NEC

Preliminary User's Manual

μPD98502

Network Controller

Document No. S15543EJ1V0UM00 (1st edition)
Date Published December 2001 NS CP(K)

[MEMO]

SUMMARY OF CONTENTS

NOTES FOR CMOS DEVICES

① PRECAUTION AGAINST ESD FOR SEMICONDUCTORS

Note:

Strong electric field, when exposed to a MOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop generation of static electricity as much as possible, and quickly dissipate it once, when it has occurred. Environmental control must be adequate. When it is dry, humidifier should be used. It is recommended to avoid using insulators that easily build static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work bench and floor should be grounded. The operator should be grounded using wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions need to be taken for PW boards with semiconductor devices on it.

② HANDLING OF UNUSED INPUT PINS FOR CMOS

Note:

No connection for CMOS device inputs can be cause of malfunction. If no connection is provided to the input pins, it is possible that an internal input level may be generated due to noise, etc., hence causing malfunction. CMOS devices behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using a pull-up or pull-down circuitry. Each unused pin should be connected to V_{DD} or GND with a resistor, if it is considered to have a possibility of being an output pin. All handling related to the unused pins must be judged device by device and related specifications governing the devices.

③ STATUS BEFORE INITIALIZATION OF MOS DEVICES

Note:

Power-on does not necessarily define initial status of MOS device. Production process of MOS does not define the initial operation status of the device. Immediately after the power source is turned ON, the devices with reset function have not yet been initialized. Hence, power-on does not guarantee out-pin levels, I/O settings or contents of registers. Device is not initialized until the reset signal is received. Reset operation must be executed immediately after power-on for devices having reset function.

VR4100, VR4102, VR4111, VR4120A, VR4300, VR4305, VR4310, VR4400, VR5000, VR10000, VR Series, VR4000 Series, VR4100 Series, and EEPROM are trademarks of NEC Corporation.
Micro Wire is a trademark of National Semiconductor Corp.
iAPX is a trademark of Intel Corp.
DEC VAX is a trademark of Digital Equipment Corp.
UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company, Ltd.
Ethernet is a trademark of Xerox Corp.
MIPS is a trademark of MIPS Technologies, Inc.

- **The information contained in this document is being issued in advance of the production cycle for the device. The parameters for the device may change before final production or NEC Corporation, at its own discretion, may withdraw the device prior to its production.**
- **Not all devices/types available in every country. Please check with local NEC representative for availability and additional information.**
- No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Corporation. NEC Corporation assumes no responsibility for any errors which may appear in this document.
- NEC Corporation does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from use of a device described herein or any other liability arising from use of such device. No license, either express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Corporation or others.
- Descriptions of circuits, software, and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software, and information in the design of the customer's equipment shall be done under the full responsibility of the customer. NEC Corporation assumes no responsibility for any losses incurred by the customer or third parties arising from the use of these circuits, software, and information.
- While NEC Corporation has been making continuous effort to enhance the reliability of its semiconductor devices, the possibility of defects cannot be eliminated entirely. To minimize risks of damage or injury to persons or property arising from a defect in an NEC semiconductor device, customers must incorporate sufficient safety measures in its design, such as redundancy, fire-containment, and anti-failure features.
- NEC devices are classified into the following three quality grades:
  "Standard", "Special", and "Specific". The Specific quality grade applies only to devices developed based on a customer designated "quality assurance program" for a specific application. The recommended applications of a device depend on its quality grade, as indicated below. Customers must check the quality grade of each device before using it in a particular application.

  Standard: Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots

  Special: Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment and medical equipment (not specifically designed for life support)

  Specific: Aircraft, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems or medical equipment for life support, etc.

  The quality grade of NEC devices is "Standard" unless otherwise specified in NEC's Data Sheets or Data Books. If customers intend to use NEC devices for applications other than those specified for Standard quality grade, they should contact an NEC sales representative in advance.

M5D 98. 12

PREFACE

Readers

This manual is intended for engineers who need to be familiar with the capability of the μPD98502 in order to develop application systems based on it.

Purpose

The purpose of this manual is to help users understand the hardware capabilities (listed below) of the μPD98502.

Configuration

This manual consists of the following chapters:

- Introduction
- VR4120A™ CPU
- System controller
- ATM cell processor
- Ethernet™ controller
- USB controller
- PCI controller
- UART
- Timer
- Micro Wire™

Guidance

Readers of this manual should already have a general knowledge of electronics, logic circuits, and microcomputers.
To gain an overall understanding of the function of the μPD98502:
→ Read through all the chapters, in sequence.

To check the electrical characteristics of the μPD98502:
→ Refer to the separate data sheet.

Notation

This manual uses the following conventions:

Data bit significance:	High-order bits on the left side; low-order bits on the right side
Active low:	XXXX_B (Pin and signal names are suffixed with _B.)
Note:	Explanation of an indicated part of text
Caution:	Information requiring the user's special attention
Remark:	Supplementary information
Numerical value:	Binary ... xxxx or xxxxB Decimal ... xxxx Hexadecimal ... xxxxH

Related Document

Use this manual in combination with the following document.
The related documents indicated in this publication may include preliminary versions. However, preliminary versions are not marked as such.

- μPD98502 Data Sheet: S15409E

CONTENTS

LIST OF FIGURES (1/5)

Figure No. Title Page

LIST OF FIGURES (2/5)

LIST OF FIGURES (3/5)

LIST OF FIGURES (4/5)

LIST OF FIGURES (5/5)

LIST OF TABLES (1/2)

Table No. Title Page

LIST OF TABLES (2/2)

CHAPTER 1 INTRODUCTION

The μPD98502 is a high performance controller, which can perform the protocol conversion between IP Packets and ATM Cells, which is especially suitable for ADSL router. It includes high performance MIPS™ based 64-bit RISC processor VR4120A CPU core, ATM Cell Processor, Ethernet Controller, USB Controller Block, PCI Controller Block, UTOPIA2 interface and SDRAM interface.

1.1 Features

- Includes high performance MIPS based 64-bit RISC processor VR4120A
- Can perform RTOS and network middleware (M/W) on the chip
- Includes interface for PROM and flash ROM used for storing boot program
- Includes 32-bit RISC controller in ATM Cell Processor
- Software SAR processing by RISC controller affords flexibility for specification update
- Supports CBR/VBR/UBR service classes
- Includes 2-channel 10/100-Mbps Ethernet controller compliant to IEEE802.3, IEEE 802.3u and IEEE802.3x
- Can directly connect external Ethernet PHY device through 3.3 V MII interface
- Includes USB full speed function controller compliant to USB specification 1.1
- Supports operation conforming to the USB Communication Device Class Specification
- Can directly connect 64-Mbit and 128-Mbit SDRAM as external memory
- Includes 32-bit 33-MHz PCI Bus Master compliant to PCI Specification Rev. 2.2
- Includes 8-bit 16.5/25/33-MHz UTOPIA level 2 interface compliant to ATM Forum af-phy-0039
- Includes boundary scan function (JTAG) compliant to IEEE 1149.1
- Includes Micro Wire interface
- Includes 2-ch general purpose timers
- Using advanced CMOS technology
- Power supply 2.5V(Core)/3.3V(I/O)
- Package 500-pin T-BGA

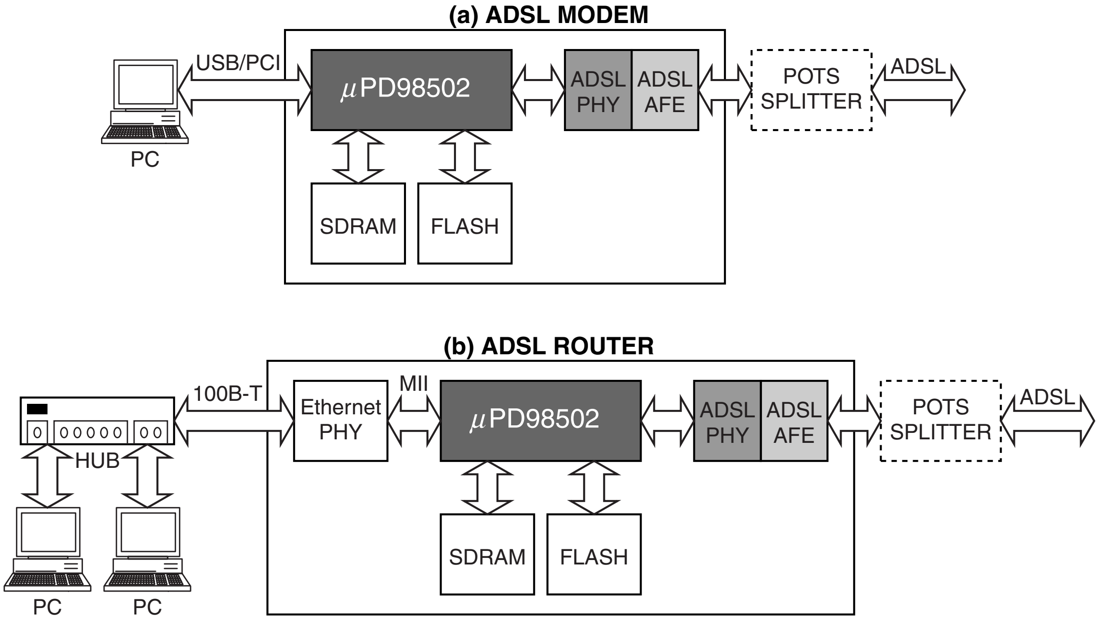
1.2 Ordering Information

Part Number	Package
μPD98502N7-H6	500-pin Tape BGA (Heat spread type) (40 × 40)

1.3 System Configuration

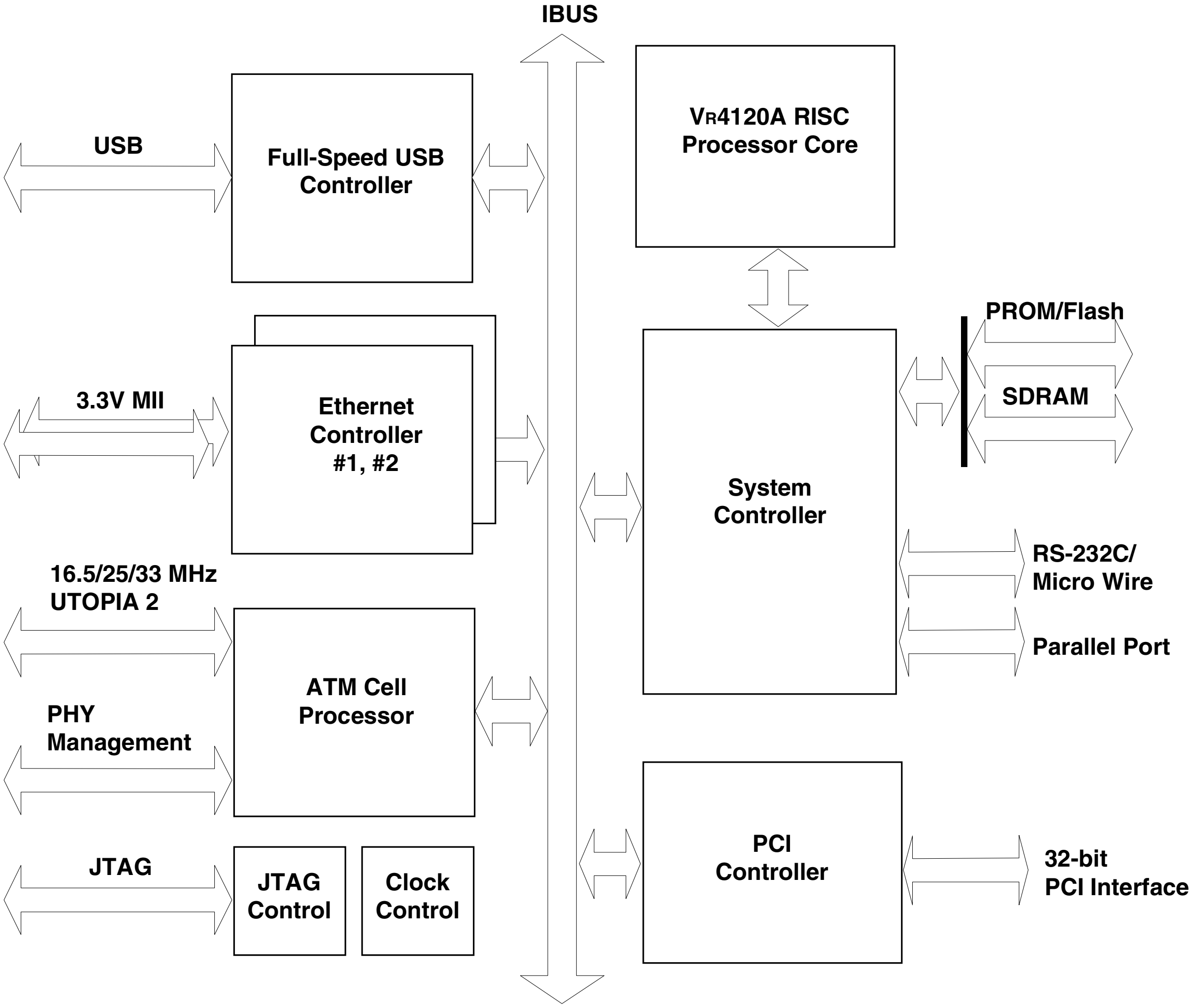
The *μ*PD98502 can perform bridging and routing function between ADSL/ATM interface and USB/Ethernet interface and provides this function in a single chip. By selecting user interface, examples of system configuration will be realized as shown below. USB and Ethernet functions will exclusively operate each other.

Figure 1-1. Examples of the *μ*PD98502 System Configuration



1.4 Block Diagram (Summary)

Figure 1-2. Block Diagram of the μPD98502



1.5 Block Diagram (Detail)

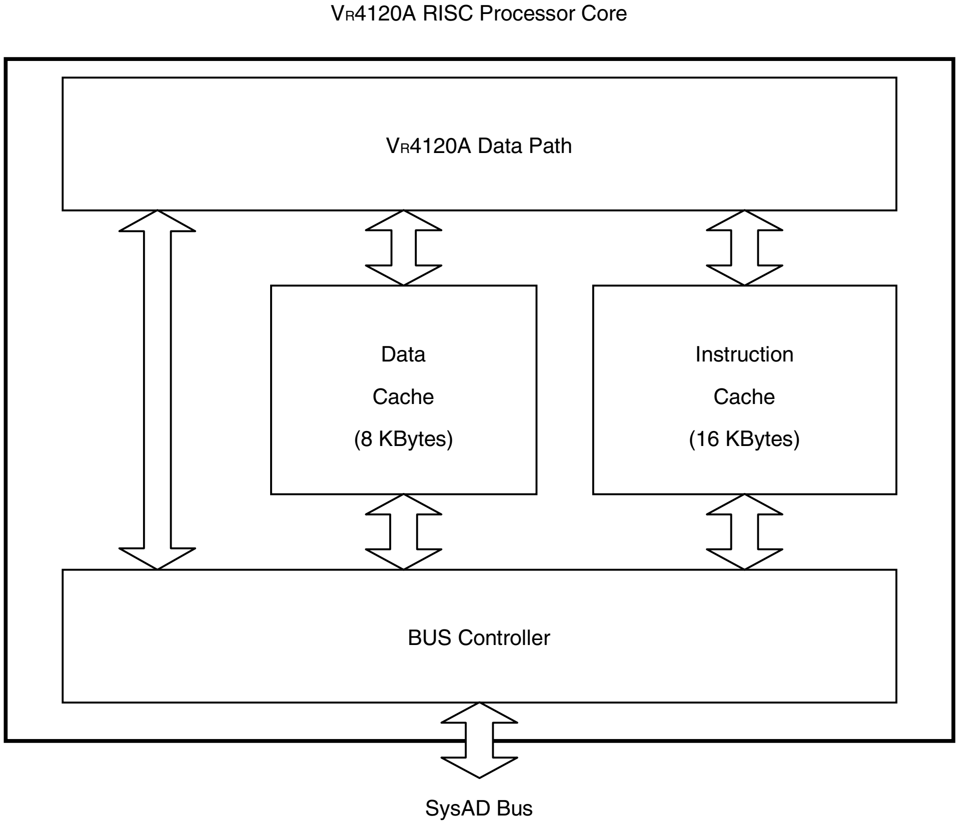
1.5.1 VR4120A RISC processor core

We will support real-time OS running on high performance RISC processor VR4120A core and can perform network protocols (TCP/IP, PPP, SNMP, HTTP etc) to realize ADSL router and modem. Middleware including RTOS will be loaded to SDRAM from external PROM and Flash ROM and by setting write protected area for such an area, high speed processing will be realized together with large size instruction cache.

Features of VR4120A RISC Processor Core are as follows;

- MIPS/I/II/III instruction set will be supported (FPU, LL, LLD, SC, SCD instruction will be excluded)
- Realize high speed processing of application by supporting high speed multiply and accumulate function
- Includes large size cache memory (Instruction: 16 Kbytes, Data: 8 Kbytes)
- Supports up to 1T byte virtual address space by using full associative TLB
- Implements switching function between Big-Endian and Little-Endian

Figure 1-3. Block Diagram of VR4120A RISC Processor



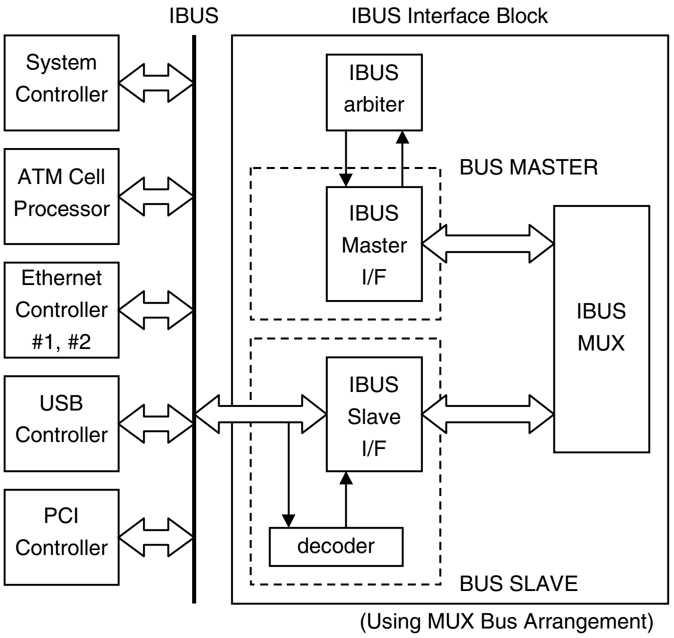
1.5.2 IBUS

The IBUS is a 32-bit, 66-MHz high-speed on-chip bus, which enables interconnection each controller blocks.

The IBUS supports the following bus protocols;

- Single read/write transfer
- Burst read/write transfer
- Slave lock
- Retry and disconnect
- Bus parking

Figure 1-4. Block Diagram of IBUS



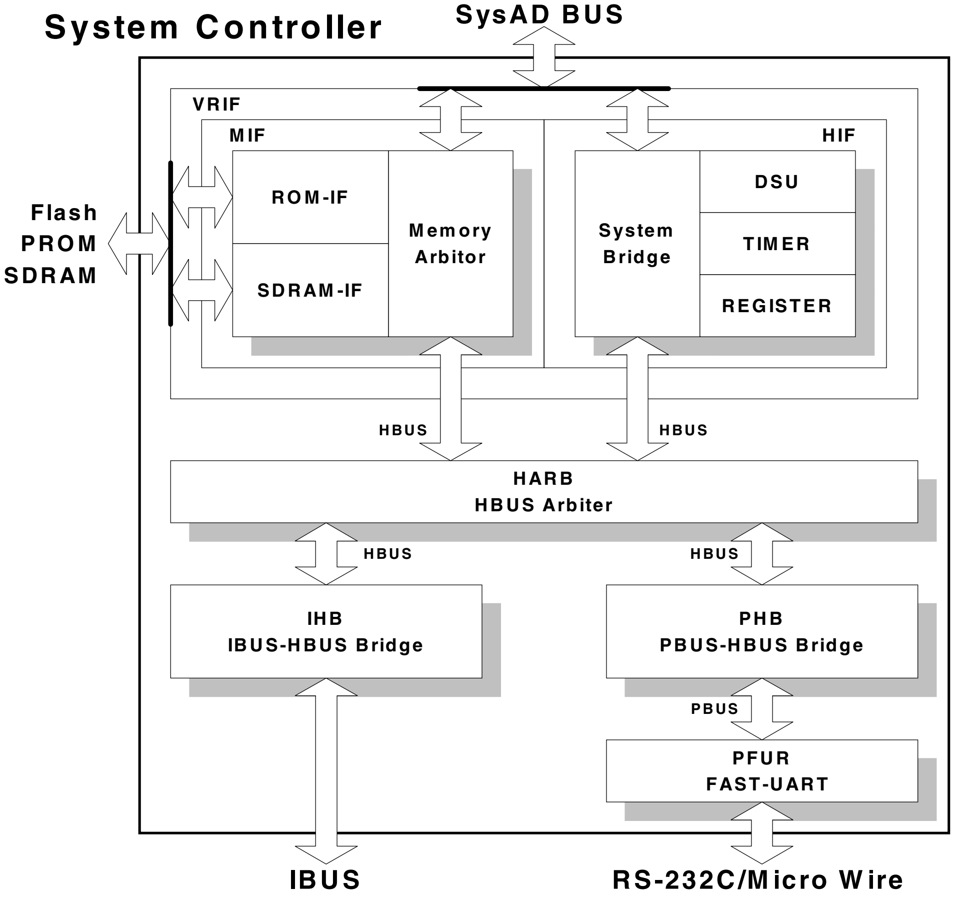
1.5.3 System controller

System Controller is μPD98502's internal system controller. System Controller provides bridging function among the VR4120A System Bus "SysAD", NEC original high-speed on-chip bus "IBUS" and memory bus for SDRAM/PROM/Flash.

Features of System Controller are as follows;

- Implements 4-word prefetch FIFO buffer between SysAD and Memory
- Implements 32-bit×64-word FIFO buffer for each Tx and Rx to IBUS
- Implements 32-bit× 4-word FIFO buffer for each Tx and Rx to HBUS
- Provides bus bridging function among SysAD bus and IBUS (internal bus) and Memory
- Supports Endian Converting function on SysAD bus
- Can directly connect SDRAM (MAX. 32 MBytes) and PROM/Flash (MAX. 8 MBytes) memory
- Supports all VR4120A bus cycles at 66 MHz or 100 MHz
- PROM/Flash data signals multiplexed on SDRAM data signals
- Supports 266-MB/sec (32 bits @66 MHz) bursts on IBUS
- Generates NMI and INT
- Supports NS16550 compatible Universal Asynchronous Receiver/Transmitter (UART)
- Supports separated 2-ch Timer
- Supports Deadman's Switch Unit (Watch Dog Timer)
- Supports Micro Wire interface

Figure 1-5. Block Diagram of System Controller



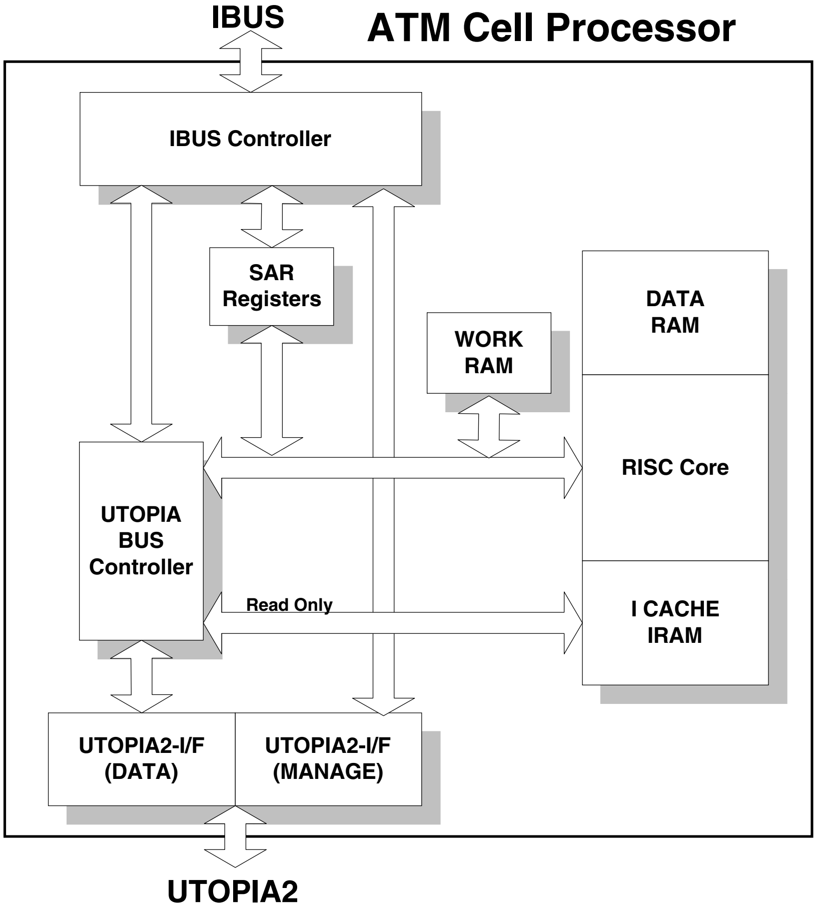
1.5.4 ATM cell processor

By using NEC proprietary 32-bit controller, we will realize ATM Cell processor Unit. ATM Cell processing by firmware realizes more flexibility than before.

