectors(decimal)
0x00	0x000	0
0x01	0x020	32
0x02	0x040	64
0x04	0x080	128
0x08	0x100	256
0x10	0x200	512

2.3.2 TLB Management Registers

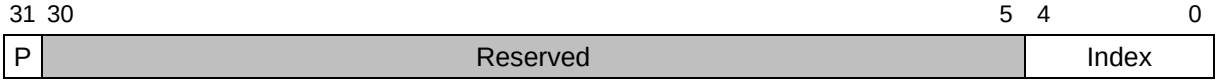
This section contains the following TLB management registers.

- Section 2.3.2.1," Index Register (CP0 Register 0, Select 0)"
- Section 2.3.2.2 ,"Random Register (CP0 Register 1, Select 0)"
- Section 2.3.2.3 ,"EntryLo0, EntryLo1 Register (CP0 Register 2, 3, Select 0)"
- Section 2.3.2.4 ,"EntryHi Register (CP0 Register 10, Select 0)"
- Section 2.3.2.5 ,"Context Register (CP0 Register 4, Select 0)"
- Section 2.3.2.6 ,"PageMask Register (CP0 Register 5, Select 0)"
- Section 2.3.2.7 ,"Page Granularity-PageGrain (CP0 Register 5, Select 1)"
- Section 2.3.2.8 ,"Wired Register (CP0 Register 6, Select 0)"
- Section 2.3.2.9 ,"BadVAddr Register (CP0 Register 8, Select 0)"

2.3.2.1 Index Register (CP0 Register 0, Select 0)

A 32-bit read/write register that contains the index used to access the TLB for TLBP, TLBR, and TLBWI instructions. The width of the index field is 5.

Index Register



Name	Bits	Description	R/W	Reset
P	31	Probe Failure. Hardware writes this bit during execution of the TLBP instruction to indicate whether a TLB match occurred: **Encoding** / **Meaning** 0 — A match occurred, and the Index field contains the index of the matching entry 1 — No match occurred and the Index field is UNPREDICTABLE	R	0
Reserved	30:5	Must be written as zero; returns zero on read	R	0
Index	4:0	TLB index. Software writes this field to provide the index to the TLB entry referenced by the TLBR and TLBWI instructions. Hardware writes this field with the index of the matching TLB entry during execution of the TLBP instruction. If the TLBP fails to find a match, the contents of this field are UNPREDICTABLE.	RW	0

2.3.2.2 Random Register (CP0 Register 1, Select 0)

The *Random* register is a read-only register whose value is used to index the TLB during a TLBWR instruction. The width of the Random field is calculated in the same manner as that described for the *Index* register above.

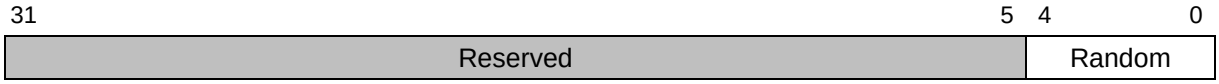
The value of the register varies between an upper and lower bound as follow:

- A lower bound is set by the number of TLB entries reserved for exclusive use by the operating system (the contents of the *Wired* register). The entry indexed by the *Wired* register is the first entry available to be written by a TLB Write Random operation.
- An upper bond is set by the total number of TLB entries minus 1.

Within the required constraints of the upper and lower bounds, the manner in which the processor selects values for the *Random* register is implementation-dependent.

The processor initializes the *Random* register to the upper bound on a Reset Exception, and when the *Wired* register is written.

Random Register



Name	Bits	Description	R/W	Reset
Reserved	31:5	Must be written as zero; returns zero on read	R	0
Random	4:0	TLB Random Index	R	0x1f

2.3.2.3 EntryLo0, EntryLo1 Register (CP0 Register 2, 3, Select 0)

The pair of *EntryLo* registers acts as the interface between the TLB and the TLBR, TLBWI, and TLBWR instructions. The contents of the *EntryLo0* and *EntryLo1* registers are undefined after an address error, TLB invalid, TLB modified, or TLB refill exceptions.

Software may determine the value of *PABITS* by writing all ones to the *EntryLo0* or *EntryLo1* registers and reading the value back. Bits read as "1" from the PFN field allow software to determine the boundary between the PFN and Fill fields to calculate the value of *PABITS.*

The contents of the *EntryLo0* and *EntryLo1* registers are not defined after an address error exception and some field may be modified by hardware during the address error exception sequence. Software writes of the *EntryHi* register (via MTC0) do not cause the implicit update of address-related fields in the *BadVAddr* or *Context* registers.

EntryLo0, EntryLo1 Register

31	30	29 26	25 6	5 3	2	1	0
RI	XI	Reserved	PFN	C	D	V	G

Name	Bits	Description	R/W	Reset
RI	31	Read Inhibit. If this bit is set, an attempt to read data from the page cause a TLB Invalid exception, even if the *V*(Valid) bit is set. The *RI* bit is enabled only if the *RIE* bit of the *PageGrain* register is set. If the *RIE* bit of *PageGrain* is not set, the RI bit of *EntryLo0/EntryLo1* is a reserved 0 bit .	RW	0
XI	30	Execute Inhibit. If this bit is set, an attempt to fetch from the page cause a TLB Invalid exception, even if the *V* (Valid) bit is set. The *XI* bit is enabled only if the *XIE* bit of the *PageGrain* register is set. If the *XIE* bit of *PageGrain* is not set, the RI bit of *EntryLo0/EntryLo1* is a reserved 0 bit.	RW	0
Reserved	29:26	Must be written as zero; returns zero on read	R	0x0
PFN	25:6	Page Frame Number. Contributes to the definition of the high-order bits of the physical address. Our Processor is not enabled to support 1KB pages (*Config3*$_{SP}$*=0*), the PFN field corresponds to bits 31..12 of the physical address.	RW	0x0
C	5:3	Cache attribute of the page. See Table Cache Coherency Attributes for more information.	RW	0
D	2	Dirty attribute of the page. The "Dirty" flag. Indicates that the page has been written, and/or is writable. If this bit is a one, stores to the page are permitted. If this bit is a zero, stores to the page cause a TLB Modified exception. Software can use this bit to track pages that have been written to. When a page is first mapped, this bit should be cleared. It set on the first write that causes an exception.	RW	0

V	1	Valid attribute of the page. The "Valid" flag indicates that the TLB entries, and thus the virtual page mapping, are valid. If this bit is a set, accesses to the page are permitted. If this bit is a zero, accesses to the page cause a TLB *Invalid* exception. This bit can be used to make just one of a pair of pages valid	RW	0
G	0	Global attribute of the page. The "Global" bit. On a TLB write, the logical AND of the G bits in both the *Entry0* and *Entry1* registers become the G bit in the TLB entry. If the TLB entry G bit is a one, then the ASID comparisons are ignored during TLB matches. On a read from a TLB entry, the G bits of both *Entry 0* and *Entry 1* reflect the state of the TLB G bit.	RW	0

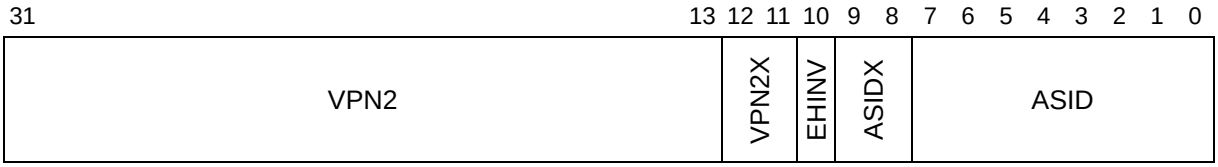
2.3.2.4 EntryHi Register (CP0 Register 10, Select 0)

The *EntryHi* register contains the virtual address match information used for TLB read, write, and access operations.

A TLB exception (TLB Refill, TLB Invalid, or TLB Modified) causes bits $VA_{31:13}$ of the virtual address to be written into the VPN2 field of the *EntryHi* register. The ASID field is written by software with the current address space identifier value and is used during the TLB comparison process to determine TLB match.

The VPN2 field of the *EntryHi* register is not defined after an address error exception.

EntryHi Register



Name	Bits	Description	R/W	Reset
VPN2	31:13	$VA_{31:13}$ of the virtual address (virtual page number/2). This field written by hardware on a TLB exception or on a TLB read. and is written by software before a TLB write.	RW	0x0
VPN2X	12：11	*Config3*$_{SP}$ hardwired to '0', indicated CPU does not support 1Kbyte TLB pages, then must be written as zero, returns zero on read.	R	0x0
EHINV	10	*Config4*$_{IE}$ hardwired to '0', indicate TLB invalidate instruction not support, and then must be written as zero, returns zero on read.	R	0
ASIDX	9:8	*Config4*$_{AE}$ hardwired to '0', then must be written as zero, returns zero on read.	R	0x0
ASID	7:0	Address space identifier. This field is written by hardware on a TLB read and by software to establish the current ASID valued for TLB write and against which TLB references math each entry's TLB ASID field.	RW	0x0

2.3.2.5 Context Register (CP0 Register 4, Select 0)

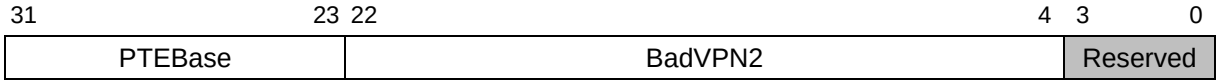
The *Context* register is a read/write register containing a pointer to an entry in the page table entry (PTE) array. This array is an operating system data structure that stores virtual-to-physical translations. During a TLB miss, the operating system loads the TLB with the missing translation from the PTE array. The *Contex* register duplicates some of the information provided in the *BadVAddr* register .

The *Context* register is a organized in such a way that the operating system can directly reference a 16-biyte structure in memory that describes the mapping. For PTE structures of other sizes, the content of this register can be used by the TLB refill handler after appropriate shifting and masking.

A TLB exception (TLB Refill, TLB Invalid, or TLB Modified) causes bits VA31:13 of the virtual address to be written into the *BadVPN2* field of the *Context* register. The *PTEBase* field is written and used by the operating system.

The *BadVPN2* field of the *Context* register is not defined after an address error exception and this field may be modified by hardware during the address error exception sequence.

Context register

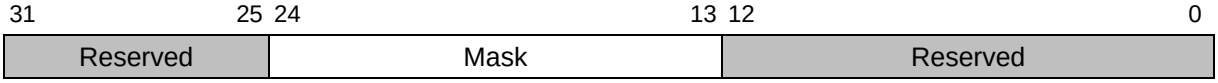


Name	Bits	Description	R/W	Reset
PTEBase	31:23	This field is for use by the operating system and is normally written with a value that allows the operating system to use the *Context* Register as a pointer into the current PTE array in memory.	R/W	0x0
BadVPN2	22:4	This field is written by hardware on a TLB exception. It contains bits VA [31:13] of the virtual address that cause the exception.	R	0
Reserved	3:0	Must be written as zero; returns zero on read	R	0

2.3.2.6 PageMask Register (CP0 Register 5, Select 0)

The *PageMask* register is a read/write register used for reading from and writing to the TLB. It holds a comparison mask that sets the variable page size for each TLB entry.

PageMask Register

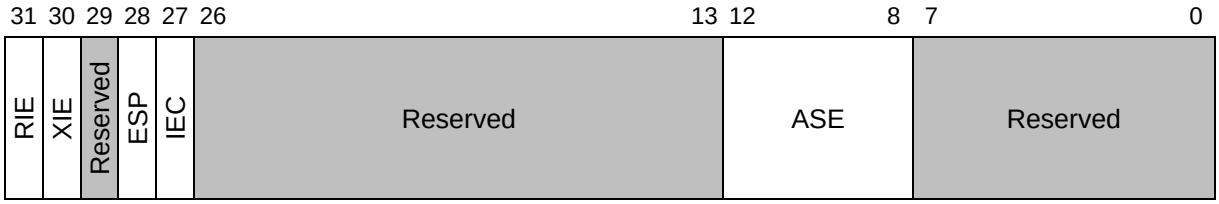


Name	Bits	Description	R/W	Reset
Reserved	31:25	Must be written as zero; returns zero on read	R	0
Mask	24:13	Mask bits for varying page size. 0000_0000_0000: 4KB 0000_0000_0011: 16KB 0000_0000_1111: 64KB 0000_0011_1111: 256KB 0000_1111_1111: 1MB 0011_1111_1111: 4MB 1111_1111_1111: 16MB	RW	0x0
Reserved	12:0	Must be written as zero; returns zero on read	R	0

2.3.2.7 Page Granularity-PageGrain (CP0 Register 5, Select 1)

The *PageGrain* register is a read/write register used for enabling 1KB page support, the XI/RI TLB protection bits.

PageGrain Register



Name	Bits	Description	R/W	Reset
RIE	31	Read Inhibit Enable. Read always as 1,indicates RI bit of the *EntryLo0* and *EntryLo1* registers is enable.	R	1
XIE	30	Execute Inhibit Enable. Read always as 1, indicates XI bit of the *EntryLo0* and *EntryLo1* registers is enabled.	R	1
Reserved	29	Must be written as zero; returns zero on read	R	0
ESP	28	This bit is always 0 as 1K pages are not supported. This bit must be written with 0.	R	0
IEC	27	Enable unique exception codes for the Read-Inhibit and Execute-Inhibit exceptions. 0: Read-Inhibit and Execute-Inhibit exceptions both use the TLBL exception code. 1: Read-Inhibit exceptions use the TLBRI exception code. Execute-Inhibit exceptions use the TLBXI exception code. Note: Software not change this bit to 1.	R/W	0
Reserved	26:13	Must be written as zero; returns zero on read	R	0
ASE	12:8	Ignored on write; return zero on read.	R	0
Reserved	7:0	Must be written as zero; returns zero on read	R	0

2.3.2.8 Wired Register (CP0 Register 6, Select 0)

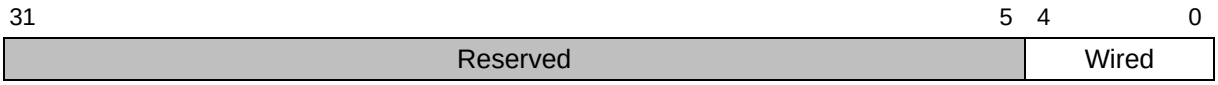
The *Wired* register is a read/write register that specifies the boundary between the wired and random entries in the TLB.

The width of the Wired field is calculated in the same manner as that described for the *Index* register. *Wired* entries are fixed, non-replaceable entries which are not overwritten by a TLBWR instruction. *Wired* entries can be overwritten by a TLBWI instruction.

The *Wired* register is set to zero by a Reset exception. Writing the *Wire* entries cause the *Random* register to reset to its upper bound.

The operation of the processor is UNDEFINED if a value greater than or equal to the number of TLB entries is written to the *Wired* register.

Wired Register



Name	Bits	Description	R/W	Reset
Reserved	31:5	Must be written as zero; returns zero on read	R	0
Wired	4:0	TLB wired boundary	RW	0x0

2.3.2.9 BadVAddr Register (CP0 Register 8, Select 0)

The *BadVAddr* register is a read only register that captures the most recent virtual address that caused one of the following exceptions:

- Address error (AdEL or AdES)
- TLB Refill
- TLB Invalid
- TLB Modified

The *BadVAddr* register does not capture address information for cache or bus errors, or for Watch Exception, since none is an address error

BadVAddr Register

31	0
BadVAddr	

Name	Bits	Description	R/W	Reset
BadVAddr	31:0	Failed virtual address in Address Error or TLB Faults.	R	0x0

2.3.3 Exception Control Registers

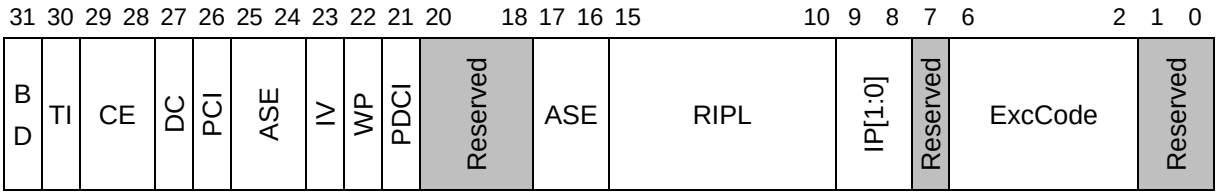
This section contains the following exception controls registers.