Features of ATM Cell Processor are as follows;

- Realize software SAR function by using 32-bit RISC controller (76 MIPS @66 MHz)
- Firmware is downloaded from external memory to Instruction Cache
- Supports 64 VCs
- Supports UTOPIA level 2 (including management interface) as PHY layer interface
- Supports processing AAL2, AAL5, Raw cell (AAL0) and F5 OAM cells
- Supports 3 service classes (CBR, VBR, UBR)
- Supports up to 50 Mbps Cell speed together with upstream and downstream
- Supports fine grain ATM cell shaping in 1cell/sec granularity on per VC basis

Figure 1-6. Block Diagram of ATM Cell Processor



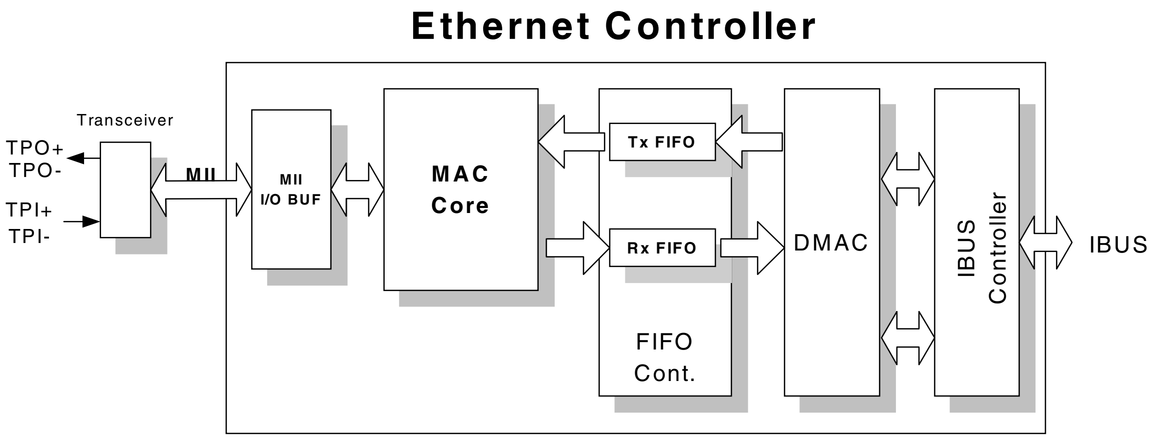
1.5.5 Ethernet controller

Ethernet Controller supports 2-channel 10 Mbps/100 Mbps Ethernet MAC (Media Access Control) function and MII (Media Independent Interface) function.

Features of Ethernet Controller are as follows;

- Supports 10 M/100 M Ethernet MAC function compliant to IEEE802.3 and IEEE802.3u
- Supports 3.3 V MII compliant to IEEE802.3u
- Supports full duplex operation for both 100 Mbps and 10 Mbps
- Supports flow control function compliant to IEEE802.3x/D3.2
- Implements 256-Byte FIFO buffer for each Tx and Rx
- Implements address filtering functions for unicast/multicast/broadcast
- Implements MIB counters for network management (MIB II, Ether-like MIB, IEEE802.3LME are supported)
- Implements local DMA controller with individual DMA channels for each Tx and Rx

Figure 1-7. Block Diagram of Ethernet Controller

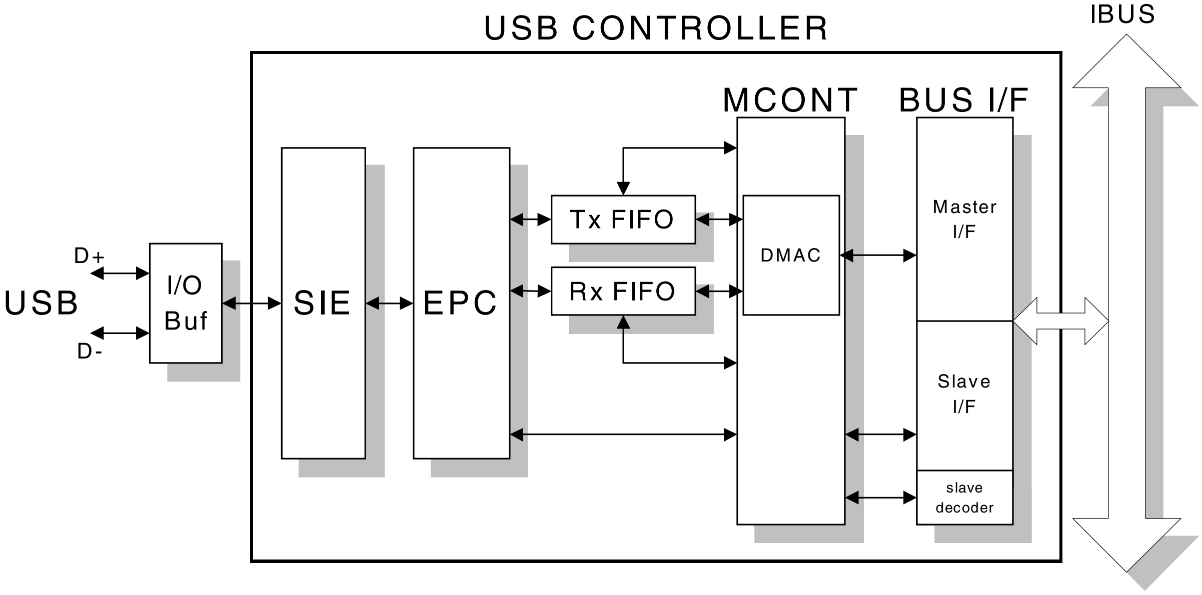


1.5.6 USB controller

USB Controller provides Full Speed Function device function defined in Universal Serial Bus.
Features of USB Controller are as follows;

- Compliant to Universal Serial Bus Specification Rev. 1.1
- Supports Device class function by software running on VR4120A
- Performs 12 Mbps Full Speed USB function device (Hub function will be not supported)
- Can handle Suspend, Resume and Wake-up management signaling
- Supports Remote Wake-up.
- Implements 7 kinds of endpoints (Control, Interrupt IN/OUT, Isochronous IN/OUT, Bulk IN/OUT)
- Implements 64 Bytes FIFO buffer used for Control transfer for Tx
- Implements 128 Bytes FIFO buffer used for Isochronous transfer for Tx
- Implements 128 Bytes FIFO buffer used for Bulk transfer for Tx
- Implements 64 Bytes FIFO buffer used for Interrupt transfer for Tx
- Implements 128 Bytes shared FIFO buffer used for Control/Isochronous/Bulk/Interrupt transfer for Rx
- Implements local DMAC (DMA controller) block
- Can directly connect USB connector through USB dedicated I/O buffer

Figure 1-8. Block Diagram of USB Controller

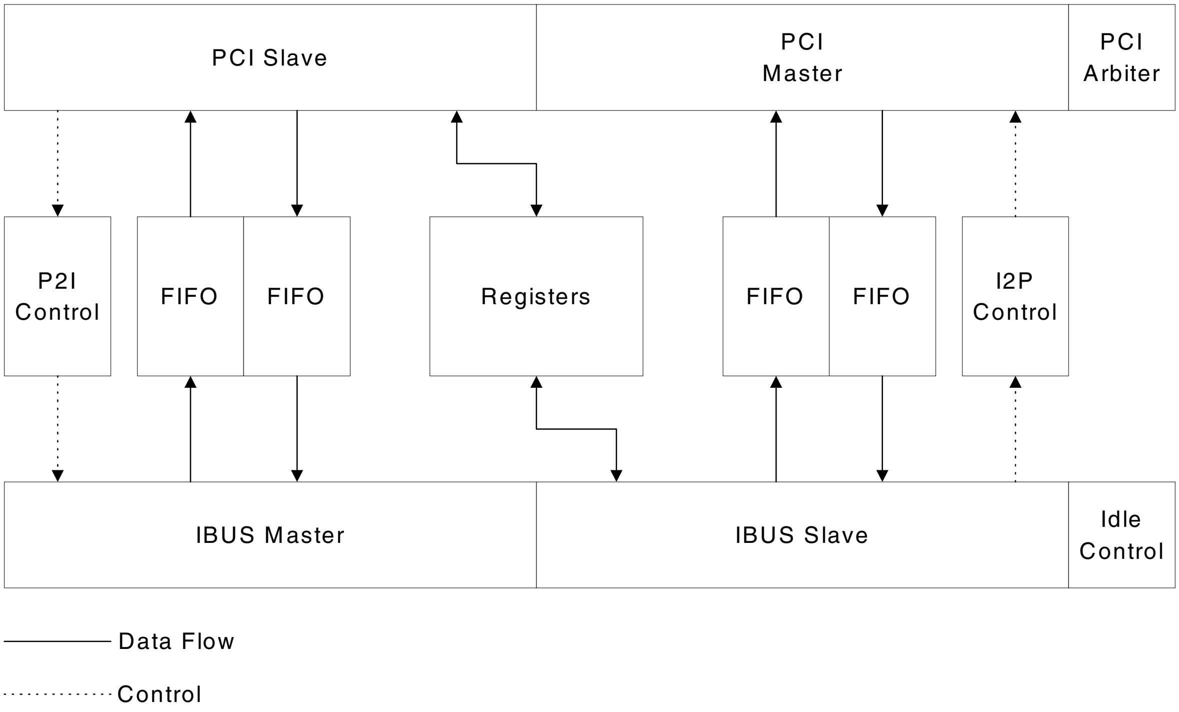


1.5.7 PCI controller

PCI Controller provides PCI Bus function defined by PCI SIG. This block is bridging between IBUS and PCI. Features of PCI Controller are as follows;

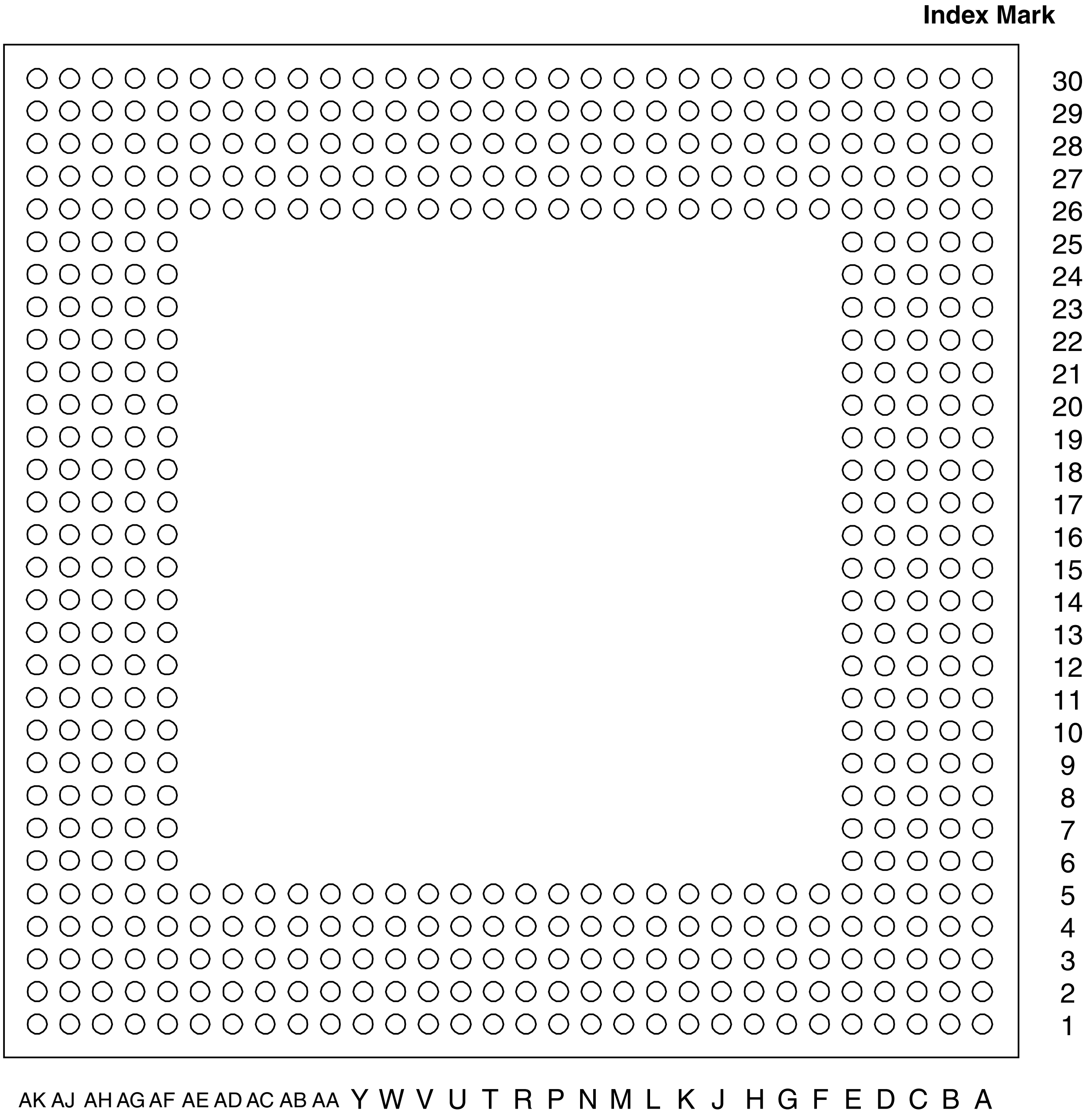
- 32-bit PCI Interface (up to 33 MHz)
- 32-bit IBUS Interface (up to 33 MHz)
- Supports PCI Dual Address Cycle as master
- 33-MHz-PCI-frequency capable
- Compliant to PCI Local Bus Specification Rev. 2.2
- Compliant to PCI Bus Power Management Interface Rev. 1.1
- Supports up to 16 words burst for each directions
- Implements PCI bus arbiter that supports up to 4 external PCI-master devices at Host-mode

Figure 1-9. Block Diagram of PCI Bus controller



1.6 Pin Configuration (Bottom View)

- 500-pin Tape BGA (Heat spread type) (40 × 40)
  μPD98502N7-H6


Index Mark
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
AK AJ AH AG AF AE AD AC AB AA Y W V U T R P N M L K J H G F E D C B A

Pin Name

(1/3)

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
A1	SMA13	B10	URSDO	C19	NJTRST	D28	PGTO2_B	F27	GND
A2	SMD0	B11	RMSL1	C20	IC-OPEN	D29	PRQI1_B	F28	GND
A3	SMD4	B12	MWDO	C21	JDI	D30	PAD0	F29	PAD5
A4	SMD7	B13	POM3	C22	GND	E1	GND	F30	PAD6
A5	SMD19	B14	POM5	C23	USBDM	E2	SDRAS_B	G1	SMA8
A6	SMD22	B15	EVDD	C24	IC-OPEN	E3	SMA0	G2	SMA15
A7	SRMCS_B	B16	IC-OPEN	C25	PUDGND	E4	SMA10	G3	SDCLK1
A8	URDSR_B	B17	IC-OPEN	C26	IVDD	E5	GND	G4	EVDD
A9	URDCD_B	B18	IC-OPEN	C27	PMODE	E6	EVDD	G5	SDCAS_B
A10	URDTR_B	B19	IC-OPEN	C28	PGTO3_B	E7	SMD16	G26	PAD1
A11	MWSK	B20	GND	C29	PGTO1_B	E8	GND	G27	PAD3
A12	MWDI	B21	IC-PDn	C30	PRQI0_B	E9	EVDD	G28	EVDD
A13	EXNMI	B22	JDO	D1	SDWE_B	E10	GND	G29	PAD7
A14	POM6	B23	GND	D2	SMA1	E11	RMSL0	G30	PAD8
A15	EXINT	B24	USBDP	D3	SMA11	E12	GND	H1	SMA4
A16	IVDD	B25	PUDVD	D4	IVDD	E13	POM0	H2	SMA7
A17	IC-OPEN	B26	IC-OPEN	D5	SMD3	E14	GND	H3	SMA9
A18	IC-PDnR	B27	PUMD	D6	SMD6	E15	POM7	H4	IVDD
A19	IC-OPEN	B28	PHINT_B	D7	EVDD	E16	GND	H5	GND
A20	IC-OPEN	B29	PRSTO_B	D8	IVDD	E17	GND	H26	GND
A21	IC-PDn	B30	PGTO0_B	D9	URCLK	E18	IC-OPEN	H27	IVDD
A22	JCK	C1	SMA2	D10	IVDD	E19	GND	H28	PCBE0_B
A23	JMS	C2	GND	D11	GND	E20	IC-PDn	H29	PAD9
A24	EVDD	C3	SMA16	D12	IVDD	E21	GND	H30	GND
A25	EVDD	C4	SMD2	D13	POM1	E22	EVDD	J1	SMA18
A26	PUAVD	C5	GND	D14	IVDD	E23	GND	J2	SMA3
A27	GND	C6	SMD18	D15	IC-PDnR	E24	USBCLK	J3	SMA5
A28	IC-OPEN	C7	SMD21	D16	BIG	E25	EVDD	J4	SMA6
A29	GND	C8	SRMOE_B	D17	IVDD	E26	GND	J5	EVDD
A30	PSERI_B	C9	GND	D18	IC-OPEN	E27	PRQI3_B	J26	EVDD
B1	SMA12	C10	URRTS_B	D19	IVDD	E28	PRQI2_B	J27	PAD10
B2	SMA14	C11	EVDD	D20	IC-OPEN	E29	PAD2	J28	PAD11
B3	SMD1	C12	MWCS	D21	IVDD	E30	PAD4	J29	PAD12
B4	SMD5	C13	POM2	D22	JRSTB_B	F1	SMA17	J30	PAD13
B5	SMD17	C14	POM4	D23	IVDD	F2	SDCKE1	K1	SMD31
B6	SMD20	C15	ENDCEN	D24	PUAGND	F3	SDCS_B	K2	SMA20
B7	SMD23	C16	GND	D25	PUSTBY	F4	GND	K3	SMA19
B8	URCTS_B	C17	NJTMS	D26	PARBN	F5	EVDD	K4	IVDD
B9	URSDI	C18	NJTDO	D27	IVDD	F26	EVDD	K5	GND

(2/3)

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
K26	GND	P5	GND	V4	IVDD	AB3	IC-PUp	AF2	MICRS
K27	IVDD	P26	GND	V5	GND	AB4	IC-PDn	AF3	MIMCLK
K28	PAD14	P27	IVDD	V26	PAD26	AB5	EVDD	AF4	MITD3
K29	PAD15	P28	PAD16	V27	PAD25	AB26	EVDD	AF5	GND
K30	EVDD	P29	PAD17	V28	PAD24	AB27	IC-OPEN	AF6	EVDD
L1	SDCLK0	P30	PAD18	V29	GND	AB28	IC-OPEN	AF7	MI2TE
L2	GND	R1	EVDD	V30	PSCLK	AB29	GND	AF8	GND
L3	SDCKE0	R2	SMD12	W1	IC-PUp	AB30	RSTB	AF9	EVDD
L4	SMD30	R3	SMD9	W2	IVDD	AC1	IC-PUp	AF10	GND
L5	EVDD	R4	SMD10	W3	IC-PUp	AC2	MIRD3	AF11	UDRD1
L26	PCBE1_B	R5	SMD11	W4	IVDD	AC3	GND	AF12	GND
L27	GND	R26	EVDD	W5	GND	AC4	IVDD	AF13	UDRAD2
L28	PAR	R27	PAD19	W26	GND	AC5	GND	AF14	GND
L29	PSERO_B	R28	PAD20	W27	IVDD	AC26	GND	AF15	UDTAD3
L30	PER_B	R29	GND	W28	PAD28	AC27	IVDD	AF16	UDTD7
M1	EVDD	R30	PAD21	W29	EVDD	AC28	IC-OPEN	AF17	GND
M2	SMD28	T1	SMD8	W30	PAD27	AC29	IC-OPEN	AF18	EVDD
M3	SMD29	T2	GND	Y1	IC-OPEN	AC30	PINT_B	AF19	GND
M4	IVDD	T3	CLKUSL1	Y2	PSDGND	AD1	MIRD2	AF20	UMWR_B
M5	GND	T4	CLKUSL0	Y3	PSAGND	AD2	MIRD1	AF21	GND
M26	GND	T5	EVDD	Y4	PSAVD	AD3	MIRCLK	AF22	EVDD
M27	IVDD	T26	PAD22	Y5	PSDVD	AD4	MIRER	AF23	GND
M28	PSTP_B	T27	IVDD	Y26	PAD31	AD5	MIRDV	AF24	GND
M29	PDSEL_B	T28	GND	Y27	PME_B	AD26	GND	AF25	EVDD
M30	EVDD	T29	PAD23	Y28	PRQO_B	AD27	EVDD	AF26	GND
N1	SMD24	T30	PCBE3_B	Y29	PAD30	AD28	IC-PDnR	AF27	GND
N2	SMD25	U1	CLKSL	Y30	PAD29	AD29	IC-OPEN	AF28	IC-PDnR
N3	GND	U2	GND	AA1	IC-OPEN	AD30	GND	AF29	IC-PDnR
N4	SMD26	U3	IC-PUp	AA2	PSTBY	AE1	MIRD0	AF30	IC-PDnR
N5	SMD27	U4	IVDD	AA3	PSMD	AE2	GND	AG1	MIMD
N26	PTRY_B	U5	GND	AA4	IVDD	AE3	MITER	AG2	GND
N27	PIRY_B	U26	GND	AA5	GND	AE4	IVDD	AG3	MITD2
N28	GND	U27	IVDD	AA26	GND	AE5	EVDD	AG4	IVDD
N29	PFRA_B	U28	PIDSEL	AA27	IVDD	AE26	EVDD	AG5	MI2COL
N30	PCBE2_B	U29	GND	AA28	PGTI_B	AE27	IC-PDnR	AG6	IVDD
P1	SMD13	U30	EVDD	AA29	GND	AE28	IC-PDnR	AG7	MI2CRS
P2	SMD14	V1	SCLK	AA30	EVDD	AE29	IC-PDnR	AG8	IVDD
P3	SMD15	V2	GND	AB1	IC-OPEN	AE30	IC-PDnR	AG9	UDRSC
P4	IVDD	V3	IC-PUp	AB2	GND	AF1	MITE	AG10	IVDD

(3/3)

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
AG11	UDRD0	AH3	MICOL	AH25	EVDD	AJ17	UDTD5	AK9	UDRD5
AG12	IVDD	AH4	MI2RD0	AH26	IVDD	AJ18	GND	AK10	UDRD2
AG13	UDRAD1	AH5	MI2MD	AH27	UMAD8	AJ19	UMRDY_B	AK11	UDRCLK
AG14	IVDD	AH6	MI2TER	AH28	UMAD7	AJ20	GND	AK12	UDRAD3
AG15	UDTAD2	AH7	MI2TD3	AH29	UMAD3	AJ21	EVDD	AK13	UDRAD0
AG16	UDTAD0	AH8	GND	AH30	UMAD1	AJ22	UMD13	AK14	UDTE_B
AG17	IVDD	AH9	UDRCLV	AJ1	MITD0	AJ23	UMD9	AK15	UDTAD4
AG18	UDTD1	AH10	UDRD4	AJ2	MI2MCLK	AJ24	UMD7	AK16	UDTCLK
AG19	IVDD	AH11	UDTCLV	AJ3	MI2RD1	AJ25	UMD4	AK17	UDTD6
AG20	UMMD	AH12	IC-OPEN	AJ4	GND	AJ26	UMD1	AK18	UDTD3
AG21	IVDD	AH13	EVDD	AJ5	MI2RER	AJ27	GND	AK19	UDTD0
AG22	UMD10	AH14	UDTSC	AJ6	GND	AJ28	GND	AK20	UMRST_B
AG23	IVDD	AH15	EVDD	AJ7	MI2TD1	AJ29	UMAD6	AK21	UMRD_B
AG24	UMD2	AH16	UDTAD1	AJ8	UDRE_B	AJ30	UMAD4	AK22	UMD14
AG25	UMAD11	AH17	UDTD4	AJ9	UDRD6	AK1	MI2RD3	AK23	UMD12
AG26	UMAD9	AH18	UDTD2	AJ10	UDRD3	AK2	MI2RD2	AK24	UMD8
AG27	IVDD	AH19	UMINT_B	AJ11	UDRAD4	AK3	MI2RCLK	AK25	UMD6
AG28	UMAD2	AH20	UMSL_B	AJ12	IC-OPEN	AK4	MI2RDV	AK26	UMD3
AG29	UMAD0	AH21	UMD15	AJ13	GND	AK5	MI2TCLK	AK27	UMD0
AG30	IVDD	AH22	UMD11	AJ14	GND	AK6	MI2TD2	AK28	UMAD10
AH1	MITCLK	AH23	GND	AJ15	IVDD	AK7	MI2TD0	AK29	IC-PUp
AH2	MITD1	AH24	UMD5	AJ16	GND	AK8	UDRD7	AK30	UMAD5

Special pin name description:

IC-PDn:	Pull Down
IC-PDnR:	Pull Down with Resistor
IC-PUp:	Pull Up
IC-PUpR:	Pull Up with Resistor

Remark In this document, XXX_B stands for active low pin.

1.7 Pin Function

Symbol of I/O column indicates following status in this section.

I	: Input
O	: Output
I/O	: Bidirection
I/OZ	: Bidirection (Include Hi-Z state)
I/OD	: Bidirection (Open drain output)
OZ	: Output (Include Hi-Z state)
OD	: Output (Open drain)

1.7.1 Power supply

Pin Name	Pin No.	I/O	Active Level	Function
GND	A27, A29, B20, B23, C16, C2, C22, C5, C9, D11, E1, E10, E12, E14, E16, E17, E19, E21, E23, E26, E5, E8, F27, F28, F4, H26, H30, H5, K26, K5, L2, L27, M26, M5, N28, N3, P26, P5, R29, T2, T28, U2, U26, U29, U5, V2, V29, V5, W26, W5, AA26, AA29, AA5, AB2, AB29, AC26, AC3, AC5, AD26, AD30, AE2, AF10, AF12, AF14, AF17, AF19, AF21, AF23, AF24, AF26, AF27, AF5, AF8, AG2, AH23, AH8, AJ13, AJ14, AJ16, AJ18, AJ20, AJ27, AJ28, AJ4, AJ6			GND (0 V)
IVDD	A16, C26, D10, D12, D14, D17, D19, D21, D23, D27, D4, D8, H27, H4, K27, K4, M27, M4, P27, P4, T27, U27, U4, V4, W2, W27, W4, AA27, AA4, AC27, AC4, AE4, AG10, AG12, AG14, AG17, AG19, AG21, AG23, AG27, AG30, AG4, AG6, AG8, AH26, AJ15			Internal logic core power supply (+2.5 V)
EVDD	A24, A25, B15, C11, D7, E22, E25, E6, E9, F26, F5, G28, G4, J26, J5, K30, L5, M1, M30, R1, R26, T5, U30, W29, AA30, AB26, AB5, AD27, AE26, AE5, AF18, AF22, AF25, AF6, AF9, AH13, AH15, AH25, AJ21			External (I/O) power supply (+3.3 V)

1.7.2 System PLL power supply

Pin Name	Pin No.	I/O	Active Level	Function
PSAGND	Y3	I		Analog ground
PSAVD	Y4	I		Analog power supply
PSDGND	Y2	I		Digital ground
PSDVD	Y5	I		Digital power supply

1.7.3 USB PLL power supply

Pin Name	Pin No.	I/O	Active Level	Function
PUAGND	D24	I		Analog ground
PUAVD	A26	I		Analog power supply
PUDGND	C25	I		Digital ground
PUDVD	B25	I		Digital power supply

1.7.4 System control interface

Pin Name	Pin No.	I/O	Active Level	Function
SCLK	V1	I		System clock (33 MHz)
CLKSL	U1	I		Clock select (100 MHz/66 MHz)
PSMD	AA3	I		System PLL mode control (0: normal, 1: through)
PSTBY	AA2	I		System PLL standby mode control (0: active, 1: standby)
PUMD	B27	I		USB PLL mode control (0: normal, 1: through)
PUSTBY	D25	I		USB PLL standby mode control (0: active, 1: standby)
BIG	D16	I	H	VR4120 big endian mode
ENDCEN	C15	I		Endian conversion enable
EXINT_B	A15	I	L	External interrupt
EXNM_BI	A13	I	L	External non-maskable interrupt
RSTB_B	AB30	I	L	System reset
RMSL0	E11	I		ROM access bus width select (RMSL1/0 = L/L: 32-bit, L/H: 16-bit, H/L: 8-bit)
RMSL1	B11	I		

1.7.5 Memory interface

(1/2)

Pin Name	Pin No.	I/O	Active Level	Function
SDCLK0	L1	O		SDRAM clock
SDCLK1	G3	O		SDRAM clock
SDCKE0	L3	O	H	SDRAM clock enable
SDCKE1	F2	O	H	SDRAM clock enable
SDCS_B	F3	O	L	Chip select
SDRAS_B	E2	O	L	Row address strobe
SDCAS_B	G5	O	L	Column address strobe
SDWE_B	D1	O	L	Write enable
SRMCS_B	A7	O	L	PROM/FLASH chip select
SRMOE_B	C8	O	L	PROM/FLASH output enable
SMA0	E3	O		Memory address
SMA1	D2	O		Memory address
SMA2	C1	O		Memory address
SMA3	J2	O		Memory address
SMA4	H1	O		Memory address
SMA5	J3	O		Memory address
SMA6	J4	O		Memory address
SMA7	H2	O		Memory address
SMA8	G1	O		Memory address
SMA9	H3	O		Memory address
SMA10	E4	O		Memory address
SMA11	D3	O		Memory address
SMA12	B1	O		Memory address
SMA13	A1	O		Memory address
SMA14	B2	O		Memory address
SMA15	G2	O		Memory address
SMA16	C3	O		Memory address
SMA17	F1	O		Memory address
SMA18	J1	O		Memory address
SMA19	K3	O		Memory address
SMA20	K2	O		Memory address
SMD0	A2	I/O		Memory data
SMD1	B3	I/O		Memory data
SMD2	C4	I/O		Memory data
SMD3	D5	I/O		Memory data
SMD4	A3	I/O		Memory data
SMD5	B4	I/O		Memory data
SMD6	D6	I/O		Memory data
SMD7	A4	I/O		Memory data
SMD8	T1	I/O		Memory data
SMD9	R3	I/O		Memory data
SMD10	R4	I/O		Memory data

(2/2)

Pin Name	Pin No.	I/O	Active Level	Function
SMD11	R5	I/O		Memory data
SMD12	R2	I/O		Memory data
SMD13	P1	I/O		Memory data
SMD14	P2	I/O		Memory data
SMD15	P3	I/O		Memory data
SMD16	E7	I/O		Memory data
SMD17	B5	I/O		Memory data
SMD18	C6	I/O		Memory data
SMD19	A5	I/O		Memory data
SMD20	B6	I/O		Memory data
SMD21	C7	I/O		Memory data
SMD22	A6	I/O		Memory data
SMD23	B7	I/O		Memory data
SMD24	N1	I/O		Memory data
SMD25	N2	I/O		Memory data
SMD26	N4	I/O		Memory data
SMD27	N5	I/O		Memory data
SMD28	M2	I/O		Memory data
SMD29	M3	I/O		Memory data
SMD30	L4	I/O		Memory data
SMD31	K1	I/O		Memory data

1.7.6 PCI interface

(1/2)

Pin Name	Pin No.	I/O	Active Level	Function
PSCLK	V30	I		PCI clock
PARBN	D26	I		PCI arbiter enable
PMODE	C27	I		PCI mode select (L: host, H: NIC)
PIDSEL	U28	I	H	Initialization device select
PDSEL_B	M29	I/OZ	L	Device select
PER_B	L30	I/OZ	L	Parity error
PFRA_B	N29	I/OZ	L	Cycle frame
PHINT_B	B28	I	L	PCI host interrupt
PINT_B	AC30	O	L	Interrupt_A
PIRY_B	N27	I/OZ	L	Initiator ready
PME_B	Y27	OD	L	Power management event
PRSTO_B	B29	O	L	PCI system reset out
PSERI_B	A30	I	L	System error in
PSERO_B	L29	O	L	System error out
PTRY_B	N26	I/OZ	L	Target ready
PSTP_B	M28	I/OZ	L	Stop request from target
PCBE0_B	H28	I/OZ	L	Bus command and byte enable
PCBE1_B	L26	I/OZ	L	Bus command and byte enable
PCBE2_B	N30	I/OZ	L	Bus command and byte enable
PCBE3_B	T30	I/OZ	L	Bus command and byte enable
PRQO_B	Y28	O	L	Bus request out
PRQI0_B	C30	I	L	Bus request in[0]
PRQI1_B	D29	I	L	Bus request in[1]
PRQI2_B	E28	I	L	Bus request in[2]
PRQI3_B	E27	I	L	Bus request in[3]
PGTI_B	AA28	I	L	Bus grant in
PGTO0_B	B30	O	L	Bus grant out[0]
PGTO1_B	C29	O	L	Bus grant out[1]
PGTO2_B	D28	O	L	Bus grant out[2]
PGTO3_B	C28	O	L	Bus grant out[3]
PAR	L28	I/OZ		Parity of address/data
PAD0	D30	I/OZ		PCI address and data
PAD1	G26	I/OZ		PCI address and data
PAD2	E29	I/OZ		PCI address and data
PAD3	G27	I/OZ		PCI address and data
PAD4	E30	I/OZ		PCI address and data
PAD5	F29	I/OZ		PCI address and data
PAD6	F30	I/OZ		PCI address and data
PAD7	G29	I/OZ		PCI address and data
PAD8	G30	I/OZ		PCI address and data
PAD9	H29	I/OZ		PCI address and data
PAD10	J27	I/OZ		PCI address and data

(2/2)

Pin Name	Pin No.	I/O	Active Level	Function
PAD11	J28	I/OZ		PCI address and data
PAD12	J29	I/OZ		PCI address and data
PAD13	J30	I/OZ		PCI address and data
PAD14	K28	I/OZ		PCI address and data
PAD15	K29	I/OZ		PCI address and data
PAD16	P28	I/OZ		PCI address and data
PAD17	P29	I/OZ		PCI address and data
PAD18	P30	I/OZ		PCI address and data
PAD19	R27	I/OZ		PCI address and data
PAD20	R28	I/OZ		PCI address and data
PAD21	R30	I/OZ		PCI address and data
PAD22	T26	I/OZ		PCI address and data
PAD23	T29	I/OZ		PCI address and data
PAD24	V28	I/OZ		PCI address and data
PAD25	V27	I/OZ		PCI address and data
PAD26	V26	I/OZ		PCI address and data
PAD27	W30	I/OZ		PCI address and data
PAD28	W28	I/OZ		PCI address and data
PAD29	Y30	I/OZ		PCI address and data
PAD30	Y29	I/OZ		PCI address and data
PAD31	Y26	I/OZ		PCI address and data

1.7.7 ATM interface

1.7.7.1 UTOPIA management interface

Pin Name	Pin No.	I/O	Active Level	Function
UMMD	AG20	O		Management mode select
UMINT_B	AH19	I	L	Interrupt from PHY
UMRD_B	AK21	O	L	Management read enable
UMRDY_B	AJ19	I	L	Management data ready
UMRST_B	AK20	O	L	PHY reset
UMSL_B	AH20	O	L	PHY select
UMWR_B	AF20	O	L	Management write enable
UMAD0	AG29	O		PHY address
UMAD1	AH30	O		PHY address
UMAD2	AG28	O		PHY address
UMAD3	AH29	O		PHY address
UMAD4	AJ30	O		PHY address
UMAD5	AK30	O		PHY address
UMAD6	AJ29	O		PHY address
UMAD7	AH28	O		PHY address
UMAD8	AH27	O		PHY address
UMAD9	AG26	O		PHY address
UMAD10	AK28	O		PHY address
UMAD11	AG25	O		PHY address
UMD0	AK27	I/O		Management data
UMD1	AJ26	I/O		Management data
UMD2	AG24	I/O		Management data
UMD3	AK26	I/O		Management data
UMD4	AJ25	I/O		Management data
UMD5	AH24	I/O		Management data
UMD6	AK25	I/O		Management data
UMD7	AJ24	I/O		Management data
UMD8	AK24	I/O		Management data
UMD9	AJ23	I/O		Management data
UMD10	AG22	I/O		Management data
UMD11	AH22	I/O		Management data
UMD12	AK23	I/O		Management data
UMD13	AJ22	I/O		Management data
UMD14	AK22	I/O		Management data
UMD15	AH21	I/O		Management data

1.7.7.2 UTOPIA data interface

Pin Name	Pin No.	I/O	Active Level	Function
CLKUSL0	T4	I		UTOPIA clock select (CLKUSL1/0 = L/L: 33 MHz, H/L: 25 MHz, L/H: 16.5 MHz)
CLKUSL1	T3	I		
UDRCLK	AK11	O		Receive clock
UDRCLV	AH9	I	H	Receive cell available
UDRE_B	AJ8	O	L	Receive enable
UDRSC	AG9	I	H	Receive cell start
UDRAD0	AK13	O		Receive PHY address
UDRAD1	AG13	O		Receive PHY address
UDRAD2	AF13	O		Receive PHY address
UDRAD3	AK12	O		Receive PHY address
UDRAD4	AJ11	O		Receive PHY address
UDRD0	AG11	I		Receive data
UDRD1	AF11	I		Receive data
UDRD2	AK10	I		Receive data
UDRD3	AJ10	I		Receive data
UDRD4	AH10	I		Receive data
UDRD5	AK9	I		Receive data
UDRD6	AJ9	I		Receive data
UDRD7	AK8	I		Receive data
UDTCLK	AK16	O		Transmit clock
UDTCLV	AH11	I	H	Transmit cell available
UDTE_B	AK14	O	L	Transmit enable
UDTSC	AH14	O	H	Transmit cell start position
UDTAD0	AG16	O		Transmit PHY address
UDTAD1	AH16	O		Transmit PHY address
UDTAD2	AG15	O		Transmit PHY address
UDTAD3	AF15	O		Transmit PHY address
UDTAD4	AK15	O		Transmit PHY address
UDTD0	AK19	O		Transmit data
UDTD1	AG18	O		Transmit data
UDTD2	AH18	O		Transmit data
UDTD3	AK18	O		Transmit data
UDTD4	AH17	O		Transmit data
UDTD5	AJ17	O		Transmit data
UDTD6	AK17	O		Transmit data
UDTD7	AF16	O		Transmit data

1.7.8 Ethernet interface

1.7.8.1 Ethernet interface (Channel 1)

Pin Name	Pin No.	I/O	Active Level	Function
MIMCLK	AF3	O		MII management clock
MIMD	AG1	I/O		MII management
MICOL	AH3	I		Collision
MICRS	AF2	I		Carrier sense
MIRCLK	AD3	I		Receive clock (2.5/25 MHz)
MIRDV	AD5	I		Receive data valid
MIRER	AD4	I		Receive error
MIRD0	AE1	I		Receive data
MIRD1	AD2	I		Receive data
MIRD2	AD1	I		Receive data
MIRD3	AC2	I		Receive data
MITCLK	AH1	I		Transmit clock (2.5/25 MHz)
MITE	AF1	O		Transmit enable
MITER	AE3	O		Transmit error
MITD0	AJ1	O		Transmit data
MITD1	AH2	O		Transmit data
MITD2	AG3	O		Transmit data
MITD3	AF4	O		Transmit data

1.7.8.2 Ethernet interface (Channel 2)

Pin Name	Pin No.	I/O	Active Level	Function
MI2MCLK	AJ2	O		MII management clock
MI2MD	AH5	I/O		MII management
MI2COL	AG5	I		Collision
MI2CRS	AG7	I		Carrier sense
MI2RCLK	AK3	I		Receive clock (2.5/25 MHz)
MI2RDV	AK4	I		Receive data valid
MI2RER	AJ5	I		Receive error
MI2RD0	AH4	I		Receive data
MI2RD1	AJ3	I		Receive data
MI2RD2	AK2	I		Receive data
MI2RD3	AK1	I		Receive data
MI2TCLK	AK5	I		Transmit clock (2.5/25 MHz)
MI2TE	AF7	O		Transmit enable
MI2TER	AH6	O		Transmit error
MI2TD0	AK7	O		Transmit data
MI2TD1	AJ7	O		Transmit data
MI2TD2	AK6	O		Transmit data
MI2TD3	AH7	O		Transmit data

1.7.9 USB interface

Pin Name	Pin No.	I/O	Active Level	Function
USBCLK	E24	I		External USB clock
USBDM	C23	I/O		USB data(–)
USBDP	B24	I/O		USB data(+)

1.7.10 UART interface

Pin Name	Pin No.	I/O	Active Level	Function
URCLK	D9	I		UART external clock
URCTS_B	B8	I	L	UART clear to send
URDCD_B	A9	I	L	UART data carrier detect
URDSR_B	A8	I	L	UART data set ready
URDTR_B	A10	O	L	UART data terminal ready
URRTS_B	C10	O	L	UART data request to send
URSDI	B9	I		UART serial data input
URSDO	B10	O		UART serial data output

1.7.11 Micro Wire interface

Pin Name	Pin No.	I/O	Active Level	Function
MWCS	C12	O		Micro Wire chip select
MWDI	A12	I		Micro Wire data in
MWDO	B12	O		Micro Wire data out
MWSK	A11	O		Micro Wire SK

1.7.12 Parallel port interface

Pin Name	Pin No.	I/O	Active Level	Function
POM0	E13	O		Parallel port signal output
POM1	D13	O		Parallel port signal output
POM2	C13	O		Parallel port signal output
POM3	B13	O		Parallel port signal output
POM4	C14	O		Parallel port signal output
POM5	B14	O		Parallel port signal output
POM6	A14	O		Parallel port signal output
POM7	E15	O		Parallel port signal output

1.7.13 Boundary scan interface

Pin Name	Pin No.	I/O	Active Level	Function
JCK	A22	I		B-SCAN clock
JDI	C21	I		B-SCAN input-data
JDO	B22	t/s O		B-SCAN output-data
JMS	A23	I		B-SCAN mode select
JRSTB_B	D22	I	L	B-SCAN reset

1.7.14 I.C. – open

Pin Name	Pin No.	I/O	Active Level	Function
IC-OPEN	A17, A19, A20, A28, B16, B17, B18, B19, B26, C20, C24, D18, D20, E18, Y1, AA1, AB1, AB27, AB28, AC28, AC29, AD29, AH12, AJ12	O		

1.7.15 I.C.– pull down

Pin Name	Pin No.	I/O	Active Level	Function
IC-PDn	A21, B21, E20, AB4	I		

1.7.16 I.C. – pull down with resistor

Pin Name	Pin No.	I/O	Active Level	Function
IC-PDnR	A18, D15, AE30, AD28, AE27, AE28, AE29, AF28, AF29, AF30	I/O		

1.7.17 I.C. – pull up

Pin Name	Pin No.	I/O	Active Level	Function
IC-PUp	U3, V3, W1, W3, AB3, AC1, AK29	I		

1.8 I/O Register Map

Core	Offset	Register Length (Byte)	Name	Access by VR4120A	Description
ATM	F000H	4	A_GMR	R/W	General Mode Register
ATM	F004H	4	A_GSR	R	General Status Register
ATM	F008H	4	A_IMR	R/W	Interrupt Mask Register
ATM	F00CH	4	A_RQU	R	Receive Queue Underrunning
ATM	F010H	4	A_RQA	R	Receive Queue Alert
ATM	F014H	-	N/A	-	Reserved for future use
ATM	F018H	4	A_VER	R	Version Number
ATM	F01CH	-	N/A	-	Reserved for future use
ATM	F020H	4	A_CMR	R/W	Command Register
ATM	F024H	-	N/A	-	Reserved for future use
ATM	F028H	4	A_CER	R/W	Command Extension Register
ATM	F02CH-F04CH	-	N/A	-	Reserved for future use
ATM	F050H	4	A_MSA0	R/W	Mailbox0 Start Address
ATM	F054H	4	A_MSA1	R/W	Mailbox1 Start Address
ATM	F058H	4	A_MSA2	R/W	Mailbox2 Start Address
ATM	F05CH	4	A_MSA3	R/W	Mailbox3 Start Address
ATM	F060H	4	A_MBA0	R/W	Mailbox0 Bottom Address
ATM	F064H	4	A_MBA1	R/W	Mailbox1 Bottom Address
ATM	F068H	4	A_MBA2	R/W	Mailbox2 Bottom Address
ATM	F06CH	4	A_MBA3	R/W	Mailbox3 Bottom Address
ATM	F070H	4	A_MTA0	R/W	Mailbox0 Tail Address
ATM	F074H	4	A_MTA1	R/W	Mailbox1 Tail Address
ATM	F078H	4	A_MTA2	R/W	Mailbox2 Tail Address
ATM	F07CH	4	A_MTA3	R/W	Mailbox3 Tail Address
ATM	F080H	4	A_MWA0	R/W	Mailbox0 Write Address
ATM	F084H	4	A_MWA1	R/W	Mailbox1 Write Address
ATM	F088H	4	A_MWA2	R/W	Mailbox2 Write Address
ATM	F08CH	4	A_MWA3	R/W	Mailbox3 Write Address
ATM	F090H	4	A_RCC	R	Valid Receiving Cell Counter
ATM	F094H	4	A_TCC	R	Valid Transmitting Cell Counter
ATM	F098H	4	A_RUEC	R	Receive Unprovisioned VPI/VCI Error Cell Counter
ATM	F09CH	4	A_RIDC	R	Receiving Internal Discarded Cell Counter
ATM	F0A0H-F0AFH	-	N/A	-	Reserved for future use
ATM	F0B0H-F0B3H	-	N/A	-	Reserved for future use
ATM	F0B4H-F0BCH	-	N/A	-	Reserved for future use
ATM	F0C0H	4	A_T1R	R/W	T1 Timer Register
ATM	F0C4H	-	N/A	-	Reserved for future use
ATM	F0C8H	4	A_TSR	R/W	Time Stamp Register
ATM	F200H-F2FFH	-	N/A	-	Can not access from VR4120A RISC Core.
ATM	F300H	4	A_IBBAR	R/W	IBUS Base Address Register
ATM	F304H	4	A_INBAR	R/W	Instruction Base Address Register
ATM	F308H- F31FH	-	N/A	-	Reserved for future use
ATM	F320H	4	A_UMCMD	R/W	UTOPIA Management Interface Command Register
ATM	F324H- F3FFH	-	N/A	-	Reserved for future use
ATM	F400H-F4FFH	-	N/A	-	Can not access from VR4120A RISC Core.
ATM	F500H-FFFFH	-	N/A	-	Reserved for future use
PCI	000H	4	P_PLBA	R/W	PCI Lower Base Address
PCI	008H	4	P_IBBA	R/W	Internal bus Base Address
PCI	000H	4	P_PLBA	R/W	PCI Lower Base Address
PCI	008H	4	P_IBBA	R/W	Internal bus Base Address
PCI	00CH	4	N/A	-	Reserved for future use
PCI	010H	4	P_VERR	R	Version Register
PCI	014H	4	P_PCAR	R/W	PCI Configuration Address Register
PCI	018H	4	P_PCDR	R/W	PCI Configuration Data Register
PCI	01CH	4	P_IGSR	R	Internal bus General Status Register
PCI	020H	4	P_IIMR	R/W	Internal bus Interrupt Mask Register
PCI	024H	4	P_PGSR	R/W	PCI General Status Register
PCI	028H	4	P_PIMR	R/W	PCI Interrupt Mask Register
PCI	02CH	4	N/A	-	Reserved for future use
PCI	030H	4	P_HMCR	R/W	Host Mode Control Register
PCI	034H-03CH	4	N/A	-	Reserved for future use
PCI	040H	4	P_PWCD	R/W	Power Consumption Data Register
PCI	044H	4	P_PWDD	R/W	Power Dissipation Data Register