- Section 2.3.3.1," Cause Register (CP0 Register 13, Select 0)"
- Section 2.3.3.2," Exception Program Counter (CP0 Register 14, Select 0)"
- Section 2.3.3.3," ErrorEPC Register (CP0 Register 30, Select 0)"

2.3.3.1 Cause Register (CP0 Register 13, Select 0)

The *Cause* register primarily describes the cause of the most recent exception. In addition, fields also control software interrupt requests and the vector through which interrupts are dispatched. With the exception of the $IP_{1..0}$, DC, IV, and WP fields, all fields in the *Cause* register are read-only.

Cause Register



Name	Bits	Description	R/W	Reset
BD	31	Indicates whether the last exception taken in a branch delay slot. 0: Not in delay slot 1: In delay slot The processor updates BD only if $Status_{EXL}$ was zero when the exception occurred.	R	0
TI	30	The CPU is not support timer interrupt, so this bit must be written as zero and returns zero on read.	R	0
CE	29:28	Coprocessor unit number referenced when a Coprocessor unusable exception is taken. This field is loaded by hardware on every exception, but is UNPREDICTABLE for all exception for Coprocessor Unusable. 00: Coprocessor 0 01: Coprocessor 1(FPU) 10: Coprocessor 2(MXU) 11:Coprocessor 3(not support in this core)	R	0
DC	27	Disable *Count* register. 0:Enable counting of *Count* register 1:Disable counting of *Count* register	R/W	0
PCI	26	Performance Counter Interrupt. Read on this bit always as 0 indicates Performance Counter not implemented.	R	0
ASE	25:24	These bits are reserved because MCU ASE is not implemented, these bits returns zero on reads and must be written with zeros.	R	0

IV	23	Indicates whether an interrupt exception uses the general exception vector or a special interrupt vector. 0:Use the general exception vector(0x180) 1:Use the special interrupt vector(0x200)	RW	0
WP	22	Indicates that the watch exception was deferred because $Status_{EXL}$ or $Status_{ERL}$ were a one at the time the watch exception was detected. This bit both indicates that the watch exception was deferred, and causes the exception to be initiated once $Status_{EXL}$ and $Status_{ERL}$ are both zero. As such, software must clear this bit as part of the watch exception handler to prevent a watch exception loop. Software should not write a 1 to this bit when its value is a 0, thereby causing a 0-to-1 transition. If such a transition is caused by software, it is UNPREDICTABLE whether hardware ignores the write, accepts the write with no side effects, or accepts the write and initiates a watch exception once $Status_{EXL}$ and $Status_{ERL}$ are both zero.	RW	0
FDCI	21	Fast Debug Channel Interrupt. Read always as 0(Fast Debug Channel not implement).	R	0
Reserved	20:18	Must be written as zero; returns zero on read	R	0
ASE	17:16	These bits are reserved because MCU ASE is not implemented, these bits returns zero on reads and must be written with zeros.	R	0
IP[7:2]	15:10	Indicates an Interrupt is pending. **Bit / Name / Meaning** 15 / IP7 / Hardware interrupt 5 14 / IP6 / Hardware interrupt 4 13 / IP5 / Hardware interrupt 3 12 / IP4 / Hardware interrupt 2 11 / IP3 / Hardware interrupt 1 10 / IP2 / Hardware interrupt 0	R	0
IP[1:0]	9:8	Controls the request for software interrupt. **Bit / Name / Meaning** 9 / IP1 / Request software interrupt 1 8 / IP0 / Request software interrupt 0 Note: Hardware not implemented an external interrupt controller, so software do not use this field.	RW	0
Reserved	7	Must be written as zero; returns zero on read	R	0
Exc Code	6:2	Exception code. See Table 2.5	R	0
Reserved	1:0	Must be written as zero; returns zero on read	R	0

Table 2.4 Cause Register ExcCode Field Descriptions

Exception Code Value	Mnemonic	Description
0	Int	Interrupt.
1	Mod	TLB modification exception.
2	TLBL	TLB exception. (load or instruction fetch)
3	TLBS	TLB exception. (store)
4	AdEL	Address error exception. (load or instruction fetch)
5	AdES	Address error exception. (store)
6	N/A	--
7	N/A	--
8	Sys	Syscall exception.
9	Bp	Breakpoint exception.
10	RI	Reserve red instruction exception.
11	CPU	Coprocessor unusable exception.
12	Ov	Integer overflow exception.
13	Tr	Trap exception.
14	N/A	--
15	FPE	Floating point exception.
16-22	N/A	--
23	WATCH	Reference to WatchHi/WatchLo address.
24	Mcheck	Machine Check.
25-31	-	Reserve red.

2.3.3.2 Exception Program Counter (CP0 Register 14, Select 0)

The Exception Program Counter (*EPC*) is a read/write register that contains the address at which processing resumes after an exception has been serviced. All bits of the *EPC* register are significant and must be writable.

Unless the *EXL* bit in the *Status* register is already a 1, the processor writes the *EPC* register when an exception occurs.

For synchronous (precise) exceptions, the *EPC* contains one of the following:

- The virtual address of the instruction that was the direct cause of the exception.
- The virtual address of the immediately preceding branch or jump instruction, when the exception causing instruction is in a branch delay slot and the Branch Delay bit in the Cause register is set.

Note that the processor does not update the *EPC* register when the EXL bit in the *Status* register has been set to one for a new exception. Moreover, the register can be modified via the MTC0 instruction.

Exception Program Counter

31	0
EPC	

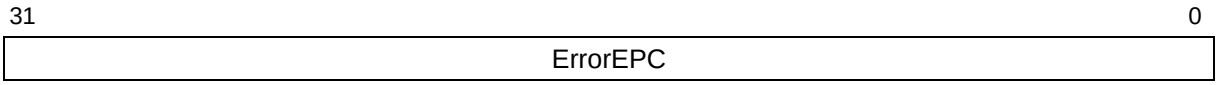
Name	Bits	Description	R/W	Reset
EPC	31:0	Exception Program Counter.	RW	0

2.3.3.3 ErrorEPC Register (CP0 Register 30, Select 0)

The *ErrorEPC* register is a read-write register, similar to the *EPC* register, except that *ErrorEPC* is used on error exceptions. All bits of the *ErrorEPC* register are significant an must be writable.It is also used to store the program counter on Reset exceptions.

This full 32-bit register is filled with the result address on a cache error exception or any kind of CPU reset--in fact, any exception which sets $Status_{ERL}$ and leaves the CPU in "error mode".

ErrorEPC Register



Name	Bits	Description	R/W	Reset
ErrorEPC	31:0	Most recent error exception program counter.	RW	0

2.3.4 Timer Registers

This section contains the following timer registers.

- Section 2.3.4.1, " Count(CP0 Register 9,Select 0)"
- Section 2.3.4.2," Compare(CP0 Register 11,Select 0)"

2.3.4.1 Count(CP0 Register 9,Select 0)

The *Count* register acts as a timer, incrementing at a constant rate. Incrementing of this register occurs whether or not an instruction is executed, retired, or any forward progress is made through the pipeline. When enabled by clearing the *DC* bit in the *Cause* register, the counter increments every other clock (half the clock rate).

The *Count* field starts counting from whatever value is load into it. However, OS timers are usually implemented by leaving *Count* free-running and writing *Compare* register as necessary. This counter rolls over when reaching it maximum value.

Count Register

31	0
Count	

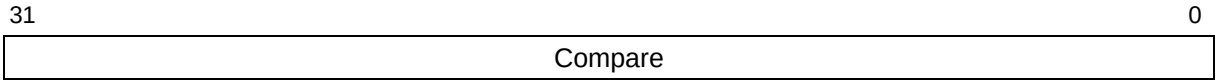
Name	Bits	Description	R/W	Reset
Count	31:0	Interval counter	RW	0

2.3.4.2 Compare (CP0 Register 11,Select 0)

The *Compare* register acts in conjunction with the *Count* register to implement a timer and timer interrupt function. When the value of the *Count* register equals the value of the *Compare* register, will generate an interrupt. Traditionally, this has been done by multiplexing it with hardware interrupt 5 in order to set interrupt bit *IP(7)* in the *Cause* register.

For diagnostic purposes, the *Compare* register is a read/write register. In normal use however, the *Compare* register is write-only. As a side effect, writing to this register clears the timer intrerupt.

Compare Register



Name	Bits	Description	R/W	Reset
Compare	31:0	Interval count compare value	RW	0

2.3.5 Cache Management Registers

- Section 2.3.5.1," TagLo Register (CP0 Reigster 28, Select 0)"
- Section 2.3.5.2," DataLo Register (CP0 number 28, Select 1)"
- Section 2.3.5.3,"ErrCtl Register (CP0 Register 26, Select 0)"

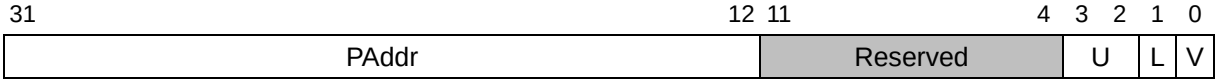
2.3.5.1 TagLo Register (CP0 Register 28, Select 0)

The TagLo register acts as the interface to the cache tag array. The Index Store Tag and Index Load Tag operations of the CACHE instruction use the TagLo register as the source and destination of tag information, respectively.

However, software must be able to write zeros into the *TagLo* registers and then use the Index Store Tag cache operation to initialize the cache tags to a valid state at powerup.

Note: ICache and Dcache share *TagLo* register.

TagLo Register



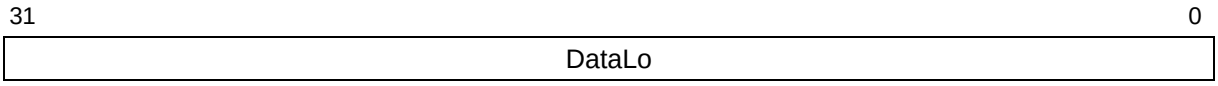
Name	Bits	Description	R/W	Reset
PAddr	31:12	Physical address of the indexed cache line.	RW	0
Reserved	11:4	Must be written as zero; returns zero on read	R	0
U	3:2	Dirty bits of data cache. Each bit for half of a cache line.	R	0
L	1	Lock bit of the cache line. (reserved)	RW	0
V	0	Valid bit of the cache line.	RW	0

2.3.5.2 DataLo Register (CP0 number 28, Select 1)

The *DataLo* register acts as the interface to the cache data array. The Index Load Tag operation of the CACHE instruction reads the corresponding data values into the *DataLo* register.

Note: ICache and Dcache share *DataLo* register.

DataLo Register



Name	Bits	Description	R/W	Reset
DataLo	31:0	Low-order data read from cache.	RW	0

2.3.5.3 ErrCtl Register (CP0 Register 26, Select 0)

This register is implementation dependent.

ErrCtl Register

31	30	29	0
Reserved		WST	Reserved

Name	Bits	Description	R/W	Reset
Reserved	31:30	Must be written as zero; returns zero on read	R	0
WST	29	0: the operation of the instruction only causes L1 cache. 1: the operation of the instruction causes L1 and L2 cache. Not support in T-series chip.	RW	0
Reserved	28:0	Must be written as zero; returns zero on read	R	0

2.3.6 Thread Context and Shadow Control Registers

The processor System does not support thread contexts or shadow registers, the shadow Register Set Control (SRSCtl) register and Shadow Register Set Map(SRSMap) register are implemented to allow software to read this register to determine that shadow registers are not implemented.

2.3.6.1 SRSCtl Register (CP0 Register 12, Select 2)

The SRSCtl register controls the operation of GPR shadow sets in the processor.

SRSCtl Register

31 30	29 26	25 22	21 18	17 16	15 12	11 10	9 6	5 4	3 0
Reserved	HSS	Reserved	EICSS	Reserved	ESS	Reserved	PSS	Reserved	CSS

Name	Bits	Description	R/W	Reset
Reserved	31:30	Must be written as zero; returns zero on read	R	0
HSS	29:26	Highest Shadow Set. A value of zero in this field indicates that only the normal GPRs are implemented.	R	0
Reserved	25:22	Must be written as zero; returns zero on read	R	0
EICSS	21:18	EIC interrupt mode shadow set. Because $Config3_{VEIC}$=0, this field must be written as zero, and returns zero on read.	R	0
Reserved	17:16	Must be written as zero; returns zero on read	R	0
ESS	15:12	Exception Shadow Set. This processor implement only the normal GPRs, so this field has no effect, must be written as zero, and returns zero on read.	R/W	0
Reserved	11:10	Must be written as zero; returns zero on read	R	0
PSS	9:6	Previous Shadow Set. Because GPR shadow registers are not implemented, this field has no effect, must be written as zero, and returns zero on read.	R/W	0
Reserved	5:4	Must be written as zero; returns zero on read	R	0
CSS	3:0	Current Shadow Set. Because GPR shadow registers are not implemented, this field has no effect, must be written as zero, and returns zero on read.	R	0

2.3.6.2 SRSMap Register (CP0 Register 12, Select 3)

$SRSCtl_{HSS}$ is zero, the results of a software read or write of this register have no significant. The SRSMap register is a 32-bit readable and writable register.

SRSMap Register

31 30 29	26 25	22 21	18 17 16 15	12 11 10 9	6 5 4 3	0
SRSMap						

Name	Bits	Description	R/W	Reset
SRSMap	31:0	Used for temporary storage of information.	RW	0

2.3.7 CPU Performance Monitor Registers

The Architecture supports implementation dependent performance counters that provide the capability to count event or cycles for use in performance analysis. Each performance counter consists of a pair of registers: a 32-bit control register and a 32-bit counter register.

Each performance counter is mapped into even-odd select values of the *PerfCnt* register: Even selects access the control register and odd selects access the counter register. Below table shows an example of two performance counter and how they map into the select values of the *PerfCnt* register.

Table 2.5 Example Performance Counter Usage of the PerfCnt CP0 Register

Performance Counter	PerfCnt Register Select Value	PerfCnt Register Usage
0	PerfCnt, Select 0	Control Register 0
	PerfCnt, Select 1	Counter Register0
1	PerfCnt, Select 2	Control Register 1
	PerfCnt, Select 3	Counter Register1

2.3.7.1 Performance Counter Control0 Register (CP0 Register 25, Select 0)

Performance Counter Control 0Register

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
M		Impl	Reserved	PCTD	EventExt	Event	IE	U	S	K	EXL

Name	Bits	Description	R/W	Reset
M	31	This bit is a one, indicate another pair of Performance Counter Control and Counter registers is implemented at a MTC0 or MFC0 select field value of "2" and "3".	R	1
Reserved	30	Must be written as zeros return zeros on reads.	R	0
Impl	29:25	This field not used by the implementation, must be written as zero, return zero on read.	R	0
Reserved	24:16	Must be written as zeros return zeros on reads.	R	0
PCTD	15	PDTrace Performance Counter Tracing feature is not implemented. This bit readable and writable, but has no effect.	RW	0
EventExt	14:11	The CPU not support more than the 64 encodings possible in the 6-bit Event field, must be written as zero, return zero on read.	R	0
Event	10:5	Selects the event to be counted by the corresponding Counter Register. **Encoding** / **Meaning** 0 — Count CPU clock cycles (resolution is updating per clock cycle)	R/W	0

		1 Count pipeline stall cycles 2 Count number of I-cache miss occurred 3 Count number of D-cache miss occurred 4 Count instruction's number of being fetched but finally discarded 5 Count total number of issued instructions 6 Count times of TLB exception due to Instruction fetch 7 Count times of TLB exception due to load/store		
IE	4	In CPU not support performance counter interrupt, so this bit has no significant. It is writable and readable.	RW	0
U	3	Enables event counting in User Mode. 0: Disable event counting in User Mode. 1:Enable event counting in User Mode	RW	0
S	2	The processor does not implement Supervisor Mode; this bit must be ignored on write and return zero on read.	R	0
K	1	Enables event counting in Kernel Mode. Note: this enables event counting only when the UM, EXL and ERL bits in the *Status* register are zero. 0:Disable event counting in Kernel Mode 1:Enable event counting in Kernel Mode	RW	0
EXL	0	Enables event counting when the EXL bit in the *Status* registers is one and the ERL bit in the *Status* register is zero. 0: Disable event counting while EXL=1,ERL=0 1:Enable event counting while EXL=1,ERL=0 Counting is never enabled when the ERL bit in the *Status* register of the DM bit in the *Debug* register is one.	RW	0

2.3.7.2 Performance Counter Counter0 Register (CP0 Register 25, Select 1)

Performance Counter Counter0 Register

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
Event Count																															

Name	Bits	Description	R/W	Reset
Event Count	31:0	Increments once for each event that is enabled by the corresponding Control Register.	RW	0

2.3.7.3 Performance Counter Control1 Register (CP0 Register 25, Select 2)

Performance Counter Control1 Register

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
M		Impl	Reserved	PCTD	EventExt	Event	IE	U	S	K	EXL

Name	Bits	Description	R/W	Reset
M	31	This bit is a zero, indicate there are no another pair of Performance Counter Control and Counter registers.	R	0
Reserved	30	Must be written as zeros return zeros on reads.	R	0
Impl	29:25	This field not used by the implementation, must be written as zero, return zero on read.	R	0
Reserved	24:16	Must be written as zeros return zeros on reads.	R	0
PCTD	15	PDTrace Performance Counter Tracing feature is not implemented. This bit readable and writable, but has no effect.	RW	0
EventExt	14:11	The CPU not support more than the 64 encodings possible in the 6-bit Event field, must be written as zero, return zero on read.	R	0
Event	10:5	Selects the event to be counted by the corresponding Counter Register. **Encoding** / **Meaning** 0 — Count CPU clock cycles (resolution is updating per clock cycle) 1 — Count pipeline stall cycles 2 — Count number of I-cache miss occurred 3 — Count number of D-cache miss occurred 4 — Count instruction's number of being fetched but finally discarded 5 — Count total number of issued instructions 6 — Count times of TLB exception due to Instruction fetch 7 — Count times of TLB exception due to load/store	R/W	0x0
IE	4	In CPU not support performance counter interrupt, so this bit has no significant. It is writable and readable.	RW	0
U	3	Enables event counting in User Mode. 0: Disable event counting in User Mode. 1:Enable event counting in User Mode	RW	0
S	2	The processor does not implement Supervisor Mode, this bit must be ignored on write and return zero on read.	R	0

K	1	Enables event counting in Kernel Mode. Note: this enables event counting only when the UM, EXL and ERL bits in the *Status* register are zero. 0:Disable event counting in Kernel Mode 1:Enable event counting in Kernel Mode	RW	0
EXL	0	Enables event counting when the EXL bit in the *Status* registers is one and the ERL bit in the *Status* register is zero. 0: Disable event counting while EXL=1,ERL=0 1:Enable event counting while EXL=1,ERL=0 Counting is never enabled when the ERL bit in the *Status* register of the DM bit in the *Debug* register is one.	RW	0

2.3.7.4 Performance Counter Counter0 Register (CP0 Register 25, Select 3)

Performance Counter Counter3 Register

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
Event Count																															

Name	Bits	Description	R/W	Reset
Event Count	31:0	Increments once for each event that is enabled by the corresponding Control Register.	RW	0

2.3.8 Debug Registers

This section contains the following hardware access registers.

- Section 2.3.8.1,"Debug Register (CP0 Register 23, Select 0)"
- Section 2.3.8.1,"Debug2 Register (CP0 Register 23, Select 6)"
- Section 2.3.8.1," Debug Exception Program Counter Register (CP0 Register 24, Select 0)"
- Section 2.3.8.1," Debug Save Register (CP0 Register 31, Select 0)"
- Section 2.3.8.1," WatchLo Register (CP0 Register 18, Select 0)"
- Section 2.3.8.1," WatchHi Register (CP0 Register 19, Select 0)"

2.3.8.1 Debug Register (CP0 Register 23, Select 0)

The *Debug* register is used to control the debug exception and provide information about the cause of the debug exception and when re-entering at the debug exception vector due to a normal exception in debug mode. The R (read only) field information bits are updated by hardware every time the debug exception is taken or when a normal exception is taken when already in debug mode.

Some of the bits and fields are only updated on debug exceptions and/or exceptions in debug mode, as shown below:

– DSS, DBp, DDBL, DDBS, DIB, DINT are updated on both debug exceptions and on exceptions in debug modes.

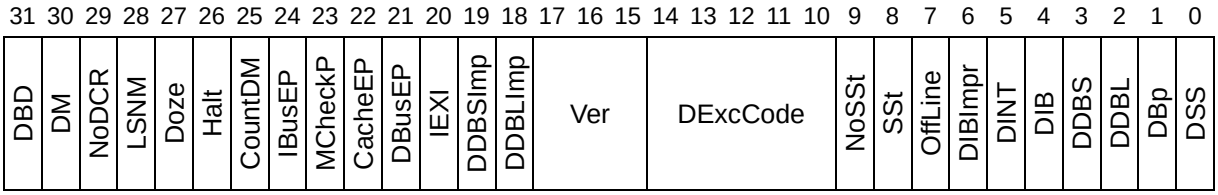
– DExcCode is updated on exceptions in debug mode, and is undefined after a debug exception.

– Halt and Doze are updated on a debug exception, and is undefined after an exception in debug mode.

– DBD is updated on both debug and on exceptions in debug modes.

All bits and fields are undefined when read from normal mode, except Ver and DM.

Debug Register



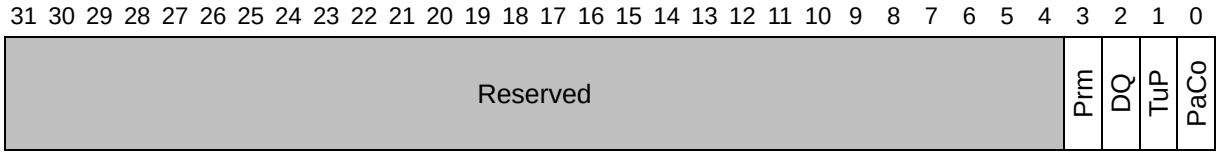
Name	Bits	Description	R/W	Reset
DBD	31	Indicates whether the most recent debug exception in Debug Mode occurred in a branch or jump delay slot. 0: not in delay slot; 1: in delay slot	R	0
DM	30	Indicates that the processor is operating in debug mode: 0: processor not in debug mode 1: processor in debug mode	R	0
NoDCR	29	Indicates whether the dseg segment is present: Read always as zero, indicates dseg segment is present.	R	0
LSNM	28	Controls access of load/store between dseg and remaining memory when the dseg segment is present: 0: Load/stores in dseg address range goes to dseg.	RW	0

		1: Load/stores in dseg address range goes to main memory.		
Doze	27	The implementation does not support low-power modes, then this bit always reads as zero.	R	0
Halt	26	The implementation does not support a halt state, then the bit always reads as zero.	R	0
CountDM	25	Controls or indicates the Count register behavior in Debug Mode. The reset value of this bit indicates the behavior after reset, and always read as one indicates Count register is running in Debug Mode.	R	1
IBusEP	24	In Debug Mode, Bus error exception not applies to a Debug Mode Bus Error exception, this bit is read-only(R) and read as zero.	R	0
MCheckP	23	In Debug Mode, a Machine Check exception not applies to a Debug Mode Machine Check exception, and this bit is read-only(R) and reads as zero.	R	0
CacheEP	22	In Debug Mode, a Cache Error exception not applies to a Debug Mode Cache Error exception. This bit is read-only(R) and read as zero.	R	0
DBusEP	21	In Debug Mode, a Bus Error exception not applies to a Debug Mode Cache Error exception. This bit is read-only(R) and read as zero.	R	0
IEXI	20	An imprecise Error Exception Inhibit(IEXI) is not implemented. This bit is read-only(R) and reads as zero.	R	0
DDBSImpr	19	A Debug Data Break Store Imprecise exception is not implemented, this bit read as zero.	R	0
DDBLImpr	18	A Debug Data Break Load Imprecise exception is not implemented, this bit read as zero.	R	0
EJTAGVer	17:15	JTAG version.	R	0
DExcCode	14:10	Indicates the cause of the latest exception in debug mode. Cause for those normal exceptions that may occur in debug mode.	R	0
NoSSt	9	Read always as zero, indicate Single-step feature available.	R	0
SSt	8	Controls whether single-step feature is enabled: 0:No debug single-step exception enabled. 1:Debug single step exception enabled.	RW	0
OffLine	7	MIPS MT processors is not implemented, this bit is read-only(R) and reads as zero.	R	0
DIBImpr	6	A Debug Instruction Break Imprecise exception not implemented, this bit reads as zero.	R	0

DINT	5	Indicates that a debug interrupt exception occurred. Cleared on exception in debug mode. 0: No debug interrupt exception 1: Debug interrupt exception	RW	0
DIB	4	Indicates that a debug instruction break exception occurred. Cleared on exception in debug mode. 0: No debug instruction exception 1: Debug instruction exception	R	0
DDBS	3	Indicates that a debug data break exception occurred on a store. Cleared on exception in debug mode. 0: No debug data exception on a store 1: Debug instruction exception on a store	R	0
DDBL	2	Indicates that a debug data break exception occurred on a load. Cleared on exception in debug mode. 0: No debug data exception on a load 1: Debug instruction exception on a load	R	0
DBp	1	Indicates that a debug software breakpoint exception occurred. Cleared on exception in debug mode. 0: No debug software breakpoint exception 1: Debug software breakpoint exception	R	0
DSS	0	Indicates that a debug single-step exception occurred. Cleared on exception in debug mode. 0: No debug single-step exception 1: Debug single-step exception	R	0

2.3.8.2 Debug2 Register (CP0 Register 23, Select 6)

Debug2 Register



Name	Bits	Description	R/W	Reset
Reserved	31:4	Must be written as zeros return zeros on reads.	R	0
Prm	3	Primed break exception is not implemented, this bit reads as zero.	R	0
DQ	2	Data qualified break exception is not implemented, this bit reads as zero.	R	0
Tup	1	Tuple break exception is not implemented, this bit reads as zero.	R	0
PaCO	0	Pass counter exception is not implemented, this bit reads as zero.	R	0

2.3.8.3 Debug Exception Program Counter Register (CP0 Register 24, Select 0)

The Debug Exception Program Counter (*DEPC*) register is a read/write register that contains the address at which processing resumes after a debug exception has been serviced. It is part of the EJTAG specification and the reader is referred there for the format and description of the register. All bits of the *DEPC* register are significant and must be writable.

When a debug exception occurs, the processor writes the DEPC register with,

- the virtual address of the instruction that was the direct cause of the exception, or
- the virtual address of the immediately preceding branch or jump instruction, when the exception causing instruction is in a branch delay slot, and the *Branch Delay* bit in the *Cause* register is set.

The processor reads the DEPC register as the result of execution of the DERET instruction.

Software may write the DEPC register to change the processor resume address and read the DEPC register to determine at what address the processor will resume.

Debug Exception Program Counter Register

Name	Bits	Description	R/W	Reset
DEPC	31:0	Debug exception point.	RW	0

2.3.8.4 Debug Save Register (CP0 Register 31, Select 0)

Software-only register, with no hardware effect. Provided because the debug exception handler can't use the K0-1 GP register, used by ordinary exception handlers to bootstrap themselves: but a debug handler can save a GPR into *DESAVE*, and then use that GPR register code which saves everything else.

DESAVE Register

31	0
DESAVE	

Name	Bits	Description	R/W	Reset
DESAVE	31:0	Debug exception save contents.	RW	0

2.3.8.5 WatchLo Register (CP0 Register 18, Select 0)

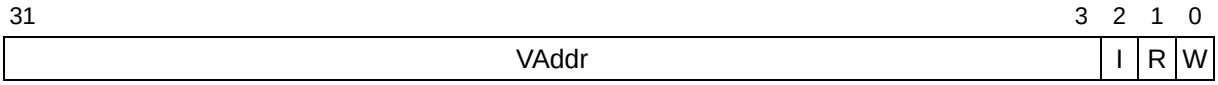
The *WatchLo* and *WatchHi* registers together provide the interface to a watchpoint debug facility which initiates a watch exception if an instruction or data access matches the address specified in the registers. Watch exceptions are taken only if the EXL and ERL bits are zero in the *Status* register. If either bit is a one, the WP bit is set in the *Cause* register, and the watch exception is deferred until both the EXL and ERL bits are zero.

An implementation may provide zero or more pairs of *WatchLo* and *WatchHi* registers, referencing them via the select field of the MTC0/MFC0 instructions, and each pair of Watch registers may be dedicated to a particular type of reference. Software may determine if at least one pair of *WatchLo* and *WatchHi* registers are implemented via the *WR* bit of the Config1 register.

The *WatchLo* register specifies the base virtual address and the type of reference (instruction fetch, load, store) to match. If a particular Watch register only supports a subset of the reference types, the unimplemented enables must be ignored on write and return zero on read. Software may determine which enables are support by a particular Watch register pair by setting all three enables bits and reading them back to see which ones were actually set.

It is implementation dependent whether a data which is triggered by a prefetch, CACHE, or SYNCI instruction whose address matches the Watch register address mach conditions.

WatchLo Register



Name	Bits	Description	R/W	Reset
VAddr	31:3	This field specifies the virtual address to match. Note that this is a double word address, since bits [2:0] are used to control the type of match.	R/W	0x0
I	2	If this bit is set, watch exceptions are enabled for instruction fetches that match the address and are actually issued by the processor(speculative instructions never cause Watch exceptions).	R/W	0
R	1	If this bit is set, watch exceptions are enabled for loads that match the address.	R/W	0
W	0	If this bit is set, watch exceptions are enabled for stores that match the address.	R/W	0

2.3.8.6 WatchHi Register (CP0 Register 19, Select 0)

The *WatchLo* and *WatchHi* register together provide the interface to a watchpoint debug facility that initiates a watch exception if an instruction or data access matches the address specified in the register. As such, they duplicate some functions of the EJTAG debug solution. Watch exception are taken only if the *EXL* and *ERL* bits are zero in the *Status* register. If either bit is a one, then the *WP* bit is set in the *Cause* register, and the watch exception is deferred until both the *EXL* and *ERL* are zero.

The *WatchHi* register contains information that qualifies the virtual address specified in the *WatchLo* register: an ASID, a Global (G) bit, and an optional address mask. If the G bit is 1, any virtual address reference that matches the specified address will cause a watch exception. If the G bit is a 0, only those virtual address references for which the ASID value in the*WatchHi* register matches the ASID value in the *EntryHi* register cause a watch exception. The optional mask field provides address masking to qualify the address specified in *WatchLo*.