Core	Offset	Register Length (Byte)	Name	Access by VR4120A	Description
PCI	048H-04CH	4	N/A	-	Reserved for future use
PCI	050H	4	P_BCNT	R/W	Bridge Control Register
PCI	054H	4	P_PPCR	R/W	Power Control Register
PCI	058H	4	P_SWRR	W	Software Reset Register
PCI	05CH	4	P_PTMR	R/W	Retry Timer Register
PCI	060H-0FFH	4	N/A	-	Reserved for future use
PCI	100H-1FFH	4	P_CONFIG	(*)	Configuration Registers. * Some registers are R/W. Other registers are Read only.
Ether	00H	4	En_MACC1	R/W	MAC configuration register 1
Ether	04H	4	En_MACC2	R/W	MAC configuration register 2
Ether	08H	4	En_IPGT	R/W	Back-to-Back IPG register
Ether	0CH	4	En_IPGR	R/W	Non Back-to-Back IPG register
Ether	10H	4	En_CLRT	R/W	Collision register
Ether	14H	4	En_LMAX	R/W	Max packet length register
Ether	18H-1CH	-	N/A	-	Reserved for future use
Ether	20H	4	En_RETX	R/W	Retry count register
Ether	24H-50H	-	N/A	-	Reserved for future use
Ether	54H	4	En_LSA2	R/W	Station Address register 2
Ether	58H	4	En_LSA1	R/W	Station Address register 1
Ether	5CH	4	En_PTVR	R	Pause timer value read register
Ether	60H	-	N/A	-	Reserved for future use
Ether	64H	4	En_VLTP	R/W	VLAN type register
Ether	80H	4	En_MIIC	R/W	MII configuration register
Ether	84H-90H	-	N/A	-	Reserved for future use
Ether	94H	4	En_MCMD	W	MII command register
Ether	98H	4	En_MADR	R/W	MII address register
Ether	9CH	4	En_MWTD	R/W	MII write data register
Ether	A0H	4	En_MRDD	R	MII read data register
Ether	A4H	4	En_MIND	R	MII indicator register
Ether	A8H-C4H	-	N/A	-	Reserved for future use
Ether	CCH	4	En_HT1	R/W	Hash table register 1
Ether	D0H	4	En_HT2	R/W	Hash table register 2
Ether	D4H-D8H	-	N/A	-	Reserved for future use
Ether	DCH	4	En_CAR1	R/W	Carry register 1
Ether	E0H	4	En_CAR2	R/W	Carry register 2
Ether	E4H-12CH	-	N/A	-	Reserved for future use
Ether	130H	4	En_CAM1	R/W	Carry mask register 1
Ether	134H	4	En_CAM2	R/W	Carry mask register 2
Ether	138H-13CH	-	N/A	-	Reserved for future use
Ether	140H	4	En_RBYT	R/W	Receive Byte Counter
Ether	144H	4	En_RPKT	R/W	Receive Packet Counter
Ether	148H	4	En_RFCS	R/W	Receive FCS Error Counter
Ether	14CH	4	En_RMCA	R/W	Receive Multicast Packet Counter
Ether	150H	4	En_RBCA	R/W	Receive Broadcast Packet Counter
Ether	154H	4	En_RXCF	R/W	Receive Control Frame Packet Counter
Ether	158H	4	En_RXPF	R/W	Receive PAUSE Frame Packet Counter
Ether	15CH	4	En_RXUO	R/W	Receive Unknown OP code Counter
Ether	160H	4	En_RALN	R/W	Receive Alignment Error Counter
Ether	164H	4	En_RFLR	R/W	Receive Frame Length Out of Range Counter
Ether	168H	4	En_RCDE	R/W	Receive Code Error Counter
Ether	16CH	4	En_RFCR	R/W	Receive False Carrier Counter
Ether	170H	4	En_RUND	R/W	Receive Undersize Packet Counter
Ether	174H	4	En_ROVR	R/W	Receive Oversize Packet Counter
Ether	178H	4	En_RFRG	R/W	Receive Error Undersize Packet Counter
Ether	17CH	4	En_RJBR	R/W	Receive Error Oversize Packet Counter
Ether	180H	4	En_R64	R/W	Receive 64 Byte Frame Counter
Ether	184H	4	En_R127	R/W	Receive 65 to 127 Byte Frame Counter
Ether	188H	4	En_R255	R/W	Receive 128 to 255 Byte Frame Counter
Ether	18CH	4	En_R511	R/W	Receive 256 to 511 Byte Frame Counter
Ether	190H	4	En_R1K	R/W	Receive 512 to 1023 Byte Frame Counter
Ether	194H	4	En_RMAX	R/W	Receive Over 1023 Byte Frame Counter
Ether	198H	4	En_RVBT	R/W	Receive Valid Byte Counter
Ether	1C0H	4	En_TBYT	R/W	Transmit Byte Counter
Ether	1C4H	4	En_TPCT	R/W	Transmit Packet Counter
Ether	1C8H	4	En_TFCS	R/W	Transmit CRC Error Packet Counter
Ether	1CCH	4	En_TMCA	R/W	Transmit Multicast Packet Counter

Core	Offset	Register Length (Byte)	Name	Access by VR4120A	Description
Ether	1D0H	4	En_TBCA	R/W	Transmit Broadcast Packet Counter
Ether	1D4H	4	En_TUCA	R/W	Transmit Unicast Packet Counter
Ether	1D8H	4	En_TXPF	R/W	Transmit PAUSE control Frame Counter
Ether	1DCH	4	En_TDFR	R/W	Transmit Single Deferral Packet Counter
Ether	1E0H	4	En_TXDF	R/W	Transmit Excessive Deferral Packet Counter
Ether	1E4H	4	En_TSCL	R/W	Transmit Single Collision Packet Counter
Ether	1E8H	4	En_TMCL	R/W	Transmit Multiple collision Packet Counter
Ether	1ECH	4	En_TLCL	R/W	Transmit Late Collision Packet Counter
Ether	1F0H	4	En_TXCL	R/W	Transmit Excessive Collision Packet Counter
Ether	1F4H	4	En_TNCL	R/W	Transmit Total Collision Counter
Ether	1F8H	4	En_TCSE	R/W	Transmit Carrier Sense Error Counter
Ether	1FCH	4	En_TIME	R/W	Transmit Internal MAC Error Counter
Ether	200H	4	En_TXCR	R/W	Transmit Configuration Register
Ether	204H	4	En_TXFCR	R/W	Transmit FIFO Control Register
Ether	208H	4	En_TXDTR	W	Transmit Data Register
Ether	20CH	4	En_TXSR	R	Transmit Status Register
Ether	210H	4	N/A	-	Reserved for future use
Ether	214H	4	En_TXDPR	R/W	Transmit Descriptor Register
Ether	218H	4	En_RXCR	R/W	Receive Configuration Register
Ether	21CH	4	En_RXFCR	R/W	Receive FIFO Control Register
Ether	220H	4	En_RXDTR	R	Receive Data Register
Ether	224H	4	En_RXSR	R	Receive Status Register
Ether	228H	4	N/A	-	Reserved for future use
Ether	22CH	4	En_RXDPR	R/W	Receive Descriptor Register
Ether	230H	4	En_RXPDR	R/W	Receive Pool Descriptor Register
SYSCNT	00H	4	S_GMR	R/W	General Mode Register
SYSCNT	04H	4	S_GSR	R	General Status Register
SYSCNT	08H	4	S_ISR	RC	Interrupt Status Register
SYSCNT	0CH	4	S_IMR	W	Interrupt Mask Register
SYSCNT	10H	4	S_NSR	R	NMI Status Register
SYSCNT	14H	4	S_NER	R/W	NMI Enable Register
SYSCNT	18H	4	S_VER	R	Version Register
SYSCNT	1CH	4	S_IOR	R/W	IO Port Register
SYSCNT	20H-2FH	-	N/A	-	Reserved for future use
SYSCNT	30H	4	S_WRCR	W	Warm Reset Control Register
SYSCNT	34H	4	S_WRSR	R	Warm Reset Status Register
SYSCNT	38H	4	S_PWCR	W	Power Control Register
SYSCNT	3CH	4	S_PWSR	R	Power Control Status Register
SYSCNT	40H-48H	-	N/A	-	Reserved for future use
SYSCNT	4CH	4	S_ITCNTR	R/W	IBUS Timeout Timer Control Register
SYSCNT	50H	4	S_ITSETR	R/W	IBUS Timeout Timer Set Register
SYSCNT	54H-7FH	-	N/A	-	Reserved for future use
SYSCNT	80H	4	UARTDLL	R/W	UART, Divisor Latch LSB Register [DLAB=1]
SYSCNT	80H	4	UARTRBR	R	UART, Receiver Buffer Register [DLAB=0,READ]
SYSCNT	80H	4	UARTTHR	W	UART, Transmitter Holding Register [DLAB=0,WRITE]
SYSCNT	84H	4	UARTDLM	R/W	UART, Divisor Latch MSB Register [DLAB=1]
SYSCNT	84H	4	UARTIER	R/W	UART, Interrupt Enable Register [DLAB=0]
SYSCNT	88H	4	UARTFCR	W	UART, FIFO control Register [WRITE]
SYSCNT	88H	4	UARTIIR	R	UART, Interrupt ID Register [READ]
SYSCNT	8CH	4	UARTLCR	R/W	UART, Line control Register
SYSCNT	90H	4	UARTMCR	R/W	UART, Modem Control Register
SYSCNT	94H	4	UARTLSR	R/W	UART, Line status Register
SYSCNT	98H	4	UARTMSR	R/W	UART, Modem Status Register
SYSCNT	9CH	4	UARTSCR	R/W	UART, Scratch Register
SYSCNT	A0H	4	DSUCNTR	R/W	DSU Control Register
SYSCNT	A4H	4	DSUSETR	R/W	DSU Dead Time Set Register
SYSCNT	A8H	4	DSUCLRR	W	DSU Clear Register
SYSCNT	ACH	4	DSUTIMR	R/W	DSU Elapsed Time Register
SYSCNT	B0H	4	TMMR	R/W	Timer Mode Register
SYSCNT	B4H	4	TM0CSR	R/W	Timer CH0 Count Set Register
SYSCNT	B8H	4	TM1CSR	R/W	Timer CH1 Count Set Register
SYSCNT	BCH	4	TM0CCR	R	Timer CH0 Current Count Register
SYSCNT	C0H	4	TM1CCR	R	Timer CH1 Current Count Register
SYSCNT	C4H-CFH	-	N/A	-	Reserved for future use
SYSCNT	D0H	4	ECCR	W	EEPROM™ Command Control Register
SYSCNT	D4H	4	ERDR	R	EEPROM Read Data Register

Core	Offset	Register Length (Byte)	Name	Access by VR4120A	Description
SYSCNT	D8H	4	MACAR1	R	MAC Address Register 1
SYSCNT	DCH	4	MACAR2	R	MAC Address Register 2
SYSCNT	E0H	4	MACAR3	R	MAC Address Register 3
SYSCNT	E4H-FFH	-	N/A	-	Reserved for future use
SYSCNT	100H	4	RMMDR	R/W	Boot ROM Mode Register
SYSCNT	104H	4	RMATR	R/W	Boot ROM Access Timing Register
SYSCNT	108H	4	SDMDR	R/W	SDRAM Mode Register
SYSCNT	10CH	4	SDTSR	R/W	SDRAM Type Selection Register
SYSCNT	110H	4	SDPTR	R/W	SDRAM Precharge Timing Register
SYSCNT	114H	4	SDRMR	R/W	SDRAM Precharge Mode Register
SYSCNT	118H	4	SDRCR	R	SDRAM Precharge Timer Count Register
SYSCNT	11CH	4	SDRMR	R/W	SDRAM Refresh Mode Register
SYSCNT	120H	4	SDRCR	R	SDRAM Refresh Timer Count Register
SYSCNT	124H	4	MBCR	R/W	Memory Bus Control Register
SYSCNT	128H-FFFH	-	N/A	-	Reserved for future use
USB	00H	4	U_GMR	R/W	USB General Mode Register
USB	04H	4	U_VER	R	USB Frame number/Version Register
USB	08H	-	N/A	-	Reserved for future use
USB	0CH	-	N/A	-	Reserved for future use
USB	10H	4	U_GSR1	R	USB General Status Register 1
USB	14H	4	U_IMR1	R/W	USB Interrupt Mask Register 1
USB	18H	4	U_GSR2	R	USB General Status Resister 2
USB	1CH	4	U_IMR2	R/W	USB Interrupt Mask Register 2
USB	20H	4	U_EP0CR	R/W	USB EP0 Control Register
USB	24H	4	U_EP1CR	R/W	USB EP1 Control Register
USB	28H	4	U_EP2CR	R/W	USB EP2 Control Register
USB	2CH	4	U_EP3CR	R/W	USB EP3 Control Register
USB	30H	4	U_EP4CR	R/W	USB EP4 Control Register
USB	34H	4	U_EP5CR	R/W	USB EP5 Control Register
USB	38H	4	U_EP6CR	R/W	USB EP6 Control Register
USB	3CH	-	N/A	-	Reserved for future use
USB	40H	4	U_CMR	R/W	USB Command Register
USB	44H	4	U_CA	R/W	USB Command Address Register
USB	48H	4	U_TEPSR	R	USB Tx EndPoint Status Register
USB	4CH	-	N/A	-	Reserved for future use
USB	50H	4	U_RP0IR	R/W	USB Rx Pool0 Information Register
USB	54H	4	U_RP0AR	R	USB Rx Pool0 Address Register
USB	58H	4	U_RP1IR	R/W	USB Rx Pool1 Information Register
USB	5CH	4	U_RP1AR	R	USB Rx Pool1 Address Register
USB	60H	4	U_RP2IR	R/W	USB Rx Pool2 Information Register
USB	64H	4	U_RP2AR	R	USB Rx Pool2 Address Register
USB	68H	-	N/A	-	Reserved for future use
USB	6CH	-	N/A	-	Reserved for future use
USB	70H	4	U_TMSA	R/W	USB Tx MailBox Start Address Register
USB	74H	4	U_TMBA	R/W	USB Tx MailBox Bottom Address Register
USB	78H	4	U_TMRA	R/W	USB Tx MailBox Read Address Register
USB	7CH	4	U_TMWA	R	USB Tx MailBox Write Address Register
USB	80H	4	U_RMSA	R/W	USB Rx MailBox Start Address Register
USB	84H	4	U_RMBA	R/W	USB Rx MailBox Bottom Address Register
USB	88H	4	U_RMRA	R/W	USB Rx MailBox Read Address Register
USB	8CH	4	U_RMWA	R	USB Rx MailBox Write Address Register
USB	90H-FFH	-	N/A	-	Reserved for future use

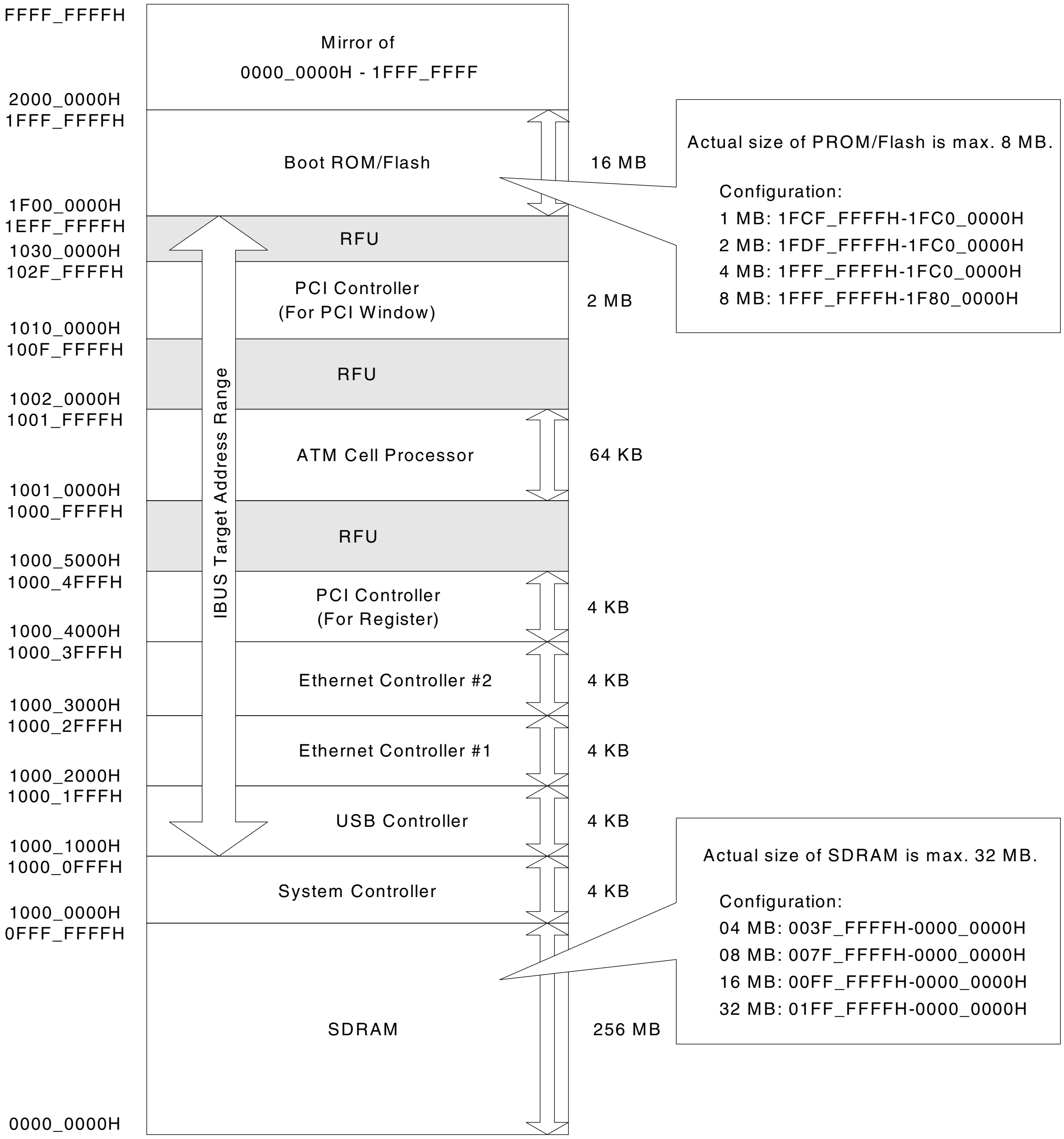
Base address

ATM Cell Processor (ATM)	- 1001_0000H
PCI Controller (PCI)	- 1000_4000H
Ethernet Controller (Ether) #1 (n = 1)	- 1000_2000H
Ethernet Controller (Ether) #2 (n = 2)	- 1000_3000H
USB Controller (USB)	- 1000_1000H
System Controller (SYSCNT)	- 1000_0000H

1.9 Memory Map

Using a 32-bit address, the processor physical address space encompasses 4 Gbytes. VR4120A uses this 4-Gbyte physical address space as shown in the following figure.

Figure 1-10. Memory Map



Actual size of PROM/Flash is max. 8 MB.

Configuration:
1 MB: 1FCF_FFFFH-1FC0_0000H
2 MB: 1FDF_FFFFH-1FC0_0000H
4 MB: 1FFF_FFFFH-1FC0_0000H
8 MB: 1FFF_FFFFH-1F80_0000H

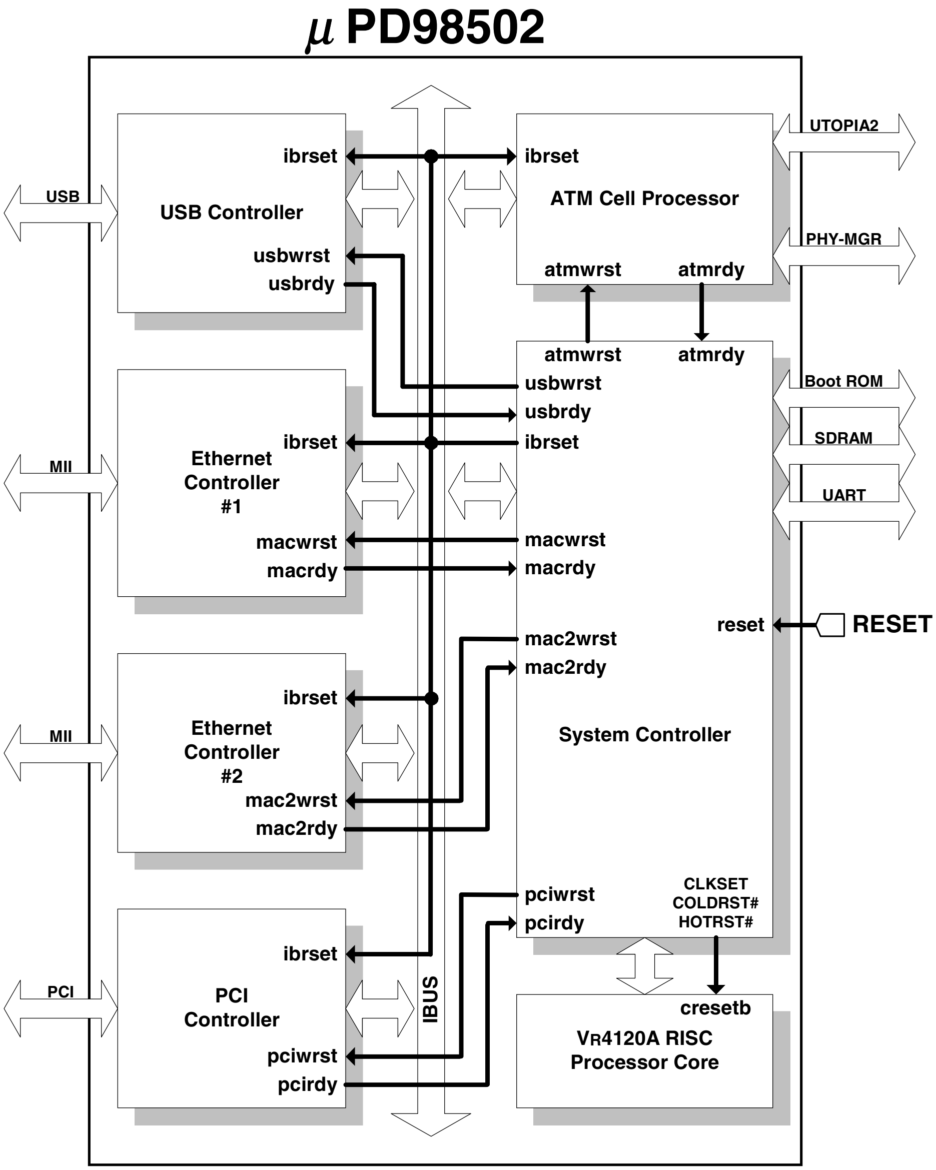
Actual size of SDRAM is max. 32 MB.