WatchHi Register

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
M	G	Reserved	EAS	ASID	Reserved	MASK	I	R	W

Name	Bits	Description	R/W	Reset
M	31	Another pair of *WatchHi/WatchLo* registers is not implemented. Must be written as zero; returns zero on read	R	0
G	30	If this bit is one, any address that matches that specified in the *WatchLo* register causes a watch exception. If this bit is zero, the ASID field of the *WatchHi* register must match the ASID field of the *EntryHi* register to cause a watch exception.	RW	0
Reserved	29:26	Must be written as zero; returns zero on read	R	0
EAS	25:24	Config4$_{AE}$=0 then must be written as zero; returns zero on read.	R	0
ASID	23:16	ASID value which is required to match that in the *EntryHi* register if the G bit is zero in the *WatchHi* register.	RW	0x0
Reserved	15:12	Must be written as zero; returns zero on read	R	0
Mask	11:3	Any bit in this field that is a one inhibits the corresponding address bit from participating in the address match. Software may determine how many mask bits are implemented by writing ones the this field and then reading back the result.	RW	0x0
I	2	This bit is set by hardware when an instruction fetch condition matches the values in this watch register pair. When set, the bit remains set until cleared by software, which is accomplished by writing a 1 to the bit.	W1C	0
R	1	This bit is set by hardware when an load condition matches the values in this watch register pair.	W1C	0

		When set, the bit remains set until cleared by software, which is accomplished by writing a 1 to the bit.		
W	0	This bit is set by hardware when an store condition matches the values in this watch register pair. When set, the bit remains set until cleared by software, which is accomplished by writing a 1 to the bit.	WIC	0

2.3.9 User Mode Support Registers

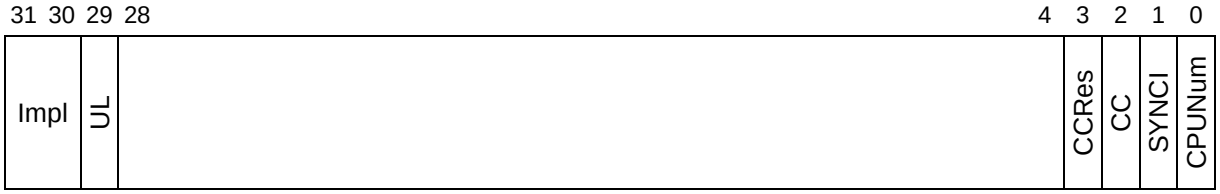
This section contains the following hardware access registers.

- Section 2.3.9.1," Hardware Enable-HWREna (CP0 Register 7, Select 0)"
- Section 2.3.9.2," Load Linked Address (CP0 Register 17, Select 0)"

2.3.9.1 Hardware Enable-HWREna (CP0 Register 7, Select 0)

The *HWREna* register contains a bit mask that determine which hardware register are accessible via the RDHWR instruction when that instruction is executed in a mode in which coprocessor 0 is not enable.

HWREna Register



Name	Bits	Description	R/W	Reset
Impl	31:30	These bits enable access to the implementation-dependent hardware register 31 and 30. If a register is not implemented, the corresponding bit returns a zero and is ignored on write. If a register is implemented , access to that register is enabled if the corresponding bit in this field is a 1 and disable if the corresponding bit is a 0.	R	0
Mask	29:0	Each bit in this field enables access by the RDHWR instruction to a particular hardware register(which may not be an actual register). If RDHWR register 'n' is not implemented , bit 'n' of this field returns a zero and is ignored on a write. If RDHWR register 'n' is implemented, access to the register is enable if bit 'n' in this field is a 1 and disable if bit 'n' of this field is a 0. See the RDHWR instruction for a list of valid hardware registers. Below Table list the RDHWR registers, and register number 'n' corresponds to bit 'n' is this field.	RW	0

Table 2.6 RDHWR Register Numbers

Register Number	Mnemonic	Description
0	CPUNum	Number of the CPU on which the program is currently running. This register provides read access to the coprocessor 0 $EBase_{CPUNum}$ field.

<table>
<tr><td>1</td><td>SYNCI_Step</td><td>Address step size to be used with the SYNCI instruction. See that instruction's description for the use of this value. In the typical implementation, this value should be zero if there are no caches in the system which must be synchronize(either because there no cahches, or because the instruction cache tracks writes to the data cache). In other cases, the return value should be the smallest line size of the caches that must be synchronize.</td></tr>
<tr><td>2</td><td>CC</td><td>High-resolution cycle counter. This register provides read access to the coprocessor 0 Count register.</td></tr>
<tr><td>3</td><td>CCRes</td><td>Resolution of the CC register. This value denotes the number of cycles between update of the register. For example:
<table>
<tr><th>CCRes Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>CC register increments every CPU cycle</td></tr>
<tr><td>2</td><td>CC register increments every second CPU cycle</td></tr>
<tr><td>3</td><td>CC register increments every third CPU cycle.</td></tr>
<tr><td colspan="2">etc</td></tr>
</table></td></tr>
<tr><td>4-31</td><td>Reserved</td><td>These register numbers are reserved for future architecture use. Access result in a Reserved Instruction Exception.</td></tr>
</table>

Using the *HWREna* register, privileged software may select which of the hardware register are accessible via the RDHWR instruction. In doing so, a register may be virtualized at the cost of handling a Reserved Instruction Exception, interpreting the instruction , and returning the virtualized value. For example, if it is not desirable to provide direct access to the *Count* register, access to that register may be individually disabled and the return value can be virtualized by the operating system.

Software may determine which register are implemented by writing all ones to the *HWREna* register, then reading the value back .If a bit reads back as a one, the processor implements that hardware register.

2.3.9.2 Load Linked Address (CP0 Register 17, Select 0)

The *LLAddr* register contains the relevant bits of the physical address read by the most recent Load Linked instruction. This register is implementation dependent and for diagnostic purposes only and serves on function during normal operation.

Load Linked Address Register

31	28 27	0
PAddr		

Name	Bits	Description	R/W	Reset
PAddr	31:0	This field encodes the physical address read by the most recent Load Linked instruction. The format of this register is implementation dependent, and an implementation may implement as many of the bits or format the address in any way that it find convenient.	R	0x0

2.3.10 Kernel Mode Support Registers

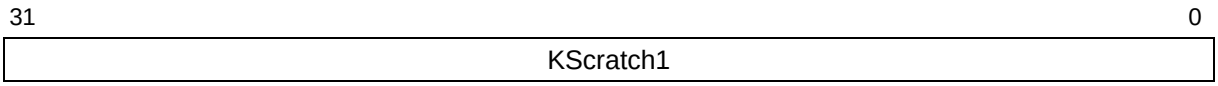
This section contains the following hardware access registers.

- Section 2.3.10.1,"Kernel Scratch Register 1 - KScratch1 (CP0 Register 31, Select 2)"
- Section 2.3.10.2,"Kernel Scratch Register 2 - KScratch2 (CP0 Register 31, Select 3)"
- Section 2.3.10.3,"Kernel Scratch Register 3 - KScratch2 (CP0 Register 31, Select 4)"
- Section 2.3.10.4,"Kernel Scratch Register 4 - KScratch2 (CP0 Register 31, Select 5)"
- Section 2.3.10.5,"Kernel Scratch Register 5 - KScratch2 (CP0 Register 31, Select 6)"
- Section 2.3.10.6,"Kernel Scratch Register 6 - KScratch2 (CP0 Register 31, Select 7)"

2.3.10.1 Kernel Scratch Register 1 - KScratch1 (CP0 Register 31, Select 2)

KScratch1 is a read-write 32-bit register that is used by the kernel for temporary storage of information. The presence of the *KScrathc1* register is indicated by $Config4_{KScrExist[2]}$=1'b1.

KScratch1 Register



Name	Bits	Description	R/W	Reset
KScratch 1	31:0	Used by the kernel for temporary storage of information.	RW	0

2.3.10.2 Kernel Scratch Register 2 - KScratch2 (CP0 Register 31, Select 3)

KScratch2 is a read-write 32-bit register that is used by the kernel for temporary storage of information. The presence of the *KScrathc2* register is indicated by $Config4_{KScrExist[3]}$=1'b1.

KScratch2 Register

31	0
KScratch1	

Name	Bits	Description	R/W	Reset
KScratch 1	31:0	Used by the kernel for temporary storage of information.	RW	0

2.3.10.3 Kernel Scratch Register 3 - KScratch3 (CP0 Register 31, Select 4)

KScratch1 is a read-write 32-bit register that is used by the kernel for temporary storage of information. The presence of the *KScrathc3* register is indicated by $Config4_{KScrExist[4]}$=1'b1.

KScratch3 Register

31	0
KScratch1	

Name	Bits	Description	R/W	Reset
KScratch 1	31:0	Used by the kernel for temporary storage of information.	RW	0

2.3.10.4 Kernel Scratch Register 4 - KScratch4 (CP0 Register 31, Select 5)

KScratch4 is a read-write 32-bit register that is used by the kernel for temporary storage of information. The presence of the *KScrathc4* register is indicated by $Config4_{KScrExist[5]}$=1'b1.

KScratch4 Register

31	0
KScratch1	

Name	Bits	Description	R/W	Reset
KScratch 4	31:0	Used by the kernel for temporary storage of information.	RW	0

2.3.10.5 Kernel Scratch Register 5 - KScratch5 (CP0 Register 31, Select 6)

KScratch5 is a read-write 32-bit register that is used by the kernel for temporary storage of information. The presence of the *KScrathc5* register is indicated by $Config4_{KScrExist[6]}$=1'b1.

KScratch5 Register

31	0
KScratch1	

Name	Bits	Description	R/W	Reset
KScratch 5	31:0	Used by the kernel for temporary storage of information.	RW	0

2.3.10.6 Kernel Scratch Register 6 - KScratch6 (CP0 Register 31, Select 7)

KScratch6 is a read-write 32-bit register that is used by the kernel for temporary storage of information. The presence of the *KScrathc6* register is indicated by $Config4_{KScrExist[7]}$=1'b1.

KScratch6 Register

31	0
KScratch1	

Name	Bits	Description	R/W	Reset
KScratch 6	31:0	Used by the kernel for temporary storage of information.	RW	0

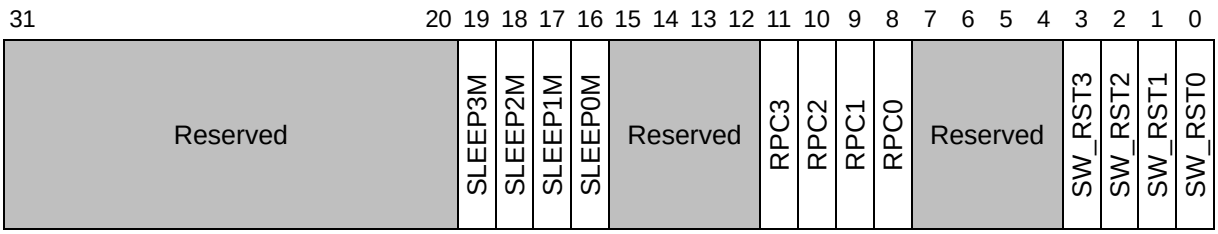
2.3.11 Multi-core Registers

This section's registers is used for multi-core system.

This section contains the following hardware access registers.

2.3.11.1 CoreCtrl Register (CP0 Register 11, Select 6)

CoreCtrl Register



Name	Bits	Description	R/W	Reset
Reserved	31:20	Must be written as zero; returns zero on read	R	0
SLEEP3M	19	Mask CORE3's clock down. 0 – not mask; 1 – mask	RW	0
SLEEP2M	18	Mask CORE2's clock down. 0 – not mask; 1 – mask	RW	0
SLEEP1M	17	Mask CORE1's clock down. 0 – not mask; 1 – mask	RW	0
SLEEP0M	16	Mask CORE0's clock down. 0 – not mask; 1 – mask	RW	0
Reserved	15:12	Must be written as zero; returns zero on read	R	0
RPC3	11	Reset exception handler entry for CORE3. 1 – Dedicated reset entry; 0 – default reset entry (0xBFC00000)	RW	0
RPC2	10	Reset exception handler entry for CORE2. 1 – Dedicated reset entry; 0 – default reset entry (0xBFC00000)	RW	0

RPC1	9	Reset exception handler entry for CORE1. 1 – Dedicated reset entry; 0 – default reset entry (0xBFC00000)	RW	0
RPC0	8	Reset exception handler entry for CORE0. 1 – Dedicated reset entry; 0 – default reset entry (0xBFC00000)	RW	0
Reserved	7:4	Must be written as zero; returns zero on read	R	0
SW_RST3	3	Software reset CORE3. 1 – keeps reset; 0 – not SW reset status	RW	0
SW_RST2	2	Software reset CORE2. 1 – keeps reset; 0 – not SW reset status	RW	0
SW_RST1	1	Software reset CORE1. 1 – keeps reset; 0 – not SW reset status	RW	0
SW_RST0	0	Software reset CORE0. 1 – keeps reset; 0 – not SW reset status	RW	0

2.3.11.2 CoreStat Register (CP0 Register 11, Select 7)

CoreStat Register

31	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		SLEEP3	SLEEP2	SLEEP1	SLEEP0	Reserved				IRQ3_P	IRQ2_P	IRQ1_P	IRQ0_P	Reserved				MIRQ3_P	MIRQ2_P	MIRQ1_P	MIRQ0_P

Name	Bits	Description	R/W	Reset
Reserved	31:20	Must be written as zero; returns zero on read	R	0
SLEEP3	19	Sleep status of CORE3. 0 – not sleep; 1 – sleep	R	0
SLEEP2	18	Sleep status of CORE2. 0 – not sleep; 1 – sleep	R	0
SLEEP1	17	Sleep status of CORE1. 0 – not sleep; 1 – sleep	R	0
SLEEP0	16	Sleep status of CORE0. 0 – not sleep; 1 – sleep	R	0
Reserved	15:12	Must be written as zero; returns zero on read	R	0
IRQ3_P	11	Pending peripheral IRQ to CORE3 when the peripheral IRQ exists meanwhile it is not masked. Software can only read the bit field. 1 – pending peripheral IRQ; 0 – no peripheral IRQ *1	R	0
IRQ2_P	10	Pending peripheral IRQ to CORE2 when the peripheral IRQ exists meanwhile it is not masked. Software can only read the bit field. 1 – pending peripheral IRQ; 0 – no peripheral IRQ *1	R	0
IRQ1_P	9	Pending peripheral IRQ to CORE1 when the peripheral IRQ exists meanwhile it is not masked. Software can only read the bit field. 1 – pending peripheral IRQ; 0 – no peripheral IRQ *1	R	0
IRQ0_P	8	Pending peripheral IRQ to CORE0 when the peripheral IRQ exists meanwhile it is not masked. Software can only read the bit field. 1 – pending peripheral IRQ;	R	0

		0 – no peripheral IRQ *1		
Reserved	7:4	Must be written as zero; returns zero on read	R	0
MIRQ3_P	3	Pending mailbox IRQ3 due to software writing mailbox3 register. Software can clear it to value 0 but setting value 1 has no effect. 1 – writing mailbox0 occurred; 0 – no mailbox0 IRQ *2	RW	0
MIRQ2_P	2	Pending mailbox IRQ2 due to software writing mailbox2 register. Software can clear it to value 0 and setting value 1 has no effect. 1 –writing mailbox1 occurred; 0 – no mailbox1 IRQ *2	RW	0
MIRQ1_P	1	Pending mailbox IRQ1 due to software writing mailbox1 register. Software can clear it to value 0 but setting value 1 has no effect. 1 – writing mailbox0 occurred; 0 – no mailbox0 IRQ *2	RW	0
MIRQ0_P	0	Pending mailbox IRQ0 due to software writing mailbox0 register. Software can clear it to value 0 and setting value 1 has no effect. 1 –writing mailbox1 occurred; 0 – no mailbox1 IRQ *2	RW	0

2.3.11.3 CoreIRQM register (CP0 Register16, Select6)

CoreIRQM Register

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
ENTRY																Reserved				IRQ3_M	IRQ2_M	IRQ1_M	IRQ0_M	Reserved				MIRQ3_M	MIRQ2_M	MIRQ1_M	MIRQ0_M

Name	Bits	Description	R/W	Reset
Entry	31:16	Reset exception entry for a CORE when its RPC is set value 1. For example, when RPC1==1, and Entry == 0x9030, then the reset exception entry for CORE1 is 0x90300000.	RW	0
Reserved	15:12	Writing has no effect, read as zero.	R	0
IRQ3_M	11	Mask of IRQ from INTC to CORE3. 0 – mask; 1 – not mask	RW	0
IRQ2_M	10	Mask of IRQ from INTC to CORE2. 0 – mask; 1 – not mask	RW	0
IRQ1_M	9	Mask of IRQ from INTC to CORE1. 0 – mask; 1 – not mask	RW	0
IRQ0_M	8	Mask of IRQ from INTC to CORE0. 0 – mask; 1 – not mask	RW	0
Reserved	7:4	Writing has no effect, read as zero.	R	0
MIRQ3_M	3	Mask of mailbox IRQ3. 0 – mask; 1 – not mask	RW	0
MIRQ2_M	2	Mask of mailbox IRQ2. 0 – mask; 1 – not mask	RW	0
MIRQ1_M	1	Mask of mailbox IRQ1. 0 – mask; 1 – not mask	RW	0
MIRQ0_M	0	Mask of mailbox IRQ0. 0 – mask; 1 – not mask	RW	0

2.3.11.4 Mailbox0 Register(CP0 Register 22,select 0)

CP0 register 22 is reserved for implementation dependent use and is not defined by the architecture.

Mailbox0 Register

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
Mailbox0																															

Name	Bits	Description	R/W	Reset
MSG0	31:0	Message to CORE0. The action of Writing CORE_MBOX0 rises a positive pulse IRQ which can be automatically latched into CORE_Status.MIRQ0_P	RW	0

2.3.11.5 Mailbox1 Register(CP0 Register 22,select 1)

CP0 register 22 is reserved for implementation dependent use and is not defined by the architecture.

Mailbox1 Register

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
Mailbox1																															

Name	Bits	Description	R/W	Reset
MSG1	31:0	Message to CORE1. The action of Writing CORE_MBOX1 rises a positive pulse IRQ which can be automatically latched into CORE_Status.MIRQ1_P	RW	0

2.3.11.6 Mailbox2 Register(CP0 Register 22,select 2)

CP0 register 22 is reserved for implementation dependent use and is not defined by the architecture.

Mailbox2 Register

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
Mailbox2																															

Name	Bits	Description	R/W	Reset
MSG2	31:0	Message to CORE2. The action of Writing CORE_MBOX2 rises a positive pulse IRQ which can be automatically latched into CORE_Status.MIRQ2_P	RW	0

2.3.11.7 Mailbox3 Register(CP0 Register 22,select 3)

CP0 register 22 is reserved for implementation dependent use and is not defined by the architecture.

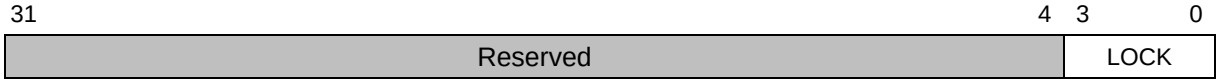
Mailbox3 Register

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
Mailbox3																															

Name	Bits	Description	R/W	Reset
MSG3	31:0	Message to CORE3. The action of Writing CORE_MBOX3 rises a positive pulse IRQ which can be automatically latched into CORE_Status.MIRQ3_P	RW	0

2.3.11.8 SpinLock Register (CP0 Register 9, Select 6)

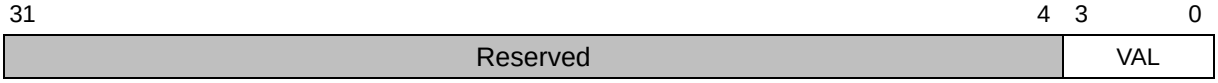
SpinLock Register



Name	Bits	Description	R/W	Reset
Reserved	31:4	Must be written as zero; returns zero on read	R	0
LOCK	3:0	Lock value. Software can only clear it to zero, writing non-zero value to it is ignored.	RW	0

2.3.11.9 SpinAtom Register (CP0 Register 9, Select 7)

SpinAtom Register



Name	Bits	Description	R/W	Reset
Reserved	31:4	Must be written as zero; returns zero on read	R	0
VAL	3:0	For write operation, the value updating the register can update SPINLOCK at the same time if the value in SPINLOCK equals zero. Therefore, an atomic writing-nonzero-to-spinlock operation can be performed.	RW	0

3 Exceptions

3.1 Overview

This section describes how XBurst®1 core handles exceptions.

3.2 Exception Priorities

More than one exception event could occur simultaneously, where, the exception with the highest priority will be accepted by XBurst®1 CPU. The exception priorities are fixed as illustrated by the table below:

Table 3.1 XBurst®1 Core Exception Priorities

Exception types	Exception Events	Exception Priorities
Reset	Reset or NMI	0 (highest)
D-fault	AdEL AdES TLBL TLBS TLB Mod MCheck	1
Dbrk	Debug data break	2
Trap Ov	Trap Overflow	3
CPU RI Sys DBp Brk	Co-processor unusable Reserved instruction SYSCALL instruction BREAK instruction Debug SDBBP instruction	4
Dss	Debug single step	5
I-fault	AdEL TLBL	6
Ibrk Dint	Debug Instruction Breakpoint or watchpoint Debug interrupt	7
INT	Hardware Interrupt or software interrupt	8 (lowest)

3.3 Exception Categories

The exception categories supported by XBurst® CPU Core are described below:

- **RESET**: Reset exceptions induced by
  - A reset.
  - Non-masking interrupt (NMI) request from external input pin.

  Check CP0.status.nmi to identify NMI.
- **INT**: Interrupt exceptions induced by
  - Hardware interrupt request issued by an interrupt controller (INTC) out of CPU core.
  - Soft interrupt request by writing CP0.cause.ip[1:0] directly.
- **D-fault**: Data access fault exceptions induced by
  - Executing a data access instruction. Check CP0.cause.exccode for specific causes such as TLB refill, TLB invalid, address error, TLB modified.
  - Machine check. (caused by TLBWI or TLBWR instruction)
- **I-fault**: Instruction fetch fault exceptions induced by
  - An instruction fetch. Check CP0.cause.exccode for specific causes such as TLB refill, TLB invalid or address error.
- **CpU**: Co-processor unusable exceptions induced by
  - Executing an implemented co-processor instruction on the illegal condition.
  - Executing an unimplemented co-processor instruction.
- **RI**: Reserved instruction exception induced by executing a reserved instruction.
- **Brk**: Break exceptions induced by
  - Executing BREAK instruction in non-debug mode.
  - Executing SDBBP instruction in debug mode.
- **DBp**: Debug breaks exception induced by executing SDBBP instruction in non-debug mode.
- **Sys**: Syscall exception induced by executing SYSCALL instruction.
- **Trap**: Trap exception induced by executing TRAP instructions with true condition.
- **Ov**: Overflow exception induced by executing one of ADD/ADDI/SUB instruction with the result overflowing.
- **Dss:** Debug single step exception induced by each instruction executing (except the one in the branch delay slot) when CP0.debug.sst is set value 1.
- **Dint:** The debug interrupt exception.
- **Ibrk**: Instruction break exception induced by
  - An instruction fetch address matches the address watch information stored in the *WatchHi* and *WatchLo* registers.
  - An instruction hardware break point triggered.
- **Dbrk**: Data break exception induced by
  - A data access address matches the address watch information stored in the *WatchHi* and *WatchLo* registers.
  - A data hardware breakpoint triggered.

3.4 Cause Code of Exception

Table 3.2 Cause.ExcCode Field Descriptions

Cause Code	Mnemonic	Description
0	Int	Interrupt
1	Mod	TLB modification exception
2	TLBL	TLB exception (load or instruction fetch)
3	TLBS	TLB exception (store)
4	AdEL	Address error exception (load or instruction fetch)
5	AdES	Address error exception (store)
6	-	Reserved
7	-	Reserved
8	Sys	Syscall exception
9	Bp	Breakpoint exception
10	RI	Reserve red instruction exception
11	CpU	Coprocessor unusable exception
12	Ov	Integer overflow exception
13	Tr	Trap exception
14	-	Reserve red
15	FPE	Floating point exception
16-22	-	Reserve red
23	WATCH	Reference to WatchHi/WatchLo address
24	Mcheck	Machine Check
25-31	-	Reserve red

3.5 Exception Handler Entry

Table 3.3 XBurst®1 core Exception Handler Entries

Exception	StatusBEV	StatusEXL	CauseIV	DCRProb	Entry Address
Reset, NMI	-	-	-	-	**0Xbfc0_0000**
Debug exception	-	-	-	0	**0xbfc0_0480**
				1	**0xff20_0200**
TLB Refill	0	0	-	-	**0x8000_0000**
		1			**0x8000_0180**
	1	0	-	-	**0xbfc0_0200**
		1			**0xbfc0_0380**
Interrupt	0	-	0	-	**0x8000_0180**
			1		**0x8000_0200**
	1		0	-	**0xbfc0_0380**
			1		**0xbfc0_0400**
Others	0	-	-	-	**0x8000_0180**
	1				**0xbfc0_0380**

3.6 Exception Handling Process

3.6.1 Enter Exception Handler Routine

It takes several CPU clock cycles to switch from the current context to the exception handler routine.

XBurst®1 CPU will perform following tasks after one kind of exceptions is granted.

- Calculate correlative handler entry and cause code in terms of granted exception type and current core context.
- Set correct return address value to ErrorPC for granted reset exception or to EPC for granted generic exception, or to DEPC for granted debug or debug mode exception.
- Set necessary status bits such as CP0statusexl, CP0debugdm and so on.
- After above all has been done, switching core context is accomplished CPU then let PC jump to the expected handler entry for service of granted exception.

3.6.2 Return from Exception Handler Routine

Return from exception routine is performed by executing ERET instruction for non-debug exceptions or DERET instruction for debug/debug mode exceptions.

4 Branch Target Buffer

4.1 Overview

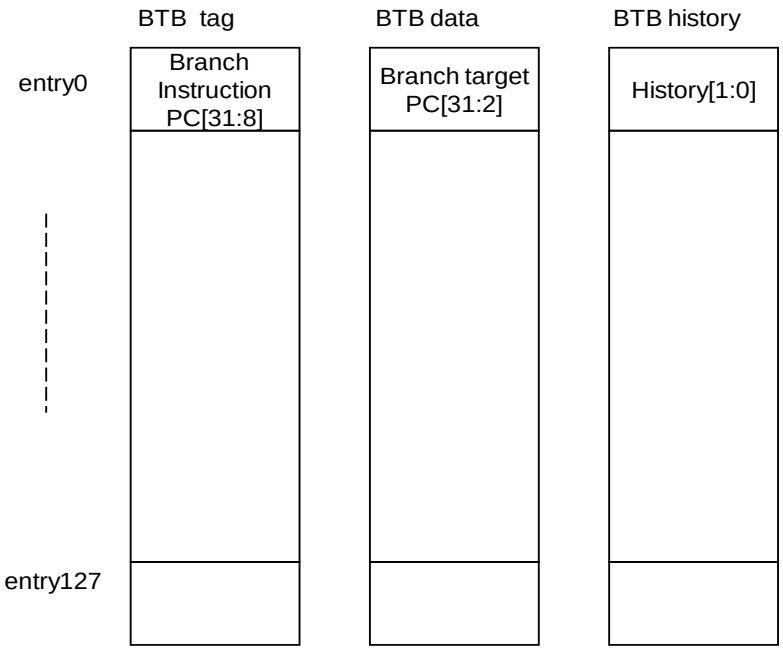
XBurst®1 core implements a 128 entries direct-mapped Branch Target Buffer (BTB). PCs of recently being executed BC (branch conditional) instructions or JMP (branch unconditional) instructions will always be registered into BTB entries. Thus when such an instruction is fetched again later, its associated PC may hit one of BTB entries, so that BTB can predict the branch direction and the branch target PC after the instruction. So if

1. Prediction is right after the branch instruction is executed, there is no pipeline penalty because no instruction of the false branch has been fetched.
2. Prediction is wrong after the branch instruction is executed, CPU has to discard the false branch that has already streamed into pipeline and then fetch correct branch. So several pipeline penalties (3 ~ 4 cycles) are induced.

Moreover, when the PC of a branch instruction does not hit BTB, Then if

1. Branch is untaken after the BC instruction is executed due to false condition, there is no pipeline penalty.
2. Branch is taken after the BC instruction is executed due to true condition, or branch is taken after the JMP instruction is executed, there are 3 cycles penalty.

Therefore, as long as the correctness of prediction is larger than 50%, the total branch penalties are less than the implementation without BTB. In practice, the ratio of correctness for such predication is about 70~90%. BTB structure is as follow.



4.2 BTB Registers

Cp0 contains a dedicated register Conig7 to control the BTB Refer to Config7 Register.

5 MMU

5.1 Overview

XBurst®1 core has an on-chip memory management unit (MMU) that implements address translation. The MMU features a resident translation look-aside buffer (TLB) that caches information for user-created address translation tables located in external memory. It enables high-speed translation of virtual addresses into physical addresses. Address translation uses the paging system and supports 7 kinds of page size. The access right to virtual address space can be set for privileged and user modes to provide memory protection.

5.2 Virtual Memory Map

XBurst®1 core uses 32-bit virtual addresses to access a 4-Gbyte virtual address space that is divided into several areas according to different operation mode. Address space mapping is shown in following figure.

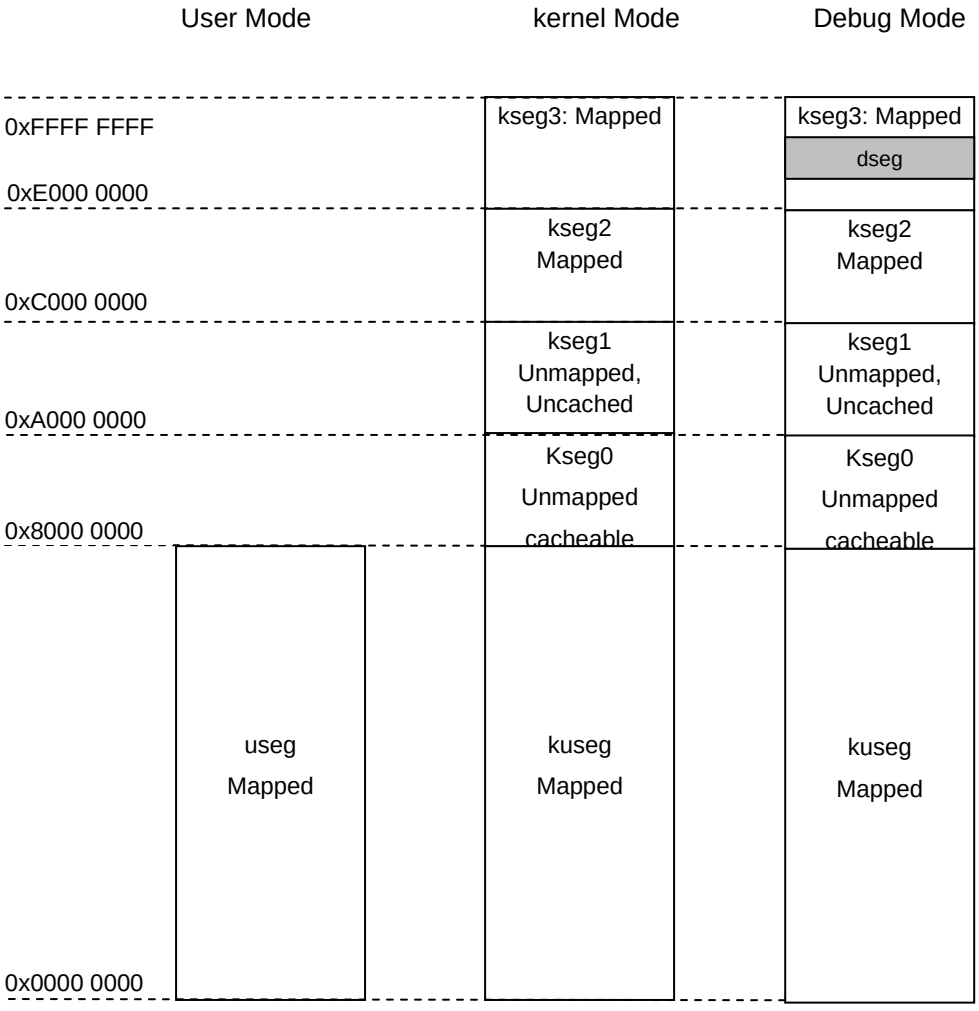


Figure 5.1 Virtual Memory Map

5.2.1 User Mode

The processor operates in user mode when the DM bit in the Debug register is 0 and the Status register contains the following bit values:

- UM = 1
- EXL = 0
- ERL = 0

The user segment useg starts at address 0x0000_0000 and ends at address 0x7FFF_FFFF. Accesses to all other addresses cause an address error exception.