Configuration:
04 MB: 003F_FFFFH-0000_0000H
08 MB: 007F_FFFFH-0000_0000H
16 MB: 00FF_FFFFH-0000_0000H
32 MB: 01FF_FFFFH-0000_0000H

1.10 Reset Configuration

The falling edge of Clock Control Unit (CCU)’s reset line (RST_B) serves as the μPD98502's internal reset. The System Controller generates the IBUS reset signal using RST_B for the global reset of the μPD98502. After 4 IBUS clock (SDCLK), the System Controller deasserts the IBUS reset signal synchronously with IBUS clock (66 MHz). And also the System Controller generates the internal Cold Reset signal and Hot Reset signal for performing the cold reset of VR4120A. Once power to the μPD98502 is established, the System Controller asserts internal CLKSET signal, internal Cold Reset (COLDRST#) signal and internal Hot Reset (HOTRST#) signal at the falling edge of RST_B signal. After 2 VR4120A clock (internal VCLOCK) cycles at rising edge of the RST_B, the System Controller deasserts the CLKSET signal synchronously with “clkm”. Then 16 “clkm” cycles (see section **1.12**) at the rising edge of the RST_B signal, the System Controller deasserts the COLDRST# synchronously with “clkm”. And also the System Controller deasserts the HOTRST# synchronously with “clkm” after 16 “clkm” clock cycles at deassertion of the COLDRST#.

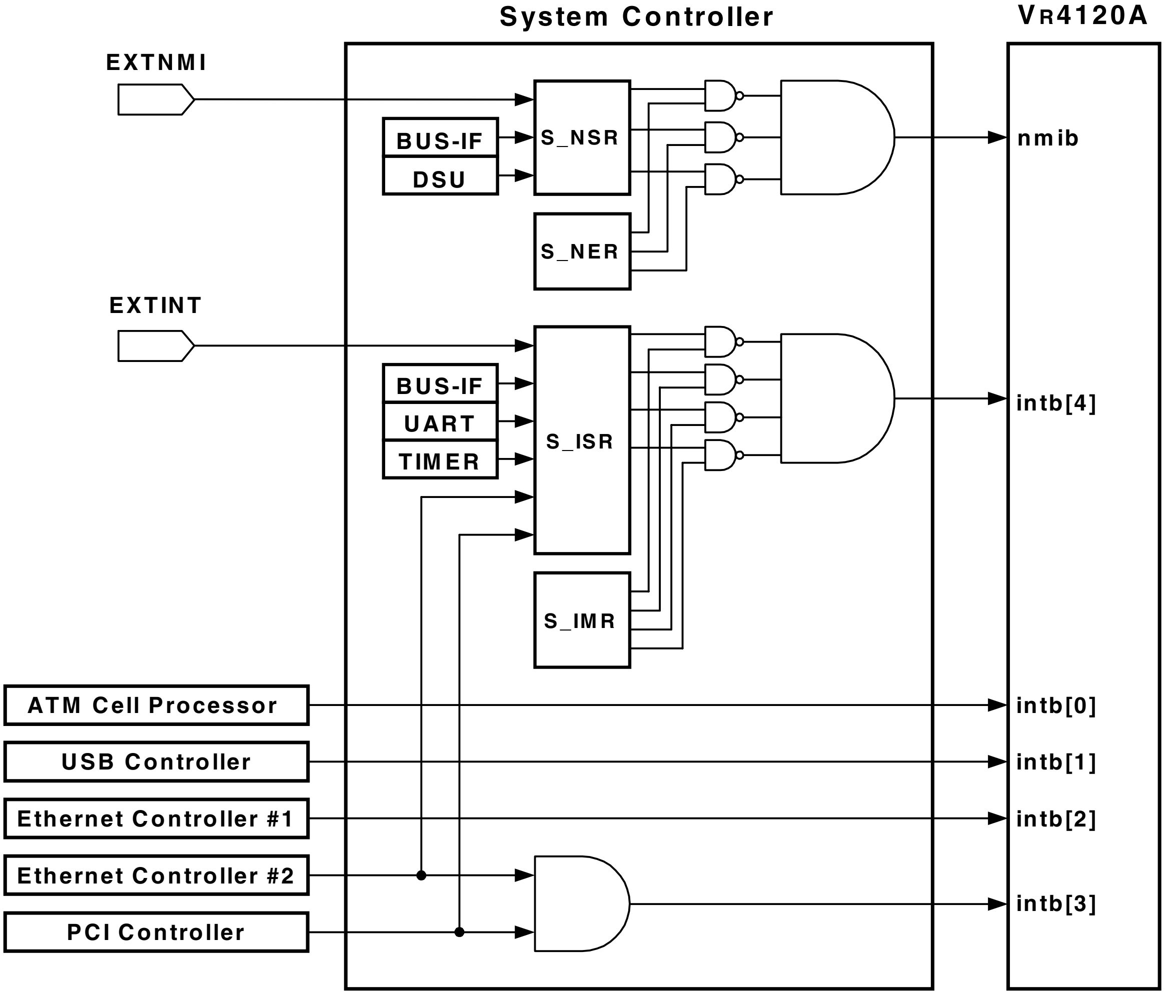
Figure 1-11. Reset Configuration



1.11 Interrupts

The controller supports maskable interrupts and Non-Maskable to the CPU.

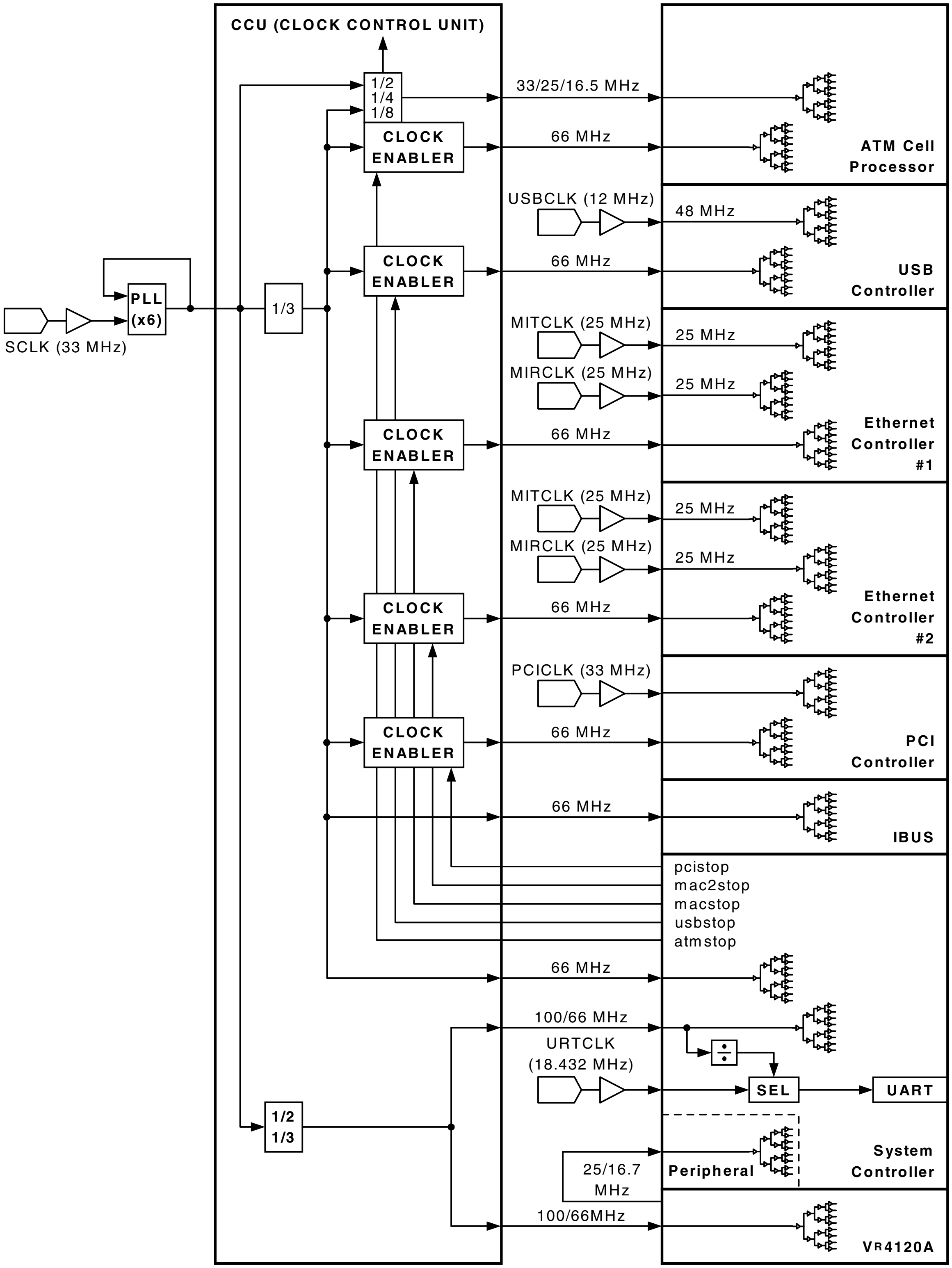
Figure 1-12. Interrupt Signal Connection



1.12 Clock Control Unit

This section describe μPD98502's internal clock is supplied by Clock Control Unit (CCU) with following figure.

Figure 1-13. Block Diagram of Clock Control Unit



Caution The μPD98502 doesn't support MIPS16 instructions.

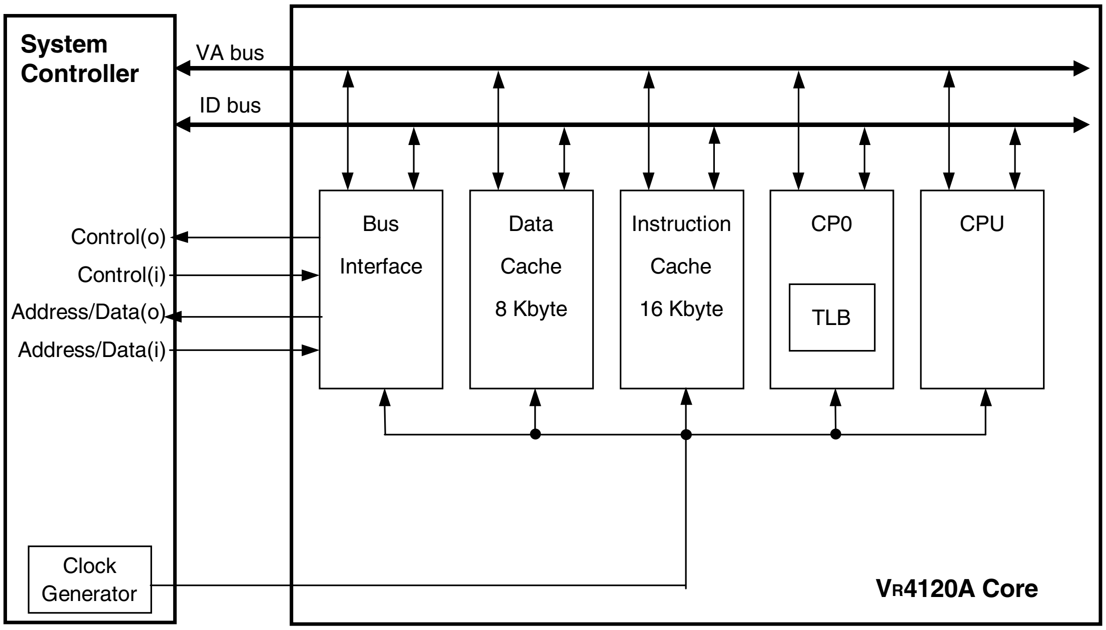
This chapter describes an VR4120A RISC Processor Core operation (MIPS instruction, Pipeline, etc.). Following in this Document, it is call for VR4120A RISC Processor Core with "VR4120A" or "VR4120A Core" simply.

2.1 Overview for VR4120A

Figure 2-1 shows the internal block diagram of the VR4120A core.

In addition to the conventional high-performance integer operation units, this CPU core has the full-associative format translation look aside buffer (TLB), which has 32 entries that provide mapping to 2-page pairs (odd and even) for one entry. Moreover, it also has instruction caches, data caches, and bus interface.

Figure 2-1. VR4120A Core Internal Block Diagram



2.1.1 Internal block configuration

2.1.1.1 CPU

CPU has hardware resources to process an integer instruction. They are the 64-bit register file, 64-bit integer data bus, and multiply-and-accumulate operation unit.

2.1.1.2 Coprocessor 0 (CP0)

CP0 incorporates a memory management unit (MMU) and exception handling function. MMU checks whether there is an access between different memory segments (user, supervisor, and kernel) by executing address conversion. The translation lookaside buffer (TLB) converts virtual addresses to physical addresses.

2.1.1.3 Instruction cache

The instruction cache employs direct mapping, virtual index, and physical tag. Its capacity is 16 Kbytes.

2.1.1.4 Data cache

The data cache employs direct mapping, virtual index, physical tag, and write back. Its capacity is 8 Kbytes.

2.1.1.5 CPU bus interface

The CPU bus interface controls data transmission/reception between the VR4120A and the BCU, which is one of peripheral units. The VR4120A interface consists of two 32-bit multiplexed address/data buses (one is for input, and another is for output), clock signals, and control signals such as interrupts.

2.1.2 VR4120A registers

The VR4120A has the following registers.

- ✧ general-purpose register (GPR): 64 bits × 32

In addition, the processor provides the following special registers:

- ✧ 64-bit Program Counter (PC)
- ✧ 64-bit HI register, containing the integer multiply and divide upper doubleword result
- ✧ 64-bit LO register, containing the integer multiply and divide lower doubleword result

Two of the general-purpose registers have assigned the following functions:

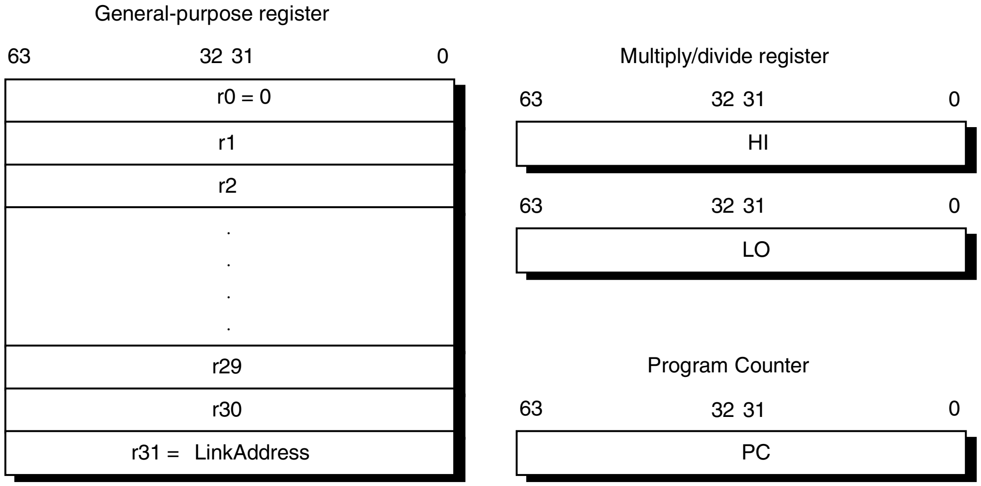
- ✧ r0 is hardwired to a value of zero, and can be used as the target register for any instruction whose result is to be discarded. r0 can also be used as a source when a zero value is needed.
- ✧ r31 is the link register used by link instruction, such as JAL (Jump and Link) instructions. This register can be used for other instructions. However, be careful that use of the register by a link instruction will not coincide with use of the register for other operations.

The register group is provided within the CP0 (system control coprocessor), to process exceptions and to manage addresses.

CPU registers can operate as either 32-bit or 64-bit registers, depending on the VR4120A processor operation mode.

Figure 2-2 shows the CPU registers.

Figure 2-2. VR4120A Registers



The VR4120A has no Program Status Word (PSW) register as such; this is covered by the Status and Cause registers incorporated within the System Control Coprocessor (CP0).

The CP0 registers are used for exception handling or address management. The overview of these registers is described in **2.1.5 Coprocessors (CP0)**.

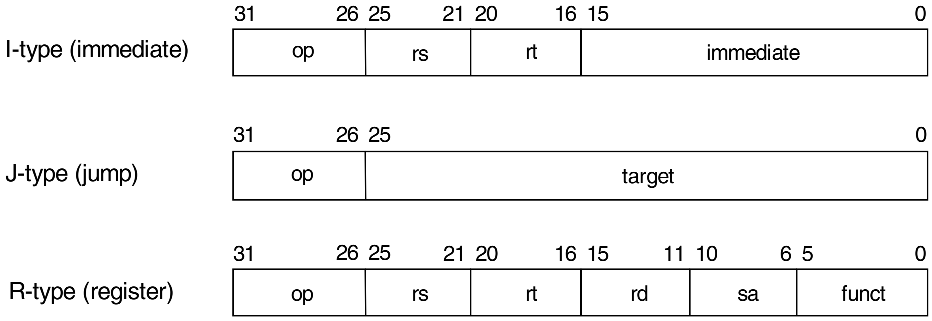
2.1.3 VR4120A instruction set overview

For CPU instructions, there are only one type of instructions – 32-bit length instruction (MIPS III).

2.1.3.1 MIPS III instruction

All the CPU instructions are 32-bit length when executing MIPS III instructions, and they are classified into three instruction formats as shown in Figure 2-3: immediate (I-type), jump (J-type), and register (R-type). The field of each instruction format is described in **Section 2.2 MIPS III Instruction Set Summary**.

Figure 2-3. CPU Instruction Formats (32-bit Length Instruction)



The instruction set can be further divided into the following five groupings:

(a) Load and store instructions move data between memory and general-purpose registers. They are all immediate (I-type) instructions, since the only addressing mode supported is base register plus 16-bit, signed immediate offset.
(b) Computational instructions perform arithmetic, logical, shift, and multiply and divide operations on values in registers. They include R-type (in which both the operands and the result are stored in registers) and I-type (in which one operand is a 16-bit signed immediate value) formats.
(c) Jump and branch instructions change the control flow of a program. Jumps are always made to an absolute address formed by combining a 26-bit target address with the high-order bits of the Program Counter (J-type format) or register address (R-type format). The format of the branch instructions is I type. Branches have 16-bit offsets relative to the Program Counter. JAL instructions save their return address in register 31.
(d) Coprocessor 0 (System Control Coprocessor, CP0) instructions perform operations on CP0 registers to control the memory-management and exception-handling facilities of the processor.
(e) Special instructions perform system calls and breakpoint operations, or cause a branch to the general exception-handling vector based upon the result of a comparison. These instructions occur in both R-type and I-type formats.

For the operation of each instruction, refer to **Section 2.2 MIPS III Instruction Set Summary** and **APPENDIX A MIPS III INSTRUCTION SET DETAILS**.

2.1.4 Data formats and addressing

The VR4120A uses following four data formats:

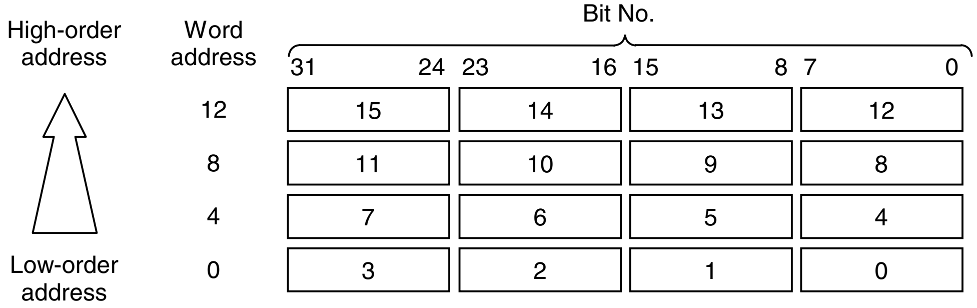
- ✧ Doubleword (64 bits)
- ✧ Word (32 bits)
- ✧ Halfword (16 bits)
- ✧ Byte (8 bits)

For the μPD98502, byte ordering within all of the larger data formats - halfword, word, doubleword - can be configured in either big-endian or little-endian order.

Endianness refers to the location of byte 0 within the multi-byte data structure.

When configured as a little-endian system, byte 0 is always the least-significant (rightmost) byte, which is compatible with iAPX™ and DEC VAX™ conventions. Figures 2-4 and 2-5 show this configuration.

Figure 2-4. Little-Endian Byte Ordering in Word Data

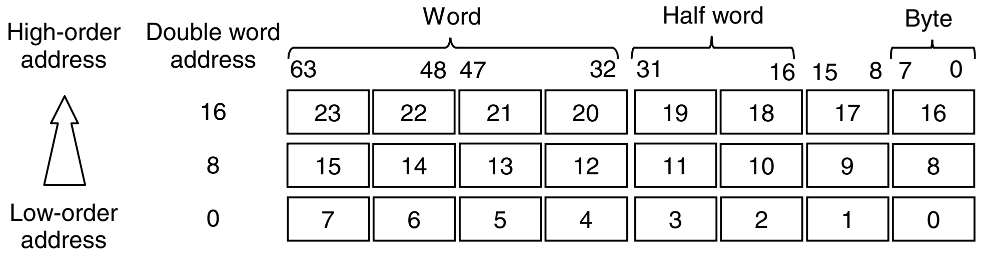


Word address	31 24	23 16	15 8	7 0
12	15	14	13	12
8	11	10	9	8
4	7	6	5	4
0	3	2	1	0

Remarks 1. The lowest byte is the lowest address.

2. The address of word data is specified by the lowest byte's address.

Figure 2-5. Little-Endian Byte Ordering in Double Word Data



Double word address	63	48	47	32	31	16	15 8	7 0
16	23	22	21	20	19	18	17	16
8	15	14	13	12	11	10	9	8
0	7	6	5	4	3	2	1	0

Remarks 1. The lowest byte is the lowest address.

2. The address of word data is specified by the lowest byte's address.

The CPU core uses the following byte boundaries for halfword, word, and doubleword accesses:

- ✧ Halfword: An even byte boundary (0, 2, 4...)
- ✧ Word: A byte boundary divisible by four (0, 4, 8...)
- ✧ Doubleword: A byte boundary divisible by eight (0, 8, 16...)

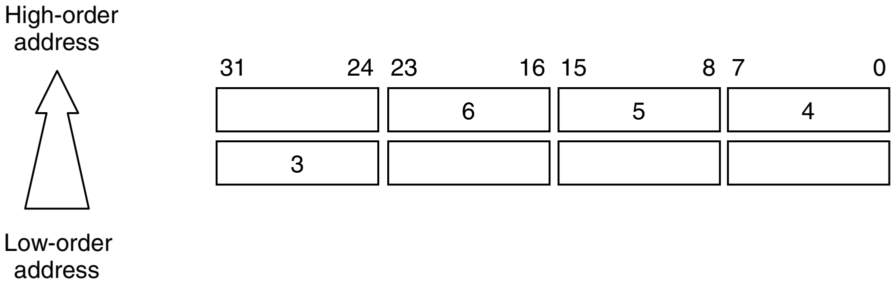
The following special instructions to load and store data that are not aligned on 4-byte (word) or 8-byte (doubleword) boundaries:

LWL	LWR	SWL	SWR
LDL	LDR	SDL	SDR

These instructions are used in pairs to provide an access to misaligned data. Accessing misaligned data incurs one additional instruction cycle over that required for accessing aligned data.

Figure 2-6 shows the access of a misaligned word that has byte address 3 for the little-endian conventions.

Figure 2-6. Misaligned Word Accessing (Little-Endian)



2.1.5 Coprocessors (CP0)

MIPS ISA defines 4 types of coprocessors (CP0 to CP3).

- CP0 translates virtual addresses to physical addresses, switches the operating mode (kernel, supervisor, or user mode), and manages exceptions. It also controls the cache subsystem to analyze a cause and to return from the error state.
- CP1 is reserved for floating-point instructions.
- CP2 is reserved for future definition by MIPS.
- CP3 is no longer defined. CP3 instructions are reserved for future extensions.

Figure 2-7 shows the definitions of the CP0 register, and Table 2-1 shows simple descriptions of each register. For the detailed descriptions of the registers related to the virtual system memory, refer to **Section 2.4 Memory Management System**. For the detailed descriptions of the registers related to exception handling, refer to **Section 2.5 Exception Processing**.

Figure 2-7. CP0 Registers

Register No.	Register name	Register No.	Register name
0	Index[Note 1]	16	Config[Note 1]
1	Random[Note 1]	17	LLAddr[Note 1]
2	EntryLo0[Note 1]	18	WatchLo[Note 2]
3	EntryLo1[Note 1]	19	WatchHi[Note 2]
4	Context[Note 2]	20	XContext[Note 2]
5	PageMask[Note 1]	21	RFU
6	Wired[Note 1]	22	RFU
7	RFU	23	RFU
8	BadVAddr[Note 1]	24	RFU
9	Count[Note 2]	25	RFU
10	EntryHi[Note 1]	26	PErr[Note 2]
11	Compare[Note 2]	27	CacheErr[Note 2]
12	Status[Note 2]	28	TagLo[Note 1]
13	Cause[Note 2]	29	TagHi[Note 1]
14	EPC[Note 2]	30	ErrorEPC[Note 2]
15	PRId[Note 1]	31	RFU

Notes 1. for Memory management
2. for Exception handling

Remark RFU: Reserved for future use

Table 2-1. System Control Coprocessor (CP0) Register Definitions

Register Number	Register Name	Description
0	Index	Programmable pointer to TLB array
1	Random	Pseudo-random pointer to TLB array (read only)
2	EntryLo0	Low half of TLB entry for even VPN
3	EntryLo1	Low half of TLB entry for odd VPN
4	Context	Pointer to kernel virtual PTE in 32-bit mode
5	PageMask	TLB page mask
6	Wired	Number of wired TLB entries
7	—	Reserved for future use
8	BadVAddr	Virtual address where the most recent error occurred
9	Count	Timer count
10	EntryHi	High half of TLB entry (including ASID)
11	Compare	Timer compare
12	Status	Status register
13	Cause	Cause of last exception
14	EPC	Exception Program Counter
15	PRId	Processor revision identifier
16	Config	Configuration register (specifying memory mode system)
17	LLAddr	Reserved for future use
18	WatchLo	Memory reference trap address low bits
19	WatchHi	Memory reference trap address high bits
20	XContext	Pointer to kernel virtual PTE in 64-bit mode
21 to 25	—	Reserved for future use
26	PErr Note	Cache parity bits
27	CacheErr Note	Index and status of cache error
28	TagLo	Cache Tag register (low)
29	TagHi	Cache Tag register (high)
30	ErrorEPC	Error Exception Program Counter
31	—	Reserved for future use

Note This register is defined to maintain compatibility with the VR4100™. This register is not used in the μPD98502 hardware.

2.1.6 Floating-point unit (FPU)

The VR4120A does not support the floating-point unit (FPU). Coprocessor Unusable exception will occur if any FPU instructions are executed. If necessary, FPU instructions should be emulated by software in an exception handler.

2.1.7 CPU core memory management system (MMU)

The VR4120A has a 32-bit physical addressing range of 4 Gbytes. However, since it is rare for systems to implement a physical memory space as large as that memory space, the CPU provides a logical expansion of memory space by translating addresses composed in the large virtual address space into available physical memory addresses. The VR4120A supports the following two addressing modes:

- 32-bit mode, in which the virtual address space is divided into 2 Gbytes for user process and 2 Gbytes for the kernel.
- 64-bit mode, in which the virtual address is expanded to1 Tbyte (2^{40} bytes) of user virtual address space.

A detailed description of these address spaces is given in **Section 2.4 Memory Management System**.

2.1.8 Translation lookaside buffer (TLB)

Virtual memory mapping is performed using the translation lookaside buffer (TLB). The TLB converts virtual addresses to physical addresses. It runs by a full-associative method. It has 32 entries, each mapping a pair of pages having a variable size (1 KB to 256 KB).

2.1.8.1 Joint TLB (JTLB)

JTLB holds both an instruction address and data address.

For fast virtual-to-physical address decoding, the VR4120A uses a large, fully associative TLB (joint TLB) that translates 64 virtual pages to their corresponding physical addresses. The TLB is organized as 32 pairs of even-odd entries, and maps a virtual address and address space identifier (ASID) into the 4-Gbyte physical address space.

The page size can be configured, on a per-entry basis, to map a page size of 1 KB to 256 KB. A CP0 register stores the size of the page to be mapped, and that size is entered into the TLB when a new entry is written. Thus, operating systems can provide special purpose maps; for example, a typical frame buffer can be memory-mapped using only one TLB entry.

Translating a virtual address to a physical address begins by comparing the virtual address from the processor with the physical addresses in the TLB; there is a match when the virtual page number (VPN) of the address is the same as the VPN field of the entry, and either the Global (G) bit of the TLB entry is set, or the ASID field of the virtual address is the same as the ASID field of the TLB entry.

This match is referred to as a TLB hit. If there is no match, a TLB Miss exception is taken by the processor and software is allowed to refill the TLB from a page table of virtual/physical addresses in memory.

2.1.9 Operating modes

The VR4120A has three operating modes:

- ✧ User mode
- ✧ Supervisor mode
- ✧ Kernel mode

The manner in which memory addresses are translated or mapped depends on these operating modes. Refer to **Section 2.4 Memory Management System** for details.

2.1.10 Cache

The VR4120A chip incorporates instruction and data caches, which are independent of each other. This configuration enables high-performance pipeline operations. Both caches have a 64-bit data bus, enabling a one-clock access. These buses can be accessed in parallel. The instruction cache of the VR4120A has a storage capacity of 16 KB, while the data cache has a capacity of 8 KB.

A detailed description of caches is given in **Section 2.7 Cache Memory**.

2.1.11 Instruction pipeline

The VR4120A has a 6-stage instruction pipeline. Under normal circumstances, one instruction is issued each cycle.

A detailed description of pipeline is provided in **Section 2.3 Pipeline**.

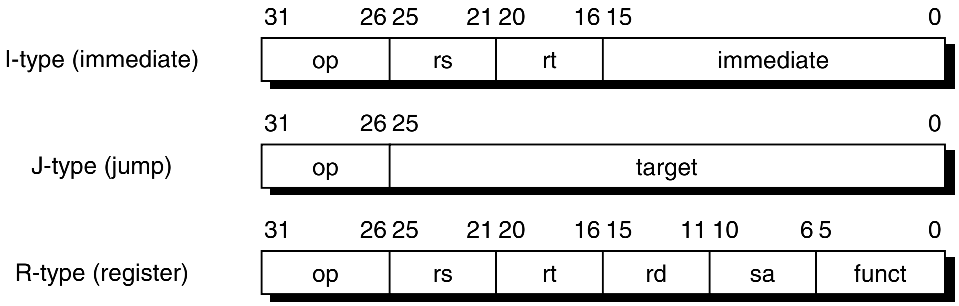
2.2 MIPS III Instruction Set Summary

This section is an overview of the MIPS III ISA central processing unit (CPU) instruction set; refer to **APPENDIX A MIPS III INSTRUCTION SET DETAILS** for detailed descriptions of individual CPU instructions.

2.2.1 MIPS III ISA instruction formats

Each MIPS III ISA CPU instruction consists of a single 32-bit word, aligned on a word boundary. There are three instruction formats - immediate (I-type), jump (J-type), and register (R-type) - as shown in Figure 2-8. The use of a small number of instruction formats simplifies instruction decoding, allowing the compiler to synthesize more complicated and less frequently used instruction and addressing modes from these three formats as needed.

Figure 2-8. MIPS III ISA CPU Instruction Formats



op:	6-bit operation code
rs:	5-bit source register specifier
rt:	5-bit target (source/destination) register specifier or branch condition
immediate:	16-bit immediate value, branch displacement, or address displacement
target:	26-bit unconditional branch target address
rd:	5-bit destination register specifier
sa:	5-bit shift amount
funct:	6-bit function field

2.2.1.1 Support of the MIPS ISA

The VR4120A CORE does not support a multiprocessor operating environment. Thus the synchronization support instructions defined in the MIPS II and MIPS III ISA - the load linked and store conditional instructions - cause reserved instruction exception. The load/link (LL) bit is eliminated.

Caution That the SYNC instruction is handled as a NOP instruction since all load/store instructions in this processor are executed in program order.

2.2.2 Instruction classes

The CPU instructions are classified into five classes.

2.2.2.1 Load and store instructions

Load and store are immediate (I-type) instructions that move data between memory and general registers. The only addressing mode that load and store instructions directly support is base register plus 16-bit signed immediate offset.

(1) Scheduling a load delay slot

A load instruction that does not allow its result to be used by the instruction immediately following is called a delayed load instruction. The instruction slot immediately following this delayed load instruction is referred to as the load delay slot.

In the VR4000 Series™, a load instruction can be followed directly by an instruction that accesses a register that is loaded by the load instruction. In this case, however, an interlock occurs for a necessary number of cycles. Any instruction can follow a load instruction, but the load delay slot should be scheduled appropriately for both performance and compatibility with the VR Series™ microprocessors. For detail, see **Section 2.3 Pipeline**.

(2) Store delay slot

When a store instruction is writing data to a cache, the data cache is kept busy at the DC and WB stages. If an instruction (such as load) that follows directly the store instruction accesses the data cache in the DC stage, a hardware-driven interlock occurs. To overcome this problem, the store delay slot should be scheduled.

Table 2-2. Number of Delay Slot Cycles Necessary for Load and Store Instructions

Instruction	Necessary Number of Cycles
Load	1
Store	1

(3) Defining access types

Access type indicates the size of a VR4120A data item to be loaded or stored, set by the load or store instruction opcode. Access types and accessed byte are shown in Table 2-3.

Regardless of access type or byte ordering (endianness), the address given specifies the low-order byte in the addressed field. For a little-endian configuration, the low-order byte is the least-significant byte.

The access type, together with the low-order three bits of the address, defines the bytes accessed within the addressed doubleword (shown in Table 2-3). Only the combinations shown in Table 2-3 are permissible; other combinations cause address error exceptions.

Tables 2-4 and 2-5 list the ISA-defined load/store instructions and extended-ISA instructions, respectively.

Table 2-3. Byte Specification Related to Load and Store Instructions

Access Type (Value)	Low-Order Address Bit			Accessed Byte							
				Little Endian							
	2	1	0	63							0
Doubleword (7)	0	0	0	7	6	5	4	3	2	1	0
7-byte (6)	0	0	0		6	5	4	3	2	1	0
	0	0	1	7	6	5	4	3	2	1	
6-byte (5)	0	0	0			5	4	3	2	1	0
	0	1	0	7	6	5	4	3	2		
5-byte (4)	0	0	0				4	3	2	1	0
	0	1	1	7	6	5	4	3			
Word (3)	0	0	0					3	2	1	0
	1	0	0	7	6	5	4				
Triple byte (2)	0	0	0						2	1	0
	0	0	1					3	2	1	
	1	0	0		6	5	4				
	1	0	1	7	6	5					
Halfword (1)	0	0	0							1	0
	0	1	0					3	2		
	1	0	0			5	4				
	1	1	0	7	6						
Byte (0)	0	0	0								0
	0	0	1							1	
	0	1	0						2		
	0	1	1					3			
	1	0	0				4				
	1	0	1			5					
	1	1	0		6						
	1	1	1	7							

Table 2-4. Load/Store Instruction

Instruction	Format and Description op base rt offset
Load Byte	LB rt, offset (base) The offset is sign extended and then added to the contents of the register base to form the virtual address. The bytes of the memory location specified by the address are sign extended and loaded into register rt.
Load Byte Unsigned	LBU rt, offset (base) The offset is sign extended and then added to the contents of the register base to form the virtual address. The bytes of the memory location specified by the address are zero extended and loaded into register rt.
Load Halfword	LH rt, offset (base) The offset is sign extended and then added to the contents of the register base to form the virtual address. The halfword of the memory location specified by the address is sign extended and loaded to register rt.
Load Halfword Unsigned	LHU rt, offset (base) The offset is sign extended and then added to the contents of the register base to form the virtual address. The halfword of the memory location specified by the address is zero extended and loaded to register rt.
Load Word	LW rt, offset (base) The offset is sign extended and then added to the contents of the register base to form the virtual address. The word of the memory location specified by the address is sign extended and loaded to register rt. In the 64-bit mode, it is further sign extended to 64 bits.
Load Word Left	LWL rt, offset (base) The offset is sign extended and then added to the contents of the register base to form the virtual address. Shifts to the left the word whose address is specified so that the address-specified byte is at the left-most position of the word. The result of the shift operation is merged with the contents of register rt and loaded to register rt. In the 64-bit mode, it is further sign extended to 64 bits.
Load Word Right	LWR rt, offset (base) The offset is sign extended and then added to the contents of the register base to form the virtual address. Shifts to the right the word whose address is specified so that the address-specified byte is at the right-most position of the word. The result of the shift operation is merged with the contents of register rt and loaded to register rt. In the 64-bit mode, it is further sign extended to 64 bits.
Store Byte	SB rt, offset (base) The offset is sign extended and then added to the contents of the register base to form the virtual address. The least significant byte of register rt is stored to the memory location specified by the address.
Store Halfword	SH rt, offset (base) The offset is sign extended and then added to the contents of the register base to form the virtual address. The least significant halfword of register rt is stored to the memory location specified by the address.
Store Word	SW rt, offset (base) The offset is sign extended and then added to the contents of the register base to form the virtual address. The lower word of register rt is stored to the memory location specified by the address.

Table 2-5. Load/Store Instruction (Extended ISA)

Instruction	Format and Description op base rt offset
Store Word Left	SWL rt, offset (base) The offset is sign extended and then added to the contents of the register base to form the virtual address. Shifts to the right the contents of register rt so that the left-most byte of the word is in the position of the address-specified byte. The result is stored to the lower word in memory.
Store Word Right	SWR rt, offset (base) The offset is sign extended and then added to the contents of the register base to form the virtual address. Shifts to the left the contents of register rt so that the right-most byte of the word is in the position of the address-specified byte. The result is stored to the upper word in memory.
Load Doubleword	LD rt, offset (base) The offset is sign extended and then added to the contents of the register base to form the virtual address. The doubleword of the memory location specified by the address are loaded into register rt.
Load Doubleword Left	LDL rt, offset (base) The offset is sign extended and then added to the contents of the register base to form the virtual address. Shifts to the left the double word whose address is specified so that the address-specified byte is at the left-most position of the double word. The result of the shift operation is merged with the contents of register rt and loaded to register rt.
Load Doubleword Right	LDR rt, offset (base) The offset is sign extended and then added to the contents of the register base to form the virtual address. Shifts to the right the double word whose address is specified so that the address-specified byte is at the right-most position of the double word. The result of the shift operation is merged with the contents of register rt and loaded to register rt.
Load Word Unsigned	LWU rt, offset (base) The offset is sign extended and then added to the contents of the register base to form the virtual address. The word of the memory location specified by the address are zero extended and loaded into register rt.
Store Doubleword	SD rt, offset (base) The offset is sign extended and then added to the contents of the register base to form the virtual address. The contents of register rt are stored to the memory location specified by the address.
Store Doubleword Left	SDL rt, offset (base) The offset is sign extended and then added to the contents of the register base to form the virtual address. Shifts to the right the contents of register rt so that the left-most byte of the double word is in the position of the address-specified byte. The result is stored to the lower doubleword in memory.
Store Doubleword Right	SDR rt, offset (base) The offset is sign extended and then added to the contents of the register base to form the virtual address. Shifts to the left the contents of register rt so that the right-most byte of the double word is in the position of the address-specified byte. The result is stored to the upper doubleword in memory.

2.2.2.2 Computational instructions

Computational instructions perform arithmetic, logical, and shift operations on values in registers. Computational instructions can be either in register (R-type) format, in which both operands are registers, or in immediate (I-type) format, in which one operand is a 16-bit immediate.

Computational instructions are classified as:

(1) ALU immediate instructions

(2) Three-operand type instructions

(3) Shift instructions

(4) Multiply/divide instructions

To maintain data compatibility between the 64- and 32-bit modes, it is necessary to sign-extend 32-bit operands correctly. If the sign extension is not correct, the 32-bit operation result is meaningless.

Table 2-6. ALU Immediate Instruction

Instruction	Format and Description op | rs | rt | immediate
Add Immediate	ADDI rt, rs, immediate The 16-bit immediate is sign extended and then added to the contents of register rs to form a 32-bit result. The result is stored into register rt. In the 64-bit mode, the operand must be sign extended. An exception occurs on the generation of 2's complement overflow.
Add Immediate Unsigned	ADDIU rt, rs, immediate The 16-bit immediate is sign extended and then added to the contents of register rs to form a 32-bit result. The result is stored into register rt. In the 64-bit mode, the operand must be sign extended. No exception occurs on the generation of integer overflow.
Set On Less Than Immediate	SLTI rt, rs, immediate The 16-bit immediate is sign extended and then compared to the contents of register rt treating both operands as signed integers. If rs is less than the immediate, the result is set to 1; otherwise, the result is set to 0. The result is stored to register rt.
Set On Less Than Immediate Unsigned	SLTIU rt, rs, immediate The 16-bit immediate is sign extended and then compared to the contents of register rt treating both operands as unsigned integers. If rs is less than the immediate, the result is set to 1; otherwise, the result is set to 0. The result is stored to register rt.
And Immediate	ANDI rt, rs, immediate The 16-bit immediate is zero extended and then ANDed with the contents of the register. The result is stored into register rt.
Or Immediate	ORI rt, rs, immediate The 16-bit immediate is zero extended and then ORed with the contents of the register. The result is stored into register rt.
Exclusive Or Immediate	XORI rt, rs, immediate The 16-bit immediate is zero extended and then Ex-ORed with the contents of the register. The result is stored into register rt.
Load Upper Immediate	LUI rt, immediate The 16-bit immediate is shifted left by 16 bits to set the lower 16 bits of word to 0. The result is stored into register rt. In the 64-bit mode, the operand must be sign extended.

Table 2-7. ALU Immediate Instruction (Extended ISA)

Instruction	Format and Description op rs rt immediate
Doubleword Add Immediate	DADDI rt, rs, immediate The 16-bit immediate is sign extended to 64 bits and then added to the contents of register rs to form a 64-bit result. The result is stored into register rt. An exception occurs on the generation of integer overflow.
Doubleword Add Immediate Unsigned	DADDIU rt, rs, immediate The 16-bit immediate is sign extended to 64 bits and then added to the contents of register rs to form a 64-bit result. The result is stored into register rt. No exception occurs on the generation of overflow.

Table 2-8. Three-Operand Type Instruction

Instruction	Format and Description op rs rt rd sa funct
Add	ADD rd, rs, rt The contents of registers rs and rt are added together to form a 32-bit result. The result is stored into register rd. In the 64-bit mode, the operand must be sign extended. An exception occurs on the generation of integer overflow.
Add Unsigned	ADDU rd, rs, rt The contents of registers rs and rt are added together to form a 32-bit result. The result is stored into register rd. In the 64-bit mode, the operand must be sign extended. No exception occurs on the generation of integer overflow.
Subtract	SUB rd, rs, rt The contents of register rt are subtracted from the contents of register rs. The 32-bit result is stored into register rd. In the 64-bit mode, the operand must be sign extended. An exception occurs on the generation of integer overflow.
Subtract Unsigned	SUBU rd, rs, rt The contents of register rt are subtracted from the contents of register rs. The 32-bit result is stored into register rd. In the 64-bit mode, the operand must be sign extended. No exception occurs on the generation of integer overflow.
Set On Less Than	SLT rd, rs, rt The contents of registers rs and rt are compared, treating both operands as signed integers. If the contents of register rs is less than that of register rt, the result is set to 1; otherwise, the result is set to 0. The result is stored to register rd.
Set On Less Than Unsigned	SLTU rd, rs, rt The contents of registers rs and rt are compared treating both operands as unsigned integers. If the contents of register rs is less than that of register rt, the result is set to 1; otherwise, the result is set to 0. The result is stored to register rd.
And	AND rd, rt, rs The contents of register rs are logical ANDed with that of general register rt bit-wise. The result is stored to register rd.
Or	OR rd, rt, rs The contents of register rs are logical ORed with that of general register rt bit-wise. The result is stored to register rd.
Exclusive Or	XOR rd, rt, rs The contents of register rs are logical Ex-ORed with that of general register rt bit-wise. The result is stored to register rd.
Nor	NOR rd, rt, rs The contents of register rs are logical NORed with that of general register rt bit-wise. The result is stored to register rd.

Table 2-9. Three-Operand Type Instruction (Extended ISA)

Instruction	Format and Description op rs rt rd sa funct
Doubleword Add	DADD rd, rt, rs The contents of register rs are added to that of register rt. The 64-bit result is stored into register rd. An exception occurs on the generation of integer overflow.
Doubleword Add Unsigned	DADDU rd, rt, rs The contents of register rs are added to that of register rt. The 64-bit result is stored into register rd. No exception occurs on the generation of integer overflow.
Doubleword Subtract	DSUB rd, rt, rs The contents of register rt are subtracted from that of register rs. The 64-bit result is stored into register rd. An exception occurs on the generation of integer overflow.
Doubleword Subtract Unsigned	DSUBU rd, rt, rs The contents of register rt are subtracted from that of register rs. The 64-bit result is stored into register rd. No exception occurs on the generation of integer overflow.

Table 2-10. Shift Instruction

Instruction	Format and Description op rs rt rd sa funct
Shift Left Logical	SLL rd, rs, sa The contents of register rt are shifted left by sa bits and zeros are inserted into the emptied lower bits. The 32-bit result is stored into register rd. In the 64-bit mode, the operand must be sign extended.
Shift Right Logical	SRL rd, rs, sa The contents of register rt are shifted right by sa bits and zeros are inserted into the emptied higher bits. The 32-bit result is stored into register rd. In the 64-bit mode, the operand must be sign extended.
Shift Right Arithmetic	SRA rd, rt, sa The contents of register rt are shifted right by sa bits and the emptied higher bits are sign extended. The 32-bit result is stored into register rd. In the 64-bit mode, the operand must be sign extended.
Shift Left Logical Variable	SLLV rd, rt, rs The contents of register rt are shifted left and zeros are inserted into the emptied lower bits. The lower five bits of register rs specify the shift count. The 32-bit result is stored into register rd. In the 64-bit mode, the operand must be sign extended.
Shift Right Logical Variable	SRLV rd, rt, rs The contents of register rt are shifted right and zeros are inserted into the emptied higher bits. The lower five bits of register rs specify the shift count. The 32-bit result is stored into register rd. In the 64-bit mode, the operand must be sign extended.
Shift Right Arithmetic Variable	SRAV rd, rt, rs The contents of register rt are shifted right and the emptied higher bits are sign extended. The lower five bits of register rs specify the shift count. The 32-bit result is stored into register rd. In the 64-bit mode, the operand must be sign extended.