The system maps all references to useg through the TLB. For XBurst1 core, the virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address before translation. Bit settings within the TLB entry for the referenced page determine its cacheability.

5.2.2 Kernel Mode

The processor operates in kernel mode when the DM bit in the *Debug* register is 0 and the *Status* register contains the following values:

- ERL = 1 or EXL=1, in spite of value of UM
- UM = 0, in spite of value of EXL and ERL

When a non-debug exception is detected, EXL or ERL will be set and the processor will enter kernel mode. At the end of the exception handler routine, an Exception Return (ERET) instruction is generally executed. The ERET instruction jumps to the Exception PC, clears ERL, and clears EXL if ERL=0. This may return the processor to user mode.
Kernel mode virtual address space is divided into regions differentiated by the high-order bits of the virtual address.

5.2.2.1 kuseg

kuseg address range is 0x0000_0000 - 0x7FFF_FFFF (2G byte). The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address.
When ERL = 1 in the *Status* register, the user address region becomes a 2^{31}-byte unmapped and uncached address space. While in this setting, the kuseg virtual address maps directly to the same physical address, and does not include the ASID field.

5.2.2.2 kseg0

In kernel mode, when the most-significant three bits of the virtual address are 100_2, 32-bit kseg0 virtual address space is selected; it is the 2^{29}-byte (512-MByte) kernel virtual space located at addresses 0x8000_0000 - 0x9FFF_FFFF. References to kseg0 are unmapped; the physical address selected is defined by subtracting 0x8000_0000 from the virtual address. The K0 field of the *Config* register controls its cacheability.

5.2.2.3 kseg1

In kernel mode, when the most-significant three bits of the 32-bit virtual address are 101_2, 32-bit kseg1 virtual address space is selected. kseg1 is the 2^{29}-byte (512-MByte) kernel virtual space located at addresses 0xA000_0000 - 0xBFFF_FFFF. References to kseg1 are unmapped; the physical address selected is defined by subtracting 0xA000_0000 from the virtual address. References to this region are always uncached.

5.2.2.4 kseg2

In kernel mode, when UM = 0 or ERL = 1 or EXL = 1 in the Status register, and DM = 0 in the Debug register, and the most-significant three bits of the 32-bit virtual address are 110_2, 32-bit kseg2 virtual address space is selected.

5.2.2.5 kseg3

In kernel mode, when the most-significant three bits of the 32-bit virtual address are 111_2, the kseg3 virtual address space is selected.

5.3 TLB

The following subsections discuss the TLB memory management scheme used in XBurst1 processor core. The TLB consists of one joint and two micro address translation buffers:

- 32 dual-entry fully associative Joint TLB (JTLB)
- 4-entry fully associative Instruction micro TLB (ITLB)
- 4-entry fully associative Data micro TLB (DTLB)

5.3.1 Joint TLB

XBurst®1 core implements a 32 dual-entry, fully associative Joint TLB that maps 64 virtual pages to their corresponding physical addresses. The JTLB is organized as 32 pairs of even and odd entries containing pages that range in size from 4-KBytes to 16-MBytes into the 4-GByte physical address space. The purpose of the TLB is to translate virtual addresses and their corresponding Address Space Identifier (ASID) into a physical memory address. Because this structure is used to translate both instruction and data virtual addresses, it is referred to as a “joint” TLB.

The JTLB is organized in page pairs to minimize its overall size. Each virtual tag entry corresponds to two physical data entries, an even page entry and an odd page entry. The highest order virtual address bit not participating in the tag comparison is used to determine which of the two data entries is used. Since page size can vary on a page-pair basis, the determination of which address bits participate in the comparison and which bit is used to make the even-odd determination must be determined dynamically during the TLB lookup.

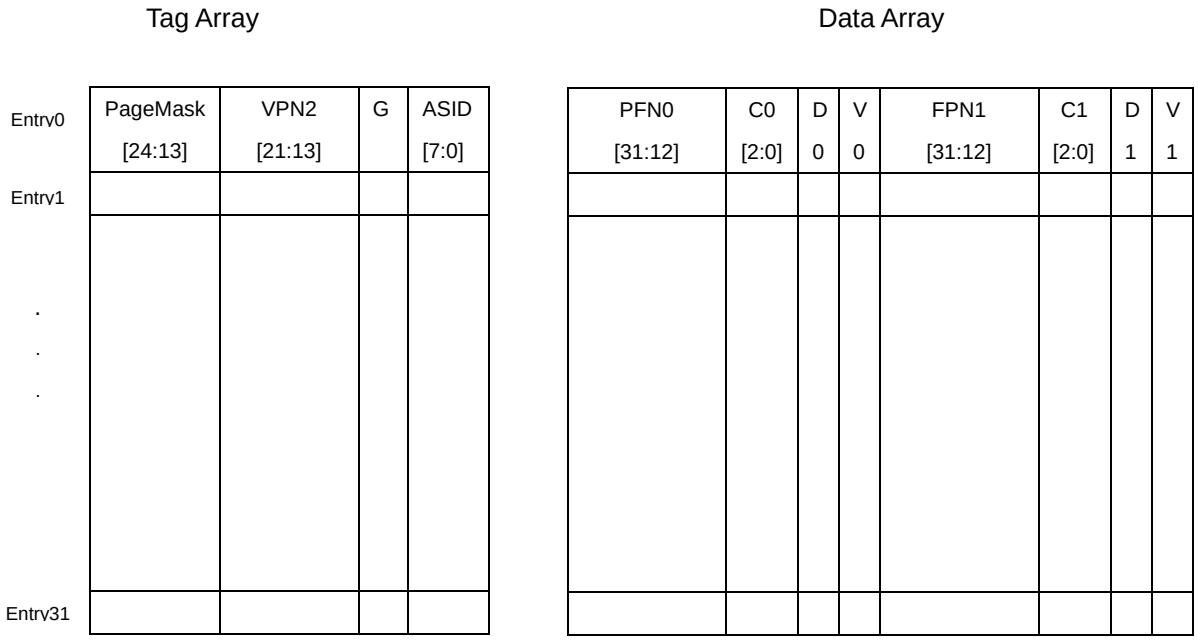


Figure 5.2 Structure of Joint TLB

Table 5.1 TLB Tag Entry Fields

Field Name	Description
PageMask[24:13]	Mask bits for varying page size 0000_0000_0000: 4KB 0000_0000_0011: 16KB 0000_0000_1111: 64KB 0000_0011_1111: 256KB 0000_1111_1111: 1MB 0011_1111_1111: 4MB 1111_1111_1111: 16MB
VPN2[31:13]	Virtual Page Number divided by 2 This field contains the upper bits of the virtual page number Because it represents a pair of TLB pages, it is divided by 2 Bits 31:25 are always included in the TLB lookup comparison Bits 24:13 are included depending on the page size, defined by PageMask
G	Global Bit When set, indicates that this entry is global to all processes and/or threads and thus disables inclusion of the ASID in the comparison
ASID	Address Space Identifier Identifies which process or thread this TLB entry is associated with

Table 5.2 TLB Data Entry Fields

Field Name	Description
PFN0[31:12], PFN1[31:12]	Physical Frame Number Defines the upper bits of the physical address For page sizes larger than 4 KBytes, only a subset of these bits is actually used
C0[2:0], C1[2:0]	Cacheability Contains an encoded value of the cacheability attributes and determines whether the page should be placed in the cache or not The field is encoded as following: 000: Cacheable, write through, no write-allocate 001: Cacheable, write through, no write-allocate 010: Uncacheable 011: Cacheable, write-back, write-allocate 100: Cacheable, write-back, write-allocate 101: Cacheable, write-back, write-allocate 110: Cacheable, write-back, write-allocate 111: Uncacheable
D0, D1	“Dirty” or Write-enable Bit Indicates that the page has been written, and/or is writable If this bit is set, stores to the page are permitted If the bit is cleared, stores to the page cause a TLB Modified exception
V0, V1	Valid Bit Indicates that the TLB entry and, thus, the virtual page mapping are valid If this bit is set, accesses to the page are permitted If the bit is cleared, accesses to the page cause a TLB Invalid exception

In order to fill an entry in the JTLB, software executes a TLBWI or TLBWR instruction. Prior to invoking one of these instructions, several CP0 registers must be preset with the information to be written to a TLB entry.

- *PageMask* is set in the CP0 *PageMask* register.
- VPN2 and ASID are set in the CP0 *EntryHi* register.
- PFN0, C0, D0, V0 and G bit are set in the CP0 *EntryLo0* register.
- PFN1, C1, D1, V1 and G bit are set in the CP0 *EntryLo1* register.

Note that the global bit “G” is part of both *EntryLo0* and *EntryLo1*. The resulting “G” bit in the JTLB entry is the logical AND between the two fields in *EntryLo0* and *EntryLo1*.

The address space identifier (ASID) helps to reduce the frequency of TLB flushing on a context switch. The existence of the ASID allows multiple processes to exist in both the TLB and instruction caches. The ASID value is stored in the *EntryHi* register and is compared to the ASID value of each entry.

5.3.2 Instruction TLB

The ITLB is a small 4-entry, fully associative TLB dedicated to performing translations for the instruction stream. The ITLB only maps 4-Kbyte pages/sub-pages.

The ITLB is managed by hardware and is transparent to software. If a fetch address cannot be translated by the ITLB, the JTLB is accessed to attempt to translate it in the following clock cycle. If successful, the translation information is copied into the ITLB. The ITLB is then re-accessed and the address will be successfully translated. It results in an ITLB miss penalty of 3 cycles.

5.3.3 Data TLB

The DTLB is a small 4-entry, fully associative TLB dedicated to performing translation for load/store addresses. The DTLB only maps 4-Kbyte pages/sub-pages.

Like the ITLB, the DTLB is managed by hardware and is transparent to software. If a load/store address cannot be translated by the DTLB, the JTLB is accessed to attempt to translate it in the following clock cycle. If successful, the translation information is copied into the DTLB. The DTLB is then re-accessed and the address will be successfully translated. It results in a DTLB miss penalty of 3 cycles.

5.4 Virtual to Physical Address Translation

During virtual-to-physical address translation, XBurst1 core compares ASID and, depending on pages size, the highest 8-to-20 bits (VPN) of the virtual address to the contents of the TLB. The following figure illustrates the TLB address translation process.

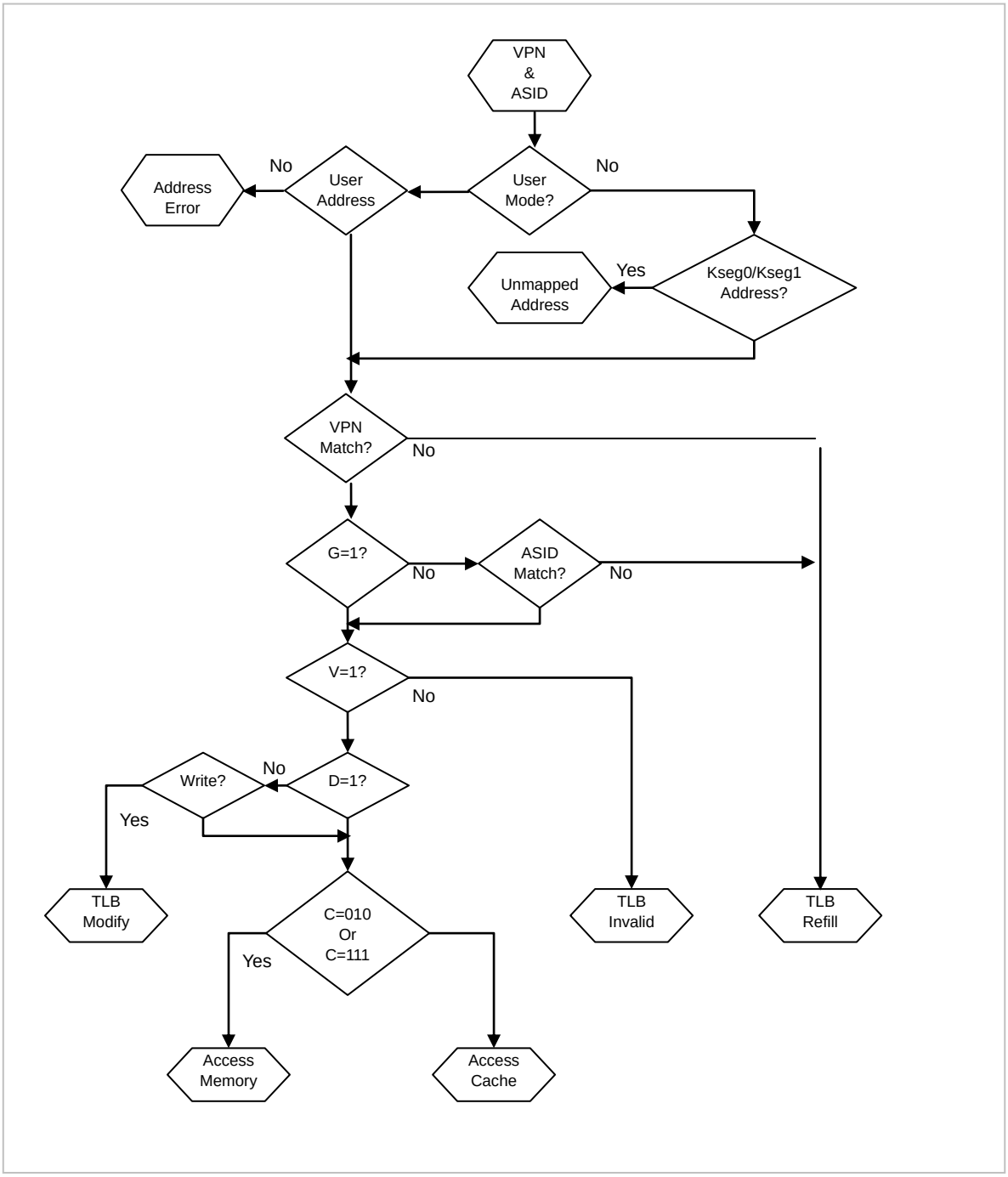


Figure 5.3 XBurst® TLB Address Translation Flow

A virtual address matches content of a TLB entry when the VPN field of the virtual address equals the VPN field of the entry, and either G bit of the TLB entry is set or ASID field of the virtual address (held in the *EntryHi* register) matches the ASID field of the TLB entry.