Table 2-11. Shift Instruction (Extended ISA)

Instruction	Format and Description	op	rs	rt	rd	sa	funct
Doubleword Shift Left Logical	DSLL rd, rt, sa The contents of register rt are shifted left by sa bits and zeros are inserted into the emptied lower bits. The 64-bit result is stored into register rd.						
Doubleword Shift Right Logical	DSRL rd, rt, sa The contents of register rt are shifted right by sa bits and zeros are inserted into the emptied higher bits. The 64-bit result is stored into register rd.						
Doubleword Shift Right Arithmetic	DSRA rd, rt, sa The contents of register rt are shifted right by sa bits and the emptied higher bits are sign extended. The 64-bit result is stored into register rd.						
Doubleword Shift Left Logical Variable	DSLLV rd, rt, rs The contents of register rt are shifted left and zeros are inserted into the emptied lower bits. The lower six bits of register rs specify the shift count. The 64-bit result is stored into register rd.						
Doubleword Shift Right Logical Variable	DSRLV rd, rt, rs The contents of register rt are shifted right and zeros are inserted into the emptied higher bits. The lower six bits of register rs specify the shift count. The 64-bit result is stored into register rd.						
Doubleword Shift Right Arithmetic Variable	DSRAV rd, rt, rs The contents of register rt are shifted right and the emptied higher bits are sign extended. The lower six bits of register rs specify the shift count. The 64-bit result is stored into register rd.						
Doubleword Shift Left Logical + 32	DSLL32 rd, rt, sa The contents of register rt are shifted left by 32 + sa bits and zeros are inserted into the emptied lower bits. The 64-bit result is stored into register rd.						
Doubleword Shift Right Logical + 32	DSRL32 rd, rt, sa The contents of register rt are shifted right by 32 + sa bits and zeros are inserted into the emptied higher bits. The 64-bit result is stored into register rd.						
Doubleword Shift Right Arithmetic + 32	DSRA32 rd, rt, sa The contents of register rt are shifted right by 32 + sa bits and the emptied higher bits are sign extended. The 64-bit result is stored into register rd.						

Table 2-12. Multiply/Divide Instructions

Instruction	Format and Description op rs rt rd sa funct
Multiply	MULT rs, rt The contents of registers rt and rs are multiplied, treating both operands as 32-bit signed integers. The 64-bit result is stored into special registers HI and LO. In the 64-bit mode, the operand must be sign extended.
Multiply Unsigned	MULTU rs, rt The contents of registers rt and rs are multiplied, treating both operands as 32-bit unsigned integers. The 64-bit result is stored into special registers HI and LO. In the 64-bit mode, the operand must be sign extended.
Divide	DIV rs, rt The contents of register rs are divided by that of register rt, treating both operands as 32-bit signed integers. The 32-bit quotient is stored into special register LO, and the 32-bit remainder is stored into special register HI. In the 64-bit mode, the operand must be sign extended.
Divide Unsigned	DIVU rs, rt The contents of register rs are divided by that of register rt, treating both operands as 32-bit unsigned integers. The 32-bit quotient is stored into special register LO, and the 32-bit remainder is stored into special register HI. In the 64-bit mode, the operand must be sign extended.
Move From HI	MFHI rd The contents of special register HI are loaded into register rd.
Move From LO	MFLO rd The contents of special register LO are loaded into register rd.
Move To HI	MTHI rs The contents of register rs are loaded into special register HI.
Move To LO	MTLO rs The contents of register rs are loaded into special register LO.

Table 2-13. Multiply/Divide Instructions (Extended ISA)

Instruction	Format and Description | op | rs | rt | rd | sa | funct
Doubleword Multiply	DMULT rs, rt The contents of registers rt and rs are multiplied, treating both operands as signed integers. The 128-bit result is stored into special registers HI and LO.
Doubleword Multiply Unsigned	DMULTU rs, rt The contents of registers rt and rs are multiplied, treating both operands as unsigned integers. The 128-bit result is stored into special registers HI and LO.
Doubleword Divide	DDIV rs, rt The contents of register rs are divided by that of register rt, treating both operands as signed integers. The 64-bit quotient is stored into special register LO, and the 64-bit remainder is stored into special register HI.
Doubleword Divide Unsigned	DDIVU rs, rt The contents of register rs are divided by that of register rt, treating both operands as unsigned integers. The 64-bit quotient is stored into special register LO, and the 64-bit remainder is stored into special register HI.
Multiply and Add Accumulate	MACC{h}{u}{s} rd, rs, rt The contents of registers rt and rs are multiplied, treating both operands as 32-bit signed integers. The result is added to the combined value of special registers HI and LO. The 64-bit result is stored into special registers HI and LO. If h=0, the same data as that stored in register LO is also stored in register rd; if h=1, the same data as that stored in register HI is also stored in register rd. If u is specified, the operand is treated as unsigned data. If s is specified, registers rs and rd are treated as a 16-bit value (32 bits sign- or zero-extended), and the value obtained by combining registers HI and LO is treated as a 32-bit value (64 bits sign- or zero-extended). Moreover, saturation processing is performed for the operation result in the format specified with u.
Doubleword Multiply and Add Accumulate	DMACC{h}{u}{s} rd, rs, rt The contents of registers rt and rs are multiplied, treating both operands as 32-bit signed integers. The result is added to value of special register LO. The 64-bit result is stored into special register LO. If h=0, the same data as that stored in register LO is also stored in register rd; if h=1, undefined data is stored in register rd. If u is specified, the operand is treated as unsigned data. If s is specified, registers rs and rd are treated as a 16-bit value (32 bits sign- or zero-extended), and register LO is treated as a 32-bit value (64 bits sign- or zero-extended). Moreover, saturation processing is performed for the operation result in the format specified with u.

MFHI and MFLO instructions after a multiply or divide instruction generate interlocks to delay execution of the next instruction, inhibiting the result from being read until the multiply or divide instruction completes.

Table 2-14 gives the number of processor cycles (PCycles) required to resolve interlock or stall between various multiply or divide instructions and a subsequent MFHI or MFLO instruction.

Table 2-14. Number of Stall Cycles in Multiply and Divide Instructions

Instruction	Number of Instruction Cycles
MULT	1
MULTU	1
DIV	36
DIVU	36
DMULT	3
DMULTU	3
DDIV	68
DDIVU	68
MACC	0
DMACC	0

2.2.2.3 Jump and branch instructions

Jump and branch instructions change the control flow of a program. All jump and branch instructions occur with a delay of one instruction: that is, the instruction immediately following the jump or branch instruction (this is known as the instruction in the delay slot) always executes while the target instruction is being fetched from memory.

For instructions involving a link (such as JAL and BLTZAL), the return address is saved in register r31.

Table 2-15. Number of Delay Slot Cycles in Jump and Branch Instructions

Instruction	Necessary Number of Cycles
Branch instruction	1
Jump instruction	1

(1) Overview of jump instructions

Subroutine calls in high-level languages are usually implemented with J or JAL instructions, both of which are J-type instructions. In J-type format, the 26-bit target address shifts left 2 bits and combines with the high-order 4 bits of the current program counter to form a 32-bit or 64-bit absolute address.

Returns, dispatches, and cross-page jumps are usually implemented with the JR or JALR instructions. Both are R-type instructions that take the 32-bit or 64-bit byte address contained in one of the general registers.

For more information, refer to **APPENDIX A MIPS III INSTRUCTION SET DETAILS**.

(2) Overview of branch instructions

A branch instruction has a PC-related signed 16-bit offset.

Tables 2-16 through 2-18 show the lists of Jump, Branch, and Expanded ISA instructions, respectively.

Table 2-16. Jump Instruction

Instruction	Format and Description op | target
Jump	JAL target The contents of 26-bit target address is shifted left by two bits and combined with the high-order four bits of the PC. The program jumps to this calculated address with a delay of one instruction.
Jump And Link	J target The contents of 26-bit target address is shifted left by two bits and combined with the high-order four bits of the PC. The program jumps to this calculated address with a delay of one instruction. The address of the instruction following the delay slot is stored into r31 (link register).

Instruction	Format and Description op | target
Jump And Link Exchange	JALX target The contents of 26-bit target address is shifted left by two bits and combined with the high-order four bits of the PC. The program jumps to this calculated address with a delay of one instruction, and then the ISA mode bit is reversed. The address of the instruction following the delay slot is stored into r31 (link register).

Instruction	Format and Description op | rs | rt | rd | sa | funct
Jump Register	JR rs The program jumps to the address specified in register rs with a delay of one instruction.
Jump And Link Register	JALR rs, rd The program jumps to the address specified in register rs with a delay of one instruction. The address of the instruction following the delay slot is stored into rd.

There are the following common restrictions for Tables 2-17 and 2-18.

(3) Branch address

All branch instruction target addresses are computed by adding the address of the instruction in the delay slot to the 16-bit offset (shifted left by 2 bits and sign-extended to 64 bits). All branches occur with a delay of one instruction.

(4) Operation when unbranched (Table 2-18)

If the branch condition does not meet in executing a likely instruction, the instruction in its delay slot is nullified.

For all other branch instructions, the instruction in its delay slot is unconditionally executed.

Remark The target instruction of the branch is fetched at the EX stage of the branch instruction. Comparison of the operands of the branch instruction and calculation of the target address is performed at phase 2 of the RF stage and phase 1 of the EX stage of the instruction. Branch instructions require one cycle of the branch delay slot defined by the architecture. Jump instructions also require one cycle of delay slot. If the branch condition is not satisfied in a branch likely instruction, the instruction in its delay slot is nullified.

There are special symbols used in the instruction formats of Tables 2-17 through 2-21.

REGIMM	: Opcode
Sub	: Sub-operation code
CO	: Sub-operation identifier
BC	: BC sub-operation code
br	: Branch condition identifier
op	: Operation code

Table 2-17. Branch Instructions

Instruction	Format and Description	op	rs	rt	offset
Branch On Equal	BEQ rs, rt, offset If the contents of register rs are equal to that of register rt, the program branches to the target address.				
Branch On Not Equal	BNE rs, rt, offset If the contents of register rs are not equal to that of register rt, the program branches to the target address.				
Branch On Less Than Or Equal To Zero	BLEZ rs, offset If the contents of register rs are less than or equal to zero, the program branches to the target address.				
Branch On Greater Than Zero	BGTZ rs, offset If the contents of register rs are greater than zero, the program branches to the target address.				

Instruction	Format and Description	REGIMM	rs	sub	offset
Branch On Less Than Zero	BLTZ rs, offset If the contents of register rs are less than zero, the program branches to the target address.				
Branch On Greater Than Or Equal To Zero	BGEZ rs, offset If the contents of register rs are greater than or equal to zero, the program branches to the target address.				
Branch On Less Than Zero And Link	BLTZAL rs, offset The address of the instruction that follows delay slot is stored to register r31 (link register). If the contents of register rs are less than zero, the program branches to the target address.				
Branch On Greater Than Or Equal To Zero And Link	BGEZAL rs, offset The address of the instruction that follows delay slot is stored to register r31 (link register). If the contents of register rs are greater than or equal to zero, the program branches to the target address.				

Instruction	Format and Description	COP0	BC	br	offset
Branch On Coprocessor 0 True	BC0T offset Adds the 16-bit offset (shifted left by two bits and sign extended to 32 bits) to the address of the instruction in the delay slot to calculate the branch target address. If the conditional signal of the coprocessor 0 is true, the program branches to the target address with one-instruction delay.				
Branch On Coprocessor 0 False	BC0F offset Adds the 16-bit offset (shifted left by two bits and sign extended to 32 bits) to the address of the instruction in the delay slot to calculate the branch target address. If the conditional signal of the coprocessor 0 is false, the program branches to the target address with one-instruction delay.				

Table 2-18. Branch Instructions (Extended ISA)

Instruction	Format and Description op rs rt offset
Branch On Equal Likely	BEQL rs, rt, offset If the contents of register rs are equal to that of register rt, the program branches to the target address. If the branch condition is not met, the instruction in the delay slot is discarded.
Branch On Not Equal Likely	BNEL rs, rt, offset If the contents of register rs are not equal to that of register rt, the program branches to the target address. If the branch condition is not met, the instruction in the delay slot is discarded.
Branch On Less Than Or Equal To Zero Likely	BLEZL rs, offset If the contents of register rs are less than or equal to zero, the program branches to the target address. If the branch condition is not met, the instruction in the delay slot is discarded.
Branch On Greater Than Zero Likely	BGTZL rs, offset If the contents of register rs are greater than zero, the program branches to the target address. If the branch condition is not met, the instruction in the delay slot is discarded.

Instruction	Format and Description REGIMM rs sub offset
Branch On Less Than Zero Likely	BLTZL rs, offset If the contents of register rs are less than zero, the program branches to the target address. If the branch condition is not met, the instruction in the delay slot is discarded.
Branch On Greater Than Or Equal To Zero Likely	BGEZL rs, offset If the contents of register rs are greater than or equal to zero, the program branches to the target address. If the branch condition is not met, the instruction in the delay slot is discarded.
Branch On Less Than Zero And Link Likely	BLTZALL rs, offset The address of the instruction that follows delay slot is stored to register r31 (link register). If the contents of register rs are less than zero, the program branches to the target address. If the branch condition is not met, the instruction in the delay slot is discarded.
Branch On Greater Than Or Equal To Zero And Link Likely	BGEZALL rs, offset The address of the instruction that follows delay slot is stored to register r31 (link register). If the contents of register rs are greater than or equal to zero, the program branches to the target address. If the branch condition is not met, the instruction in the delay slot is discarded.

Instruction	Format and Description COP0 BC br offset
Branch On Coprocessor 0 True Likely	BC0TL offset Adds the 16-bit offset (shifted left by two bits and sign extended to 32 bits) to the address of the instruction in the delay slot to calculate the branch target address. If the conditional signal of the coprocessor 0 is true, the program branches to the target address with one-instruction delay. If the branch condition is not met, the instruction in the delay slot is discarded.
Branch On Coprocessor 0 False Likely	BC0FL offset Adds the 16-bit offset (shifted left by two bits and sign extended to 32 bits) to the address of the instruction in the delay slot to calculate the branch target address. If the conditional signal of the coprocessor 0 is false, the program branches to the target address with one-instruction delay. If the branch condition is not met, the instruction in the delay slot is discarded.

2.2.2.4 Special instructions

Special instructions generate software exceptions. Their formats are R-type (Syscall, Break). The Trap instruction is available only for the VR4000 Series. All the other instructions are available for all VR Series.

Table 2-19. Special Instructions

Instruction	Format and Description | SPECIAL | rs | rt | rd | sa | funct |
Synchronize	SYNC Completes the load/store instruction executing in the current pipeline before the next load/store instruction starts execution.
System Call	SYSCALL Generates a system call exception, and then transits control to the exception handling program.
Breakpoint	BREAK Generates a break point exception, and then transits control to the exception handling program.

Table 2-20. Special Instructions (Extended ISA) (1/2)

Instruction	Format and Description | SPECIAL | rs | rt | rd | sa | funct |
Trap If Greater Than Or Equal	TGE rs, rt The contents of register rs are compared with that of register rt, treating both operands as signed integers. If the contents of register rs are greater than or equal to that of register rt, an exception occurs.
Trap If Greater Than Or Equal Unsigned	TGEU rs, rt The contents of register rs are compared with that of register rt, treating both operands as unsigned integers. If the contents of register rs are greater than or equal to that of register rt, an exception occurs.
Trap If Less Than	TLT rs, rt The contents of register rs are compared with that of register rt, treating both operands as signed integers. If the contents of register rs are less than that of register rt, an exception occurs.
Trap If Less Than Unsigned	TLTU rs, rt The contents of register rs are compared with that of register rt, treating both operands as unsigned integers. If the contents of register rs are less than that of register rt, an exception occurs.
Trap If Equal	TEQ rs, rt If the contents of registers rs and rt are equal, an exception occurs.
Trap If Not Equal	TNE rs, rt If the contents of registers rs and rt are not equal, an exception occurs.

Table 2-20. Special Instructions (Extended ISA) (2/2)

Instruction	Format and Description REGIMM rs sub immediate
Trap If Greater Than Or Equal Immediate	TGEI rs, immediate The contents of register rs are compared with 16-bit sign-extended immediate data, treating both operands as signed integers. If the contents of register rs are greater than or equal to 16-bit sign-extended immediate data, an exception occurs.
Trap If Greater Than Or Equal Immediate Unsigned	TGEIU rs, immediate The contents of register rs are compared with 16-bit zero-extended immediate data, treating both operands as unsigned integers. If the contents of register rs are greater than or equal to 16-bit sign-extended immediate data, an exception occurs.
Trap If Less Than Immediate	TLTI rs, immediate The contents of register rs are compared with 16-bit sign-extended immediate data, treating both operands as signed integers. If the contents of register rs are less than 16-bit sign-extended immediate data, an exception occurs.
Trap If Less Than Immediate Unsigned	TLTIU rs, immediate The contents of register rs are compared with 16-bit zero-extended immediate data, treating both operands as unsigned integers. If the contents of register rs are less than 16-bit sign-extended immediate data, an exception occurs.
Trap If Equal Immediate	TEQI rs, immediate If the contents of register rs and immediate data are equal, an exception occurs.
Trap If Not Equal Immediate	TNEI rs, immediate If the contents of register rs and immediate data are not equal, an exception occurs.

2.2.2.5 System control coprocessor (CP0) instructions

System control coprocessor (CP0) instructions perform operations specifically on the CP0 registers to manipulate the memory management and exception handling facilities of the processor.

Table 2-21. System Control Coprocessor (CP0) Instructions (1/2)

Instruction	Format and Description COP0 sub rt rd 0
Move To System Control Coprocessor	MTC0 rt, rd The word data of general register rt in the CPU are loaded into general register rd in the CP0.
Move From System Control Coprocessor	MFC0 rt, rd The word data of general register rd in the CP0 are loaded into general register rt in the CPU.
Doubleword Move To System Control Coprocessor 0	DMTC0 rt, rd The doubleword data of general register rt in the CPU are loaded into general register rd in the CP0.
Doubleword Move From System Control Coprocessor 0	DMFC0 rt, rd The doubleword data of general register rd in the CP0 are loaded into general register rt in the CPU.

Table 2-21. System Control Coprocessor (CP0) Instructions (2/2)

Instruction	Format and Description	COP0	CO	funct
Read Indexed TLB Entry	TLBR The TLB entry indexed by the index register is loaded into the entryHi, entryLo0, entryLo1, or page mask register.			
Write Indexed TLB Entry	TLBWI The contents of the entryHi, entryLo0, entryLo1, or page mask register are loaded into the TLB entry indexed by the index register.			
Write Random TLB Entry	TLBWR The contents of the entryHi, entryLo0, entryLo1, or page mask register are loaded into the TLB entry indexed by the random register.			
Probe TLB For Matching Entry	TLBP The address of the TLB entry that matches with the contents of entryHi register is loaded into the index register.			
Return From Exception	ERET The program returns from exception, interrupt, or error trap.			

Instruction	Format and Description	COP0	CO	funct
STANDBY	STANDBY The processor's operating mode is transited from fullspeed mode to standby mode.			
SUSPEND	SUSPEND The processor's operating mode is transited from fullspeed mode to suspend mode.			
HIBERNATE	HIBERNATE The processor's operating mode is transited from fullspeed mode to hibernate mode.			

Instruction	Format and Description	CACHE	base	op	offset
Cache Operation	Cache op, offset (base) The 16-bit offset is sign extended to 32 bits and added to the contents of the register case, to form virtual address. This virtual address is translated to physical address with TLB. For this physical address, cache operation that is indicated by 5-bit sub-opcode is performed.				

2.3 Pipeline

This section describes the basic operation of the VR4120A Core pipeline, which includes descriptions of the delay slots (instructions that follow a branch or load instruction in the pipeline), interrupts to the pipeline flow caused by interlocks and exceptions, and CP0 hazards.

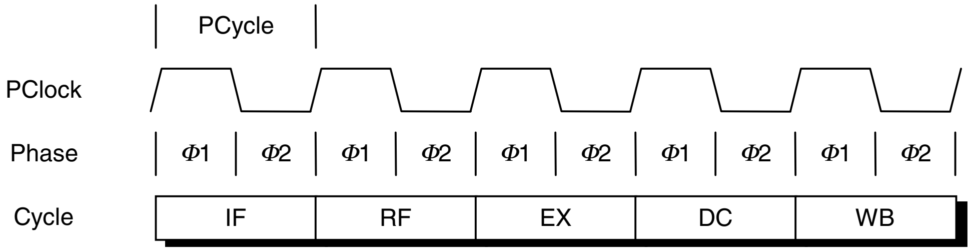
2.3.1 Pipeline stages

The pipeline is controlled by PClock(one cycle of PClock which runs at 4-times frequency of MasterClock) and one cycle of this PClock is called PCycle. Each pipeline stage takes one PCycle.

2.3.1.1 Pipeline in MIPS III instruction mode

The VR4120A has a five-stage instruction pipeline; each stage takes one PCycle, and each PCycle has two phases: *Φ*1 and *Φ*2, as shown in Figure 2-9. Thus, the execution of each instruction takes at least 5 PCycles. An instruction can take longer - for example, if the required data is not in the cache, the data must be retrieved from main memory.

Figure 2-9. Pipeline Stages (MIPS III Instruction Mode)

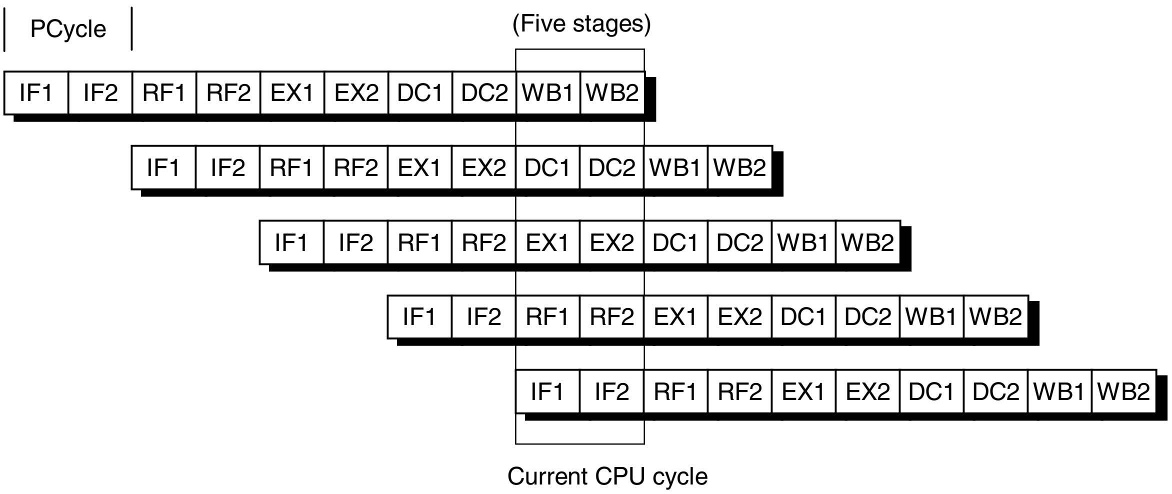


The five pipeline stages are:

- ✧ IF - Instruction cache fetch
- ✧ RF - Register fetch
- ✧ EX - Execution
- ✧ DC - Data cache fetch
- ✧ WB - Write back

Figure 2-10 shows the five stages of the instruction pipeline. In this figure, a row indicates the execution process of each instruction, and a column indicates the processes executed simultaneously.

Figure 2-10. Instruction Execution in the Pipeline


PCycle
(Five stages)
IF1 IF2 RF1 RF2 EX1 EX2 DC1 DC2 WB1 WB2
IF1 IF2 RF1 RF2 EX1 EX2 DC1 DC2 WB1 WB2
IF1 IF2 RF1 RF2 EX1 EX2 DC1 DC2 WB1 WB2
IF1 IF2 RF1 RF2 EX1 EX2 DC1 DC2 WB1 WB2
IF1 IF2 RF1 RF2 EX1 EX2 DC1 DC2 WB1 WB2
Current CPU cycle

2.3.1.2 Pipeline activities

(1) MIPS III instruction

Figure 2-11 shows the activities that can occur during each pipeline stage in MIPS III Instruction mode. Table 2-22 describes these pipeline activities.