5.5 MMU Exceptions

5.5.1 TLB Refill Exception

A TLB Refill exception occurs when no TLB entry matches a reference to a mapped address and the CP0statusexl bit is zero.

Table 5.3 Core Context for TLB Refill

Core Context	Value and Description
CP0causeexccode	2- TLBL Reference is a load or an instruction fetch 3- TLBS Reference is a store
CP0badvaddr	failing PC if reference is an instruction fetch failing data access address if reference is a load/store
CP0contextbadvpn2	badvpn2 contains the same content of CP0badvaddr[31:13]
CP0 entryhivpn2	vpn2 contains the same content of CP0badvaddr[31:13]
CP0entryhiasid	need be managed by exception handler routine
CP0epc	PC of the instruction issuing such reference PC of the instruction issuing such reference – 4 (the delay slot case)
CP0statusexl	1 after PC jumps to handler entry

5.5.2 TLB Invalid Exception

A TLB invalid exception occurs when a TLB entry matches a reference to a mapped address, but the matched entry has the valid bit off.

Table 5.4 Core Context for TLB Invalid

Core Context	Value and Description
CP0causeexccode	2- TLBL Reference is a load or an instruction fetch 3- TLBS Reference is a store
CP0badvaddr	failing PC if reference is an instruction fetch failing data access address if reference is a load/store
CP0contextbadvpn2	badvpn2 contains the same content of CP0badvaddr[31:13]
CP0 entryhivpn2	vpn2 contains the same content of CP0badvaddr[31:13]
CP0entryhiasid	need be managed by exception handler routine
CP0epc	PC of the instruction issuing such reference PC of the instruction issuing such reference – 4 (the delay slot case)
CP0statusexl	1 after PC jumps to handler entry

5.5.3 TLB Modify Exception

A TLB modify exception occurs when a TLB entry matches a store reference to a mapped address, but the matched entry is not dirty.

Table 5.5 Core Context for TLB Modify

Core Context	Value and Description
CP0causeexccode	1- TLBMod
CP0badvaddr	failing data access address
CP0contextbadvpn2	badvpn2 contains the same content of CP0badvaddr[31:13]
CP0 entryhivpn2	vpn2 contains the same content of CP0badvaddr[31:13]
CP0entryhiasid	need be managed by exception handler routine
CP0epc	PC of the instruction issuing such reference PC of the instruction issuing such reference – 4 (the delay slot case)
CP0statusexl	1 after PC jumps to handler entry

5.5.4 Address Error Exception

An address error exception occurs in following cases:

1. An instruction fetch or a data load/store accesses a word from a misaligned word address boundary.
2. A data load/store accesses a half-word from a misaligned half-word address boundary.
3. Reference kernel address space in user mode.

Table 5.6 Core Context for Address Error

Core Context	Value and Description
CP0causeexccode	1- TLBMod
CP0badvaddr	failing data access address
CP0contextbadvpn2	badvpn2 contains the same content of CP0badvaddr[31:13]
CP0 entryhivpn2	vpn2 contains the same content of CP0badvaddr[31:13]
CP0entryhiasid	need be managed by exception handler routine
CP0epc	PC of the instruction issuing such reference PC of the instruction issuing such reference – 4 (the delay slot case)
CP0statusexl	1 after PC jumps to handler entry

5.5.5 Machine Check Exception

The machine check exception occurs when executing of TLBWI or TLBWR instruction detects multiple matching entries in the JTLB.

Table 5.7 Core Context for Machine Check Exception

Core Context	Value and Description
CP0causeexccode	24- Machine Check
CP0badvaddr	Ignored
CP0contextbadvpn2	Ignored
CP0 entryhivpn2	Ignored
CP0entryhiasid	Ignored
CP0epc	PC of the instruction issuing such reference PC of the instruction issuing such reference – 4 (the delay slot case)
CP0statusexl	1 after PC jumps to handler entry

5.6 MMU CP0 Registers

CP0 contains a group of registers dedicated for MMU manipulation, configuration or TLB related exceptions. Table 5.8 lists these registers. See Section CP0 for details of these registers.

Table 5.8 CP0 Registers for MMU

CP0 register Number	Register Name	Function
0	Index	Index into the TLB array
1	Random	Randomly generated index into the TLB array
2	EntryLo0	Low-order portion of the TLB entry for even-numbered virtual pages
3	EntryLo1	Low-order portion of the TLB entry for odd-numbered virtual pages
4	Context	Pointer to page table entry in memory
5	PageMask	Controls the variable page sizes in TLB entries
6	Wired	Controls the number of fixed (“wired”) TLB entries
8	BadVaddr	Reports the address for the most recent address related exceptions
10	EntryHi	High-order portion of the TLB entry

5.7 TLB Instructions

This section describes the instructions defined for manipulating TLB contents.

Op Code	Description
TLBP	TLB Probe
TLBR	TLB Read
TLBWI	TLB Write Index
TLBWR	TLB Write Random

5.7.1 TLBP

TLBP is implemented to find a matching entry in the TLB. The *Index* register is loaded with the address of the TLB entry whose contents match the contents of the *EntryHi* register. If no TLB entry matches, the high-order bit of the *Index* register is set.

5.7.2 TLBR

TLBR is implemented to read an entry from the TLB. The *EntryHi*, *EntryLo0*, *EntryLo1*, and *PageMask* registers are loaded with the contents of the TLB entry pointed to by the Index register.

5.7.3 TLBWI

TLBWI is implemented to write a TLB entry indexed by the *Index* register. The TLB entry pointed to by the *Index* register is written from the contents of the *EntryHi*, *EntryLo0*, *EntryLo1*, and *PageMask* registers.

5.7.4 TLBWR

TLBR is implemented to write a TLB entry indexed by the *Random* register. The TLB entry pointed to by the *Random* register is written from the contents of the *EntryHi*, *EntryLo0*, *EntryLo1*, and *PageMask* registers.

6 Caches

6.1 Overview

XBurst®1 core has separate instruction cache (I-Cache) and data cache (D-Cache) that allows instruction and data references to proceed simultaneously Its key features are as following:

Table 6.1 Cache Features

Parameter	I-Cache	D-Cache
Size	16K Bytes	16k Bytes
Line Size	32 Byte	32 Byte
Numbe of Sets	128	128
Associativity	4 way	4 way
Lookup policy	Virtually indexed Physically tagged	Virtually indexed Physically tagged
Replace policy	LRU	LRU
Lock	N/A	N/A
Others	-	16-word deep write buffer

6.2 Cache Coherency Attribute

Cache coherency attribute is specified by the C[2:0] field in EntryLo0 and EntryLo1 entry of the TLB table for mapped address regions useg/kuseg,kseg2 and kseg3. For unmapped segment kseg0, Config.K0[2:0] field specifies the cache attribute. Unmapped segment kseg1 is not cacheable. The cache attribute is defined as following:

Table 6.2 Cache Coherency Attributes

CCA	Encoding	Description
0	000_2	Cacheable, write-through, no write-allocate
1	001_2	UCA (uncacheable accelerate)
2	010_2	Uncacheable
3	011_2	Cacheable, Write-back, write-allocate
4	100_2	Reserved
5	101_2	Reserved
6	110_2	Cacheable, Write-back, write-allocate, streaming
7	111_2	Reserved

6.3 Cache Structure

Both I-Cache and D-Cache uses a 4-way set associative system. It is composed of four ways (banks), each of which is divided into a tag section and a data section. Each of the tag and data sections is divided into 128 entries. An entry of data section is called a cache line. Each cache line consists of 32 bytes (4 bytes). The capacity per way is 4 k bytes (32 bytes 128 entries), with a total of 16 k bytes in the cache as a whole (4 ways).

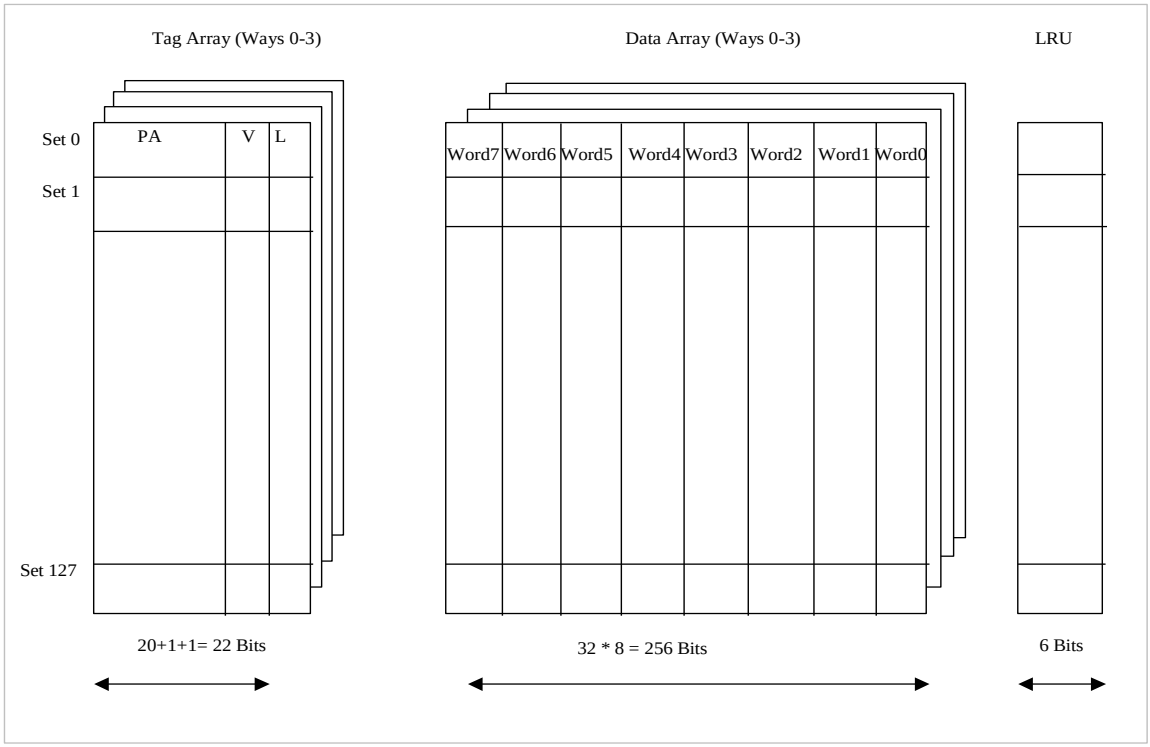


Figure 6.1 I-Cache Structure

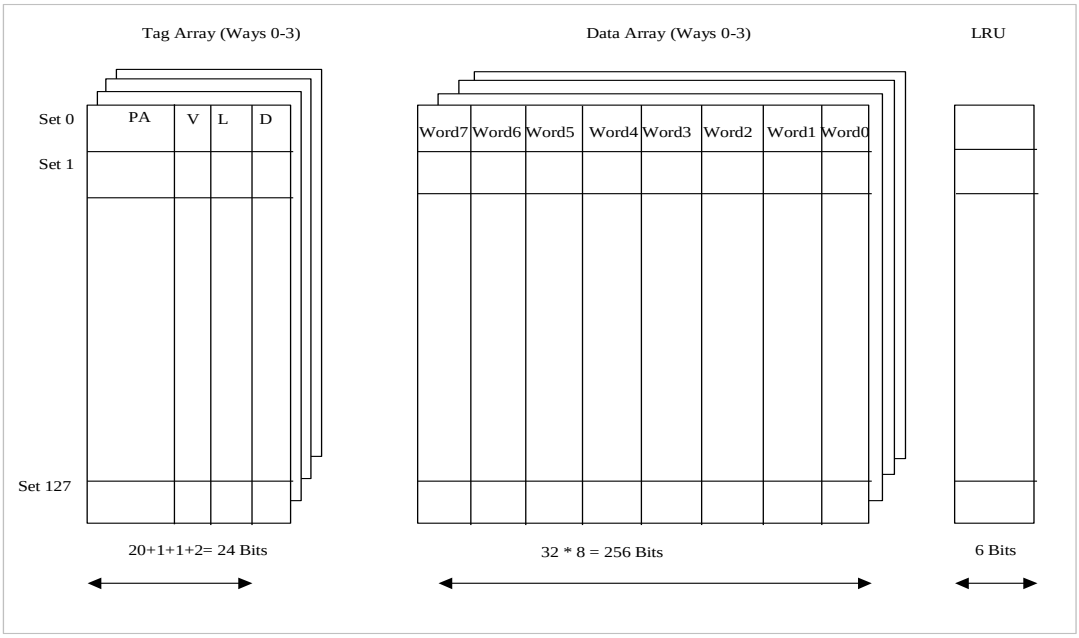


Figure 6.2 D-Cache Structure

As shown in the above figures, the structure of I-Cache and D-Cache is similar, with the exception that the Tag entry of D-Cache contains a 2 bits dirty field D.

TAG Array: The address tag PA holds the physical address used in the external memory access. It is composed of 20 bits (address bits 31–12) used for comparison during cache lookup. The V bit indicates whether the entry data is valid. When the V bit is 1, data is valid; while for value 0, data is not valid. L bit indicates whether the entry is locked (L bits is not implemented in this version). D[1:0] is only implemented in D-Cache. D[0] indicates whether the lower half of the cache line (word0-word3) is dirty. D[1] indicates whether the higher half of the cache line (word4-word7) is dirty. The tag array is not initialized by a system reset.

Data Array: Each entry contains 256 bits (32 bytes).

LRU: With the 4-way set associative system, there are six LRU bits, controlled by hardware. A least-recently-used (LRU) replacement algorithm is used to select the way to be replaced when a cache miss occurs. LRU is transparent to software.

6.4 Write Buffer

There is a 16-word depth write buffer associated with D-Cache.

When a write occurs on a WT (write through) attribute address, the write data will be written to write buffer instead of to external memory. Later the task of write-through-to-memory mandated by write buffer will be performed in the background, that is, such write operation does not block pipeline.
Similarly, when a D-Cache line to be replaced is dirty, the dirty data (lower half line or higher half line or both) will be conveyed to write buffer instead of copying back to external memory immediately. Later the task of copy-back-to-memory mandated by write buffer will be performed in the background.

6.5 Cache Registers

Table 6.3 Cache Registers

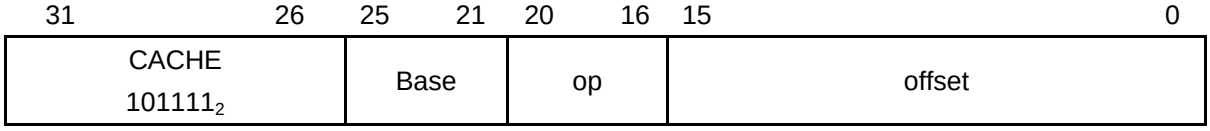
CP0 register Number	Register Name	Function
17	LLAddr	Load linked address
26	ErrCtl	Enable CACHE Index Store Data instruction
28	TagLo/ DataLo	Cache tag/data interface

6.6 Cache Instructions

This section describes the instructions defined for manipulating Cache's contents.

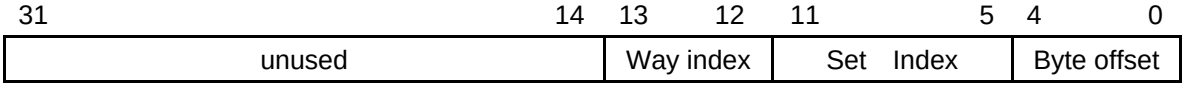
6.6.1 CACHE instruction

CACHE instruction format is shown below:



The 16-bit offset is sign-extended and added to the contents of the base register to form a virtual address. The virtual address is used in one of the following ways based on the operation to be performed and the type of cache as described in the following table. Please note that

- The virtual address need be translated by MMU to form a physical address. The physical address is then used to address the cache.
- The effective address is used to index the sets and ways of the cache, as shown below.



op[17:16] of the Cache instruction specifies the cache on which to perform the operation.

00_2 – I-Cache
01_2 – D-Cache
10_2 – Reserved
11_2 – Reserved

Table 6.4 I-Cache Operations (op[17:16] = 00_2)

Op[20:18]	Operation	Function description
000_2	Index Invalidate	Invalidate an I-cache specified by the virtual address. Virtual address is used to directly index the specified way in specified set.Software. This function can be used by software to invalidate the entire instruction cache by stepping through all valid indices.
001_2	Index Load Tag	Read the tag for the cache block at the specified index into the *TagLo* register. Also read the word corresponding to the word offset(ignore least significant two bits of the address) into the *DataLo* register.
010_2	Index Store Tag	Write the tag for the cache block at the specified index from the *TagLo* register .
011_2	Index Store data	Write the DataLo contents to the way and word index as specified. Note that: This op is valid only when Errctl[WST] is 1.
100_2	Hit Invalidate	If the virtual address hits I-cache, the hit line is invalidated; otherwise, nothing is done.
101_2	Refill a line	Fetch the cache line containing the virtual address from memory and fill it into the I-cache. **NOTE** : The cache line is re-fetched even if it is already in the cache.
110_2	Reserved	--
111_2	Prefecth and lock	If the virtual address misses cache, the line containing the address is fetched from memory.

Table 6.5 D-Cache Operations (op[17:16] = 01_2)

Op[20:18]	Operation	Function description
000_2	Index write back Invalidate	Invalidate a D-cache specified by the virtual address. Virtual address is used to directly index the specified way in specified set. If the cache line is dirty, write back the dirty data and set it invalid. This function can be used by software to invalidate the entire D-cache by stepping through all valid indices.
001_2	Index Load Tag	Read the tag for the cache block at the specified index into the *TagLo* register. Also read the word corresponding to the word index into the *DataLo* register (ignore VA[1:0]).
010_2	Index Store Tag	Write the tag for the cache block at the specified index from the *TagLo* register. This encoding may be used by software to initialize the entire D-cache by stepping through all valid indices. Doing so requires that the *TagLo* and *TagHi* registers associated with the cache be initialized first.
011_2	Index Store data	Write the DataLo contents to the way and word address as specified. Note that: This op is valid only when Errctl[WST] is 1.
100_2	Hit Invalidate	If the virtual address hits D-cache, the hit line is invalidated; otherwise, nothing is done.
101_2	Hit write back Invalidate	If the virtual address hits D-cache, invalidate the hit cache line. If the cache line is dirty, write back the dirty data and set it invalid.
110_2	Hit write back	If D-cache hit and it is dirty, write back dirty data and leave it still valid, but clear the dirty bits Otherwise, treat as NOP.
111_2	Prefecth and lock	If the virtual address misses cache, the line containing the address is fetched from memory, and if the line to be replaced is dirty, write back the dirty data.

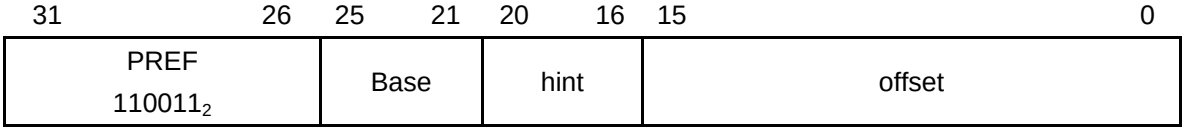
NOTES:

For index operation, software should use unmapped address to avoid TLB exceptions.

For non-index operation, the result is UNDEFINED if the virtual address is uncacheable.

6.6.2 PREF instruction

PREF instruction format is shown below:



The 16-bit offset is sign-extended and added to the contents of the base register to form a virtual address.

PREF does not cause addressing-related exceptions. If the address specified would cause an addressing exception, the PREF operation will be ignored. Similarly, PREF performs nothing if the virtual address is uncacheable.

The hint field supplies information about the manipulation way that the prefetched data to be used. PREF is an advisory instruction that may change the performance of the program. However, for all hint values and all virtual addresses, it neither changes the architecturally visible state nor does it alter the meaning of the program. XBurst1 core just implements two types of hint.

PREF is a non-blocking operation, that is, it does not block pipeline while waiting for the data to be load from external memory.

Table 6.6 Values of the *hint* Field for the PREF Instruction

Hint	Action	Description
0~31 (no 30)	Prefetch	Prefetch data in the same way as cacheable load miss.
30	Allocate	Allocate a cache line, without the overhead of filling the data from memory.

6.6.3 SYNC instruction

31 26	25 11	10 6	5 0
SPECIAL 000000_2	0	stype	SYNC 001111

SYNC is used to synchronize memory hierarchy. It forces all buffered memory access operations (maybe caused by Load/Store instruction, CACHE instruction, PREF instruction) occurred before the execution of SYNC instruction to be accomplished. In other words, SYNC instruction eliminates potential data coherency hazard in the memory hierarchy.

Executing the SYNC instruction causes the write buffer to be flushed. The SYNC instruction blocks pipeline until all prior memory accesses are completed.

7 JTAG Debug

XBurst®1 core supports two modes for JTAG Debug: MIPS mode and ACC mode. MIPS mode is compatible with MIPS specific JTAG protocol of access to dmseg. In ACC mode, XBurst®1 core specific protocol can realize accelerated access to dmseg Moreover, more dmseg space is extended to accommodate larger program and data for JTAG Debug.

7.1.1 ACC Mode Flag

AM: 1 – ACC mode; 0 – MIPS mode.

Its reset value is 0. It is invisible for JTAG probe and software. The bit is set by expanded instruction EJTAGBOOTA, cleared by NORMALBOOT or TAP reset.

7.1.2 EJTAG Control Register in ACC mode (ECR_A)

It is connected between TDI and TDO by instruction CONTROL or ALL in ACC mode. Probe polls this 2-bit register to service the processor access to dmseg region.

Field	BITS	Description	Read/write	Reset value
PA	0	Processor Access (PA) 0: No pending processor access 1: Pending processor access	R	0
PRW	1	Processor access is read or write 0: read access 1: write access	R	Undefined

7.1.3 Processor Access Address Register in ACC mode (ADDRESS_A)

It is connected between TDI and TDO by instruction ADDRESS or ALL in ACC mode. This register is used to provide the accessed address of dmseg region and more data bus operation information.

36 35	34	33 32	31 0
BST	R	SZ	PAA

Field	BITS	Description	R/W	Reset value
PAA	31: 0	Processor Access address	R	Undefined
SZ	33:32	Processor Access size 00: byte 01: half word 10: word 11: reserved	R	Undefined
R	34	Reserved		
BST	36:35	Processor Access burst pattern 00: single 01: 4-beat wrapping burst 10: 8-beat wrapping burst 11: reserved Note: 4-beat wrapping burst will never occur in this implementation. And 8-beat wrapping burst may occur only for burst read access.	R	Undefined

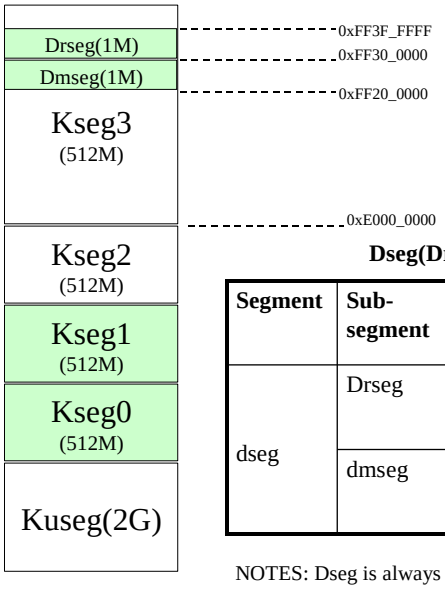
7.1.4 Processor Access Data Register in ACC mode (DATA_A)

It is connected between TDI and TDO by instruction DATA or ALL in ACC mode. This register is similar to PAD in MIPS mode, except it has one more RDY bit.

32	31 0
RDY	PAD

Field	BITS	Description	R/W	Reset value
PAD	31:0	Processor Access data The register has the written value for a write access to the dmseg. And it is also used to register the data value for load access or fetch instruction from the dmseg.	R/W	Undefined
RDY	1	Pipeline lock label. 1- processor can proceed due to processor access to dmseg done 0- processor should be locked due to unfinished processor access to dmseg	W	Undefined

7.1.5 Address space in Debug mode (AM = 0)



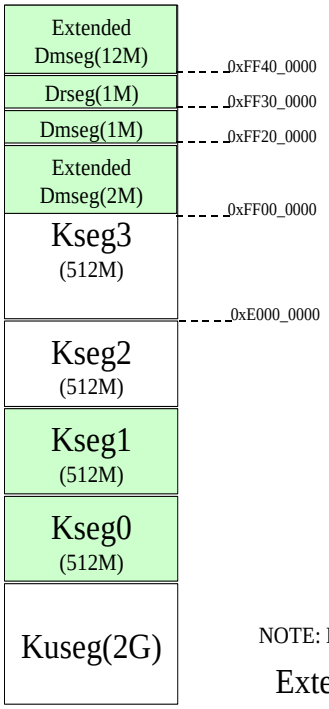
Dseg(Drseg + Dmseg) Address Space and Cache Attributes

Segment	Sub-segment	Virtual address	Physical address	Cache attribute
dseg	Drseg	0xFF30_0000 – 0xFF3F_FFFF	0xFF30_0000-0xFF3F_FFFF	2
	dmseg	0xFF20_0000 – 0xFF2F_FFFF	0xFF20_0000-0xFF2F_FFFF	2

NOTES: Dseg is always unmapped in debug mode in spite of attribute of Kseg3 .

Dseg space in MIPS mode

7.1.6 Address space in Debug mode (AM = 1)



Dseg (Drseg + Dmseg) Address Space and Cache Attributes

Segment	Sub segment	Virtual address	Physical address	Cache attribute
Dseg	extended Dmseg (2M)	0xFF00_0000-0xFF1F_FFFF	0xFF00_0000-0xFF1F_FFFF	0
	Dmseg (1M)	0xFF20_0000-0xFF2F_FFFF	0xFF20_0000-0xFF2F_FFFF	2
	Drseg (1M)	0xFF30_0000-0xFF3F_FFFF	0xFF30_0000-0xFF3F_FFFF	2
	extended Dmseg (12M)	0xFF40_0000-0xFFFF_FFFF	0xFF40_0000-0xFFFF_FFFF	0

NOTE: Dseg is always unmapped in debug mode in spite of attribute of Kseg3.

Extended Dseg space in ACC mode

7.1.7 Supported JTAG Instructions

Value	Instruction	Function
0x01	IDCODE	Select Chip Identification data register.
0x03	IMPCODE	Select implementation register.
0x08	ADDRESS	ADDRESS register is selected in MIPS mode while ADDRESS_A register is selected in ACC mode.
0x09	DATA	DATA register is selected in MIPS mode while DATA_A register is selected in ACC mode.
0x0A	CONTROL	ECR register is selected in MIPS mode while ECR_A register is selected in ACC mode.
0x0B	ALL	In MIPS mode, Selects the ADDRESS, DATA and ECR register. The scan sequence is TDI-> ADDRESS-> DATA->ECR->TDO. In ACC mode, Selects the DATA_A, ADDRESS_A, and ECR_A register. The scan sequence is TDI-> DATA_A ->ADDRESS_A ->ECR_A->TDO.
0x0C	EJTAGBOOT	Boot from probe host in MIPS mode by setting ECREjtagbrk, ECRProbEn and ECRProbTrap when reset Bypass register is selected.
0x0D	NORMALBOOT	Boot in normal way by clearing Ejtagbrk, ProbEn and ProbTrap when reset Bypass register is selected.
0x1C	EJTAGBOOTA	Boot from probe host in ACC mode by setting ECREjtagbrk, ECRProbEn, ECRProbTrap and AM when reset. Bypass register is selected.
0x1F	BYPASS	Select Bypass register.

7.1.8 Fetch/Load and Store From/to the JTAG Probe through dmseg in MIPS mode

Fetch/load from JTAG memory

1 The internal hardware latches the requested address into the Address register (ADDRESS).
2 The internal hardware sets the following bits in the JTAG Control register:
PrAcc = 1;
PRnW = 0;
Psz = Value depending on the transfer size.
3 The JTAG Probe selects the JTAG Control register, shifts out its content and tests the PrAcc bit: when the PrAcc bit is 1, it means that pending processor access need be serviced.
4 The JTAG Probe checks the PRnW bit to determine the required access.
5 The JTAG Probe selects the ADDRESS register and shifts out its content.
6 The JTAG Probe selects the DATA register and shifts in the required instruction/data.
7 The JTAG Probe selects the JTAG Control register again, and shifts a PrAcc = 0 into this register to indicate that the instruction/data is available and the processor can proceed.

Store to JTAG memory

1 The internal hardware latches the requested address into the Address register (ADDRESS).
2 The internal hardware sets the following bits in the JTAG Control register:
PrAcc = 1;
PRnW = 1;
Psz = Value depending on the transfer size.
3 The JTAG Probe selects the JTAG Control register, shifts out its content and tests the PrAcc bit: when the PrAcc bit is found 1, it means that pending processor access need be serviced.
4 The JTAG Probe checks the PRnW bit to determine the required access.
5 The JTAG Probe selects the ADDRESS register and shifts out its content.
6 The JTAG Probe selects the DATA register and shifts out its content to location determined by ADDRESS.
7 The JTAG Probe selects the JTAG Control register again and shifts a PrAcc = 0 into this register to indicate that the write has been done and the processor can proceed.

8 SOC Specific Notes

XBurst®1 Core is implemented in Ingenic's JZxxxx SOC chips. However, each chip may implement some variable features. For example, later chips with enhanced fabric process may implement larger cache size, more powerful Floating Point Units, huge size L2-cache, etc. This section describes these SOC specific features.

Features	JZ4750	JZ4760	JZ4770 JZ4775 X1000	JZ4780	M200	T-series
MIPS32-R1 ISA	Yes	Yes	Yes	Yes	Yes	Yes
MIPS32-R2 Integer Instructions	No	Yes	Yes	Yes	Yes	Yes
MIPS32-R2 Floating point ISA	No	Yes	Yes	Yes	Yes	Yes
Ingenic MXU1	Yes	Yes	Yes	Yes	Yes	No
Ingenic MXU2	No	No	No	No	No	Yes
L1 I-cache	16kB	16kB	16kB	32kB	32kB	32kB
L1 D-cache	16kB	16kB	16kB	32kB	32kB	32kB
L2 cache (unified cache)	No	No	256kB[1)]	512kB[1)]	512kB[2)]	ref-soc
Ingenic PMON	No	Yes	Yes	Yes	Yes	Yes
CoreSceduler (for MP-cores)	No	No	No	Yes	Yes	No
SMP support	No	No	No	Yes	No	No
Big-Little cores support	No	No	No	No	Yes	No
CP0.ErrCtl.WST	No	No	Ye	Yes	Yes	No

Notes:

1) 128-byte cache line, 4-way set association, WT only
2) 32-byte cache line, 8-way set association, WT & WB

Revision History

Revision	Date	Description
02.03	February 15,2017	1:update 2.3.2.4 EntryHi register EHINV and ASIDX field reset value as zero and R/W attribute as R 2:update 2.3.1.10 STATUS register BIT[9:8] field reset value as zero and R/W attribute as R/W 3:update 2.3.7.1 perfctl0 register EVENT field reset value as zero and R/W attribute as R/W 4:update 2.3.7.3 perfctl1 register EVENT field reset value as zero and R/W attribute as R/W