Figure 2-11. Pipeline Activities (MIPS III)

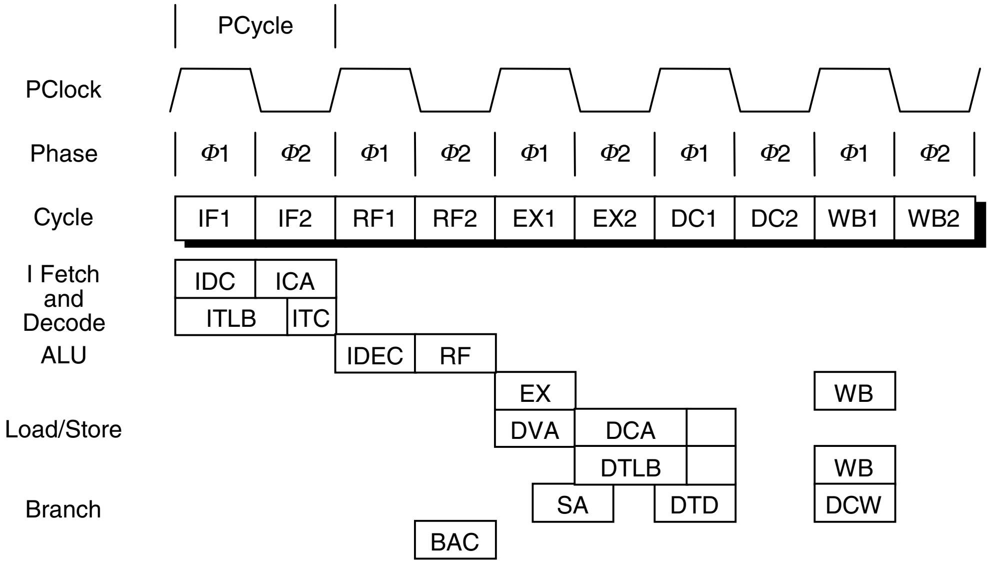


Table 2-22. Operation in Each Stage of Pipeline (MIPS III)

Cycle	Phase	Mnemonic	Description
IF	*Φ*1	IDC	Instruction cache address decode
		ITLB	Instruction address translation
	*Φ*2	ICA	Instruction cache array access
		ITC	Instruction tag check
RF	*Φ*1	IDEC	Instruction decode
	*Φ*2	RF	Register operand fetch
		BAC	Branch address calculation
EX	*Φ*1	EX	Execution stage
		DVA	Data virtual address calculation
		SA	Store align
	*Φ*2	DCA	Data cache address decode/array access
		DTLB	Data address translation
DC	*Φ*1	DLA	Data cache load align
		DTC	Data tag check
		DTD	Data transfer to data cache
WB	*Φ*1	DCW	Data cache write
		WB	Write back to register file

2.3.2 Branch delay

During a VR4120A's pipeline operation, a one-cycle branch delay occurs when:

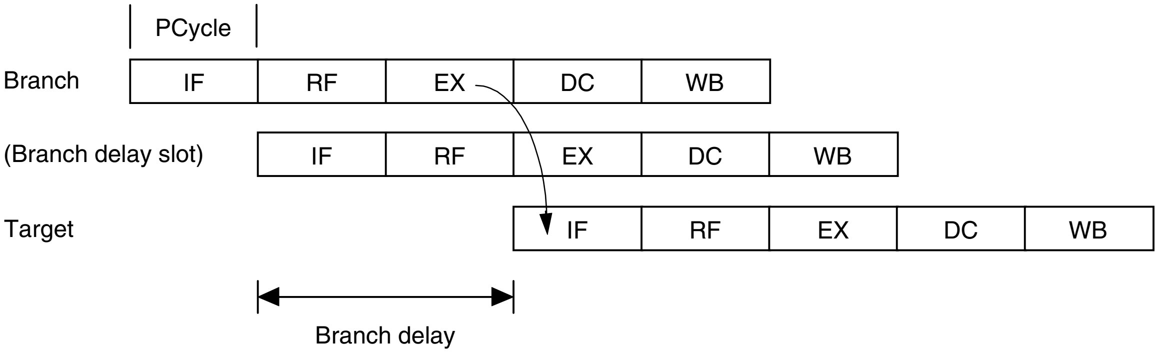
- Target address is calculated by a Jump instruction
- Branch condition of branch instruction is met and then logical operation starts for branch-destination comparison

The instruction location following the Jump/Branch instruction is called a branch delay slot.

The instruction address generated at the EX stage in the Jump/Branch instruction are available in the IF stage, two instructions later. In MIPS III instruction mode, branch delay is two cycles. One instruction in the branch delay slot is executed, except for likely instruction.

Figure 2-12 illustrates the branch delay and the location of the branch delay slot during MIPS III instruction mode.

Figure 2-12. Branch Delay (In MIPS III Instruction Mode)



2.3.3 Load delay

In the case of a load instruction, 2 cycles are required for the DC stage, for reading from the data cache and performing data alignment. In this case, the hardware automatically generates on interlock.

A load instruction that does not allow its result to be used by the instruction immediately following is called a delayed load instruction. The instruction immediately following this delayed load instruction is referred to as the load delay slot.

In the VR4120A, the instruction immediately following a load instruction can use the contents of the loaded register, however in such cases hardware interlocks insert additional delay cycles. Consequently, scheduling load delay slots can be desirable, both for performance and VR-Series processor compatibility.

2.3.4 Pipeline operation

The operation of the pipeline is illustrated by the following examples that describe how typical instructions are executed. The instructions described are six: ADD, JALR, BEQ, TLT, LW, and SW. Each instruction is taken through the pipeline and the operations that occur in each relevant stage are described.

2.3.4.1 Add instruction (ADD rd, rs, rt)

IF stage In *Φ*1 of the IF stage, the eleven least-significant bits of the virtual access are used to access the instruction cache. In *Φ*2 of the IF stage, the cache index is compared with the page frame number and the cache data is read out. The virtual PC is incremented by 4 so that the next instruction can be fetched.

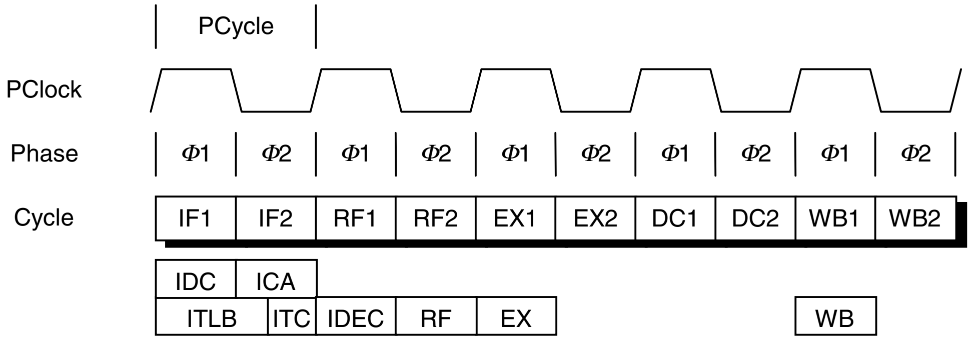
RF stage During *Φ*2, the 2-port register file is addressed with the rs and rt fields and the register data is valid at the register file output. At the same time, bypass multiplexers select inputs from either the EX- or DC-stage output in addition to the register file output, depending on the need for an operand bypass.

EX stage The ALU controls are set to do an A + B operation. The operands flow into the ALU inputs, and the ALU operation is started. The result of the ALU operation is latched into the ALU output latch during *Φ*1.

DC stage This stage is a NOP for this instruction. The data from the output of the EX stage (the ALU) is moved into the output latch of the DC.

WB stage During *Φ*1, the WB latch feeds the data to the inputs of the register file, which is addressed by the rd field. The file write strobe is enabled. By the end of *Φ*1, the data is written into the file.

Figure 2-13. ADD Instruction Pipeline Activities (In MIPS III Instruction Mode)



2.3.4.2 Jump and link register instruction (JALR rd, rs)

IF stage Same as the IF stage for the ADD instruction.

IT stage Same as the IT stage for the ADD instruction.

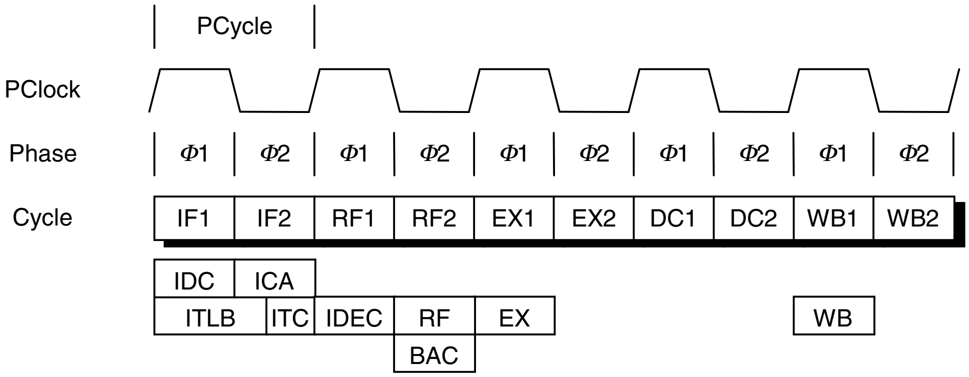
RF stage A register specified in the rs field is read from the file during Φ2 at the RF stage, and the value read from the rs register is input to the virtual PC latch synchronously. This value is used to fetch an instruction at the jump destination. The value of the virtual PC incremented during the IF stage is incremented again to produce the link address PC + 8 where PC is the address of the JALR instruction. The resulting value is the PC to which the program will eventually return. This value is placed in the Link output latch of the Instruction Address unit.

EX stage The PC + 8 value is moved from the Link output latch to the output latch of the EX stage.

DC stage The PC + 8 value is moved from the output latch of the EX stage to the output latch of the DC stage.

WB stage Refer to the ADD instruction. Note that if no value is explicitly provided for rd then register 31 is used as the default. If rd is explicitly specified, it cannot be the same register addressed by rs; if it is, the result of executing such an instruction is undefined.

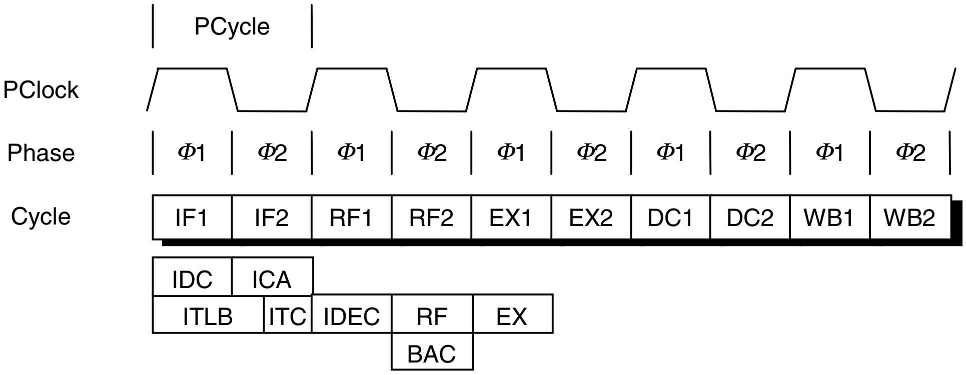
Figure 2-14. JALR Instruction Pipeline Activities (In MIPS III Instruction Mode)



2.3.4.3 Branch on equal instruction (BEQ rs, rt, offset)

IF stage	Same as the IF stage for the ADD instruction.
IT stage	Same as the IT stage for the ADD instruction.
RF stage	During Φ2, the register file is addressed with the rs and rt fields. A check is performed to determine if each corresponding bit position of these two operands has equal values. If they are equal, the PC is set to PC + target, where target is the sign-extended offset field. If they are not equal, the PC is set to PC + 4.
EX stage	The next PC resulting from the branch comparison is valid at the beginning of Φ2 for instruction fetch.
DC stage	This stage is a NOP for this instruction.
WB stage	This stage is a NOP for this instruction.

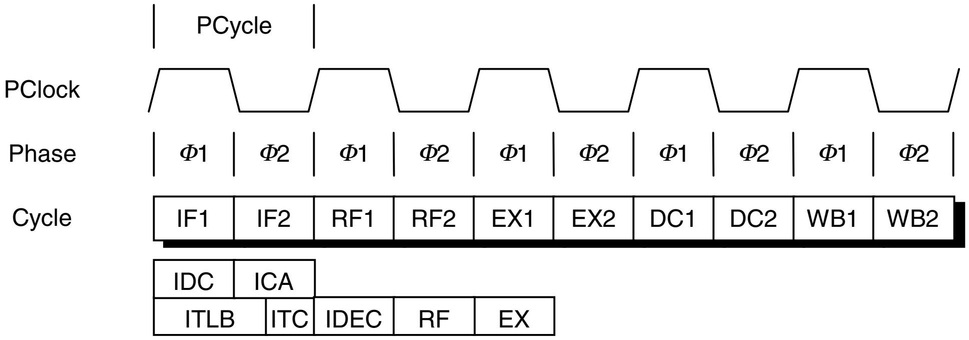
Figure 2-15. BEQ Instruction Pipeline Activities (In MIPS III Instruction Mode)



2.3.4.4 Trap if less than instruction (TLT rs, rt)

IF stage	Same as the IF stage for the ADD instruction.
RF stage	Same as the RF stage for the ADD instruction.
EX stage	ALU controls are set to do an A - B operation. The operands flow into the ALU inputs, and the ALU operation is started. The result of the ALU operation is latched into the ALU output latch during *Φ*1. The sign bits of operands and of the ALU output latch are checked to determine if a less than condition is true. If this condition is true, a Trap exception occurs. The value in the PC register is used as an exception vector value, and from now on any instruction will be invalid.
DC stage	No operation
WB stage	The value of the PC is loaded to EPC register if the less than condition was met in the EX stage. The Cause register ExCode field and BD bit are updated appropriately, as is the EXL bit of the Status register. If the less than condition was not met in the EX stage, no activity occurs in the WB stage.

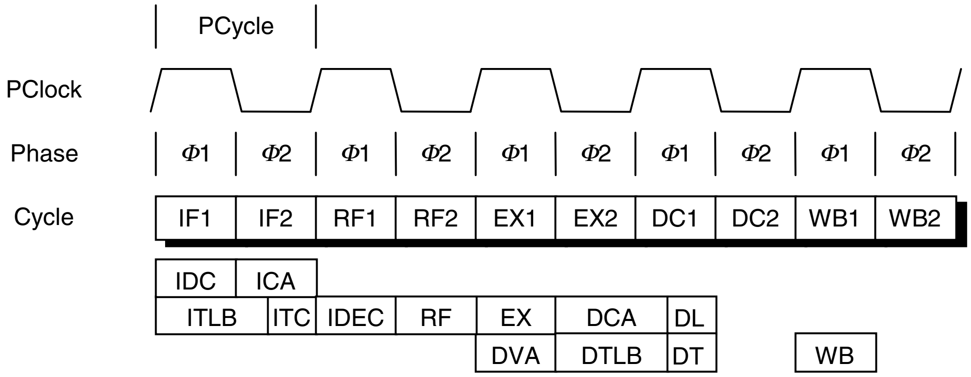
Figure 2-16. TLT Instruction Pipeline Activities



2.3.4.5 Load word instruction (LW rt, offset (base))

IF stage	Same as the IF stage for the ADD instruction.
IT stage	Same as the IT stage for the ADD instruction.
RF stage	Same as the RF stage for the ADD instruction. Note that the base field is in the same position as the rs field.
EX stage	Refer to the EX stage for the ADD instruction. For LW, the inputs to the ALU come from GPR[base] through the bypass multiplexer and from the sign-extended offset field. The result of the ALU operation that is latched into the ALU output latch in Φ1 represents the effective virtual address of the operand (DVA).
DC stage	The cache tag field is compared with the Page Frame Number (PFN) field of the TLB entry. After passing through the load aligner, aligned data is placed in the DC output latch during Φ2.
WB stage	During Φ1, the cache read data is written into the register file addressed by the rt field.

Figure 2-17. LW Instruction Pipeline Activities (In MIPS III Instruction Mode)



2.3.4.6 Store word instruction (SW rt, offset (base))

IF stage Same as the IF stage for the ADD instruction.

IT stage Same as the IT stage for the ADD instruction.

RF stage Same as the RF stage for the LW instruction.

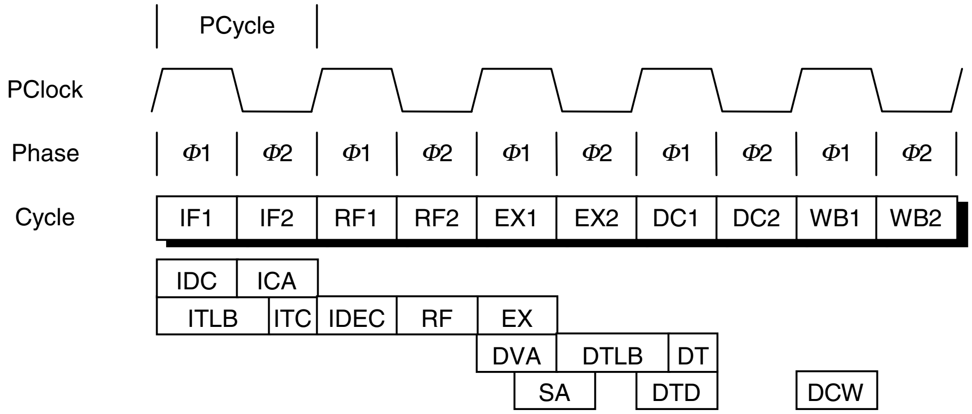
EX stage Refer to the LW instruction for a calculation of the effective address. From the RF output latch, the GPR[rt] is sent through the bypass multiplexer and into the main shifter, where the shifter performs the byte-alignment operation for the operand. The results of the ALU are latched in the output latches during *Φ*1. The shift operations are latched in the output latches during *Φ*2.

DC stage Refer to the LW instruction for a description of the cache access.

WB stage If there was a cache hit, the content of the store data output latch is written into the data cache at the appropriate word location.

Note that all store instructions use the data cache for two consecutive PCycles. If the following instruction requires use of the data cache, the pipeline is slipped for one PCycle to complete the writing of an aligned store data.

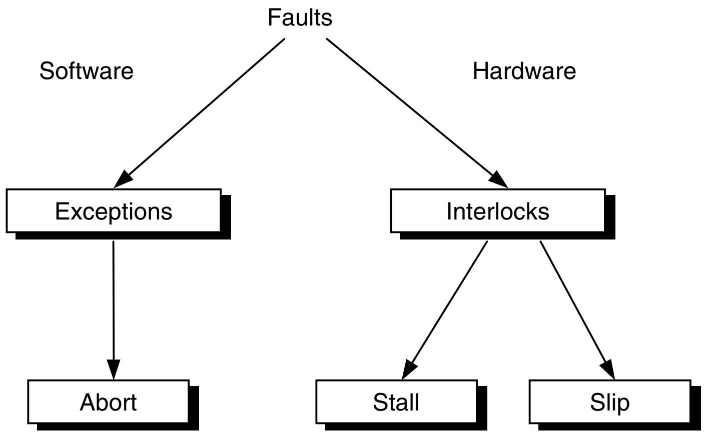
Figure 2-18. SW Instruction Pipeline Activities (In MIPS III Instruction Mode)



2.3.5 Interlock and exception handling

Smooth pipeline flow is interrupted when cache misses or exceptions occur, or when data dependencies are detected. Interruptions handled using hardware, such as cache misses, are referred to as interlocks, while those that are handled using software are called exceptions. As shown in Figure 2-19, all interlock and exception conditions are collectively referred to as faults.

Figure 2-19. Relationship among Interlocks, Exceptions, and Faults



At each cycle, exception and interlock conditions are checked for all active instructions.

Because each exception or interlock condition corresponds to a particular pipeline stage, a condition can be traced back to the particular instruction in the exception/interlock stage, as shown in Table 2-23. For instance, an LDI Interlock is raised in the Register Fetch (RF) stage.

Tables 2-24 and 2-25 describe the pipeline interlocks and exceptions listed in Table 2-23.

Table 2-23. Correspondence of Pipeline Stage to Interlock and Exception Conditions

Stage / Status		IF	RF (IT)	EX	DC	WB
Interlock	Stall	–	ITM ICM	–	DTM DCM DCB	–
	Slip	–	LDI MDI SLI CP0	–	–	–
Exception		IAErr	NMI ITLB IPErr INTr IBE SYSC BP CUn RSVD	Trap OVF DAErr	Reset DTLB TMod DPErr WAT DBE	–

Remark In the above table, exception conditions are listed up in higher priority order.

Table 2-24. Pipeline Interlock

Interlock	Description
ITM	Instruction TLB Miss
ICM	Instruction Cache Miss
LDI	Load Data Interlock
MDI	MD Busy Interlock
SLI	Store-Load Interlock
CP0	Coprocessor 0 Interlock
DTM	Data TLB Miss
DCM	Data Cache Miss
DCB	Data Cache Busy

Table 2-25. Description of Pipeline Exception

Exception	Description
IAErr	Instruction Address Error exception
NMI	Non-maskable Interrupt exception
ITLB	ITLB exception
IPErr	Instruction Parity Error exception
INTr	Interrupt exception
IBE	Instruction Bus Error exception
SYSC	System Call exception
BP	Breakpoint exception
CUn	Coprocessor Unusable exception
RSVD	Reserved Instruction exception
Trap	Trap exception
OVF	Integer overflow exception
DAErr	Data Address Error exception
Reset	Reset exception
DTLB	DTLB exception
DTMod	DTLB Modified exception
DPErr	Data Parity Error exception
WAT	Watch exception
DBE	Data Bus Error exception

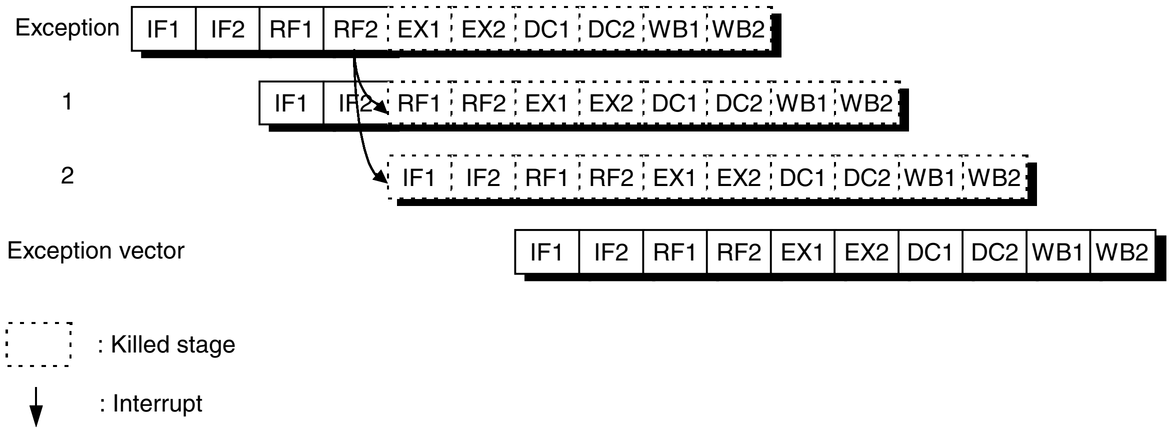
2.3.5.1 Exception conditions

When an exception condition occurs, the relevant instruction and all those that follow it in the pipeline are cancelled. Accordingly, any stall conditions and any later exception conditions that may have referenced this instruction are inhibited; there is no benefit in servicing stalls for a cancelled instruction.

When an exceptional conditions is detected for an instruction, the VR4120A will discard it and all following instructions. When this instruction reaches the WB stage, the exception flag and various information items are written to CP0 registers. The current PC is changed to the appropriate exception vector address and the exception bits of earlier pipeline stages are cleared.

This implementation allows all preceding instructions to complete execution and prevents all subsequent instructions from completing. Thus the value in the EPC is sufficient to restart execution. It also ensures that exceptions are taken in the order of execution; an instruction taking an exception may itself be killed by an instruction further down the pipeline that takes an exception in a later cycle.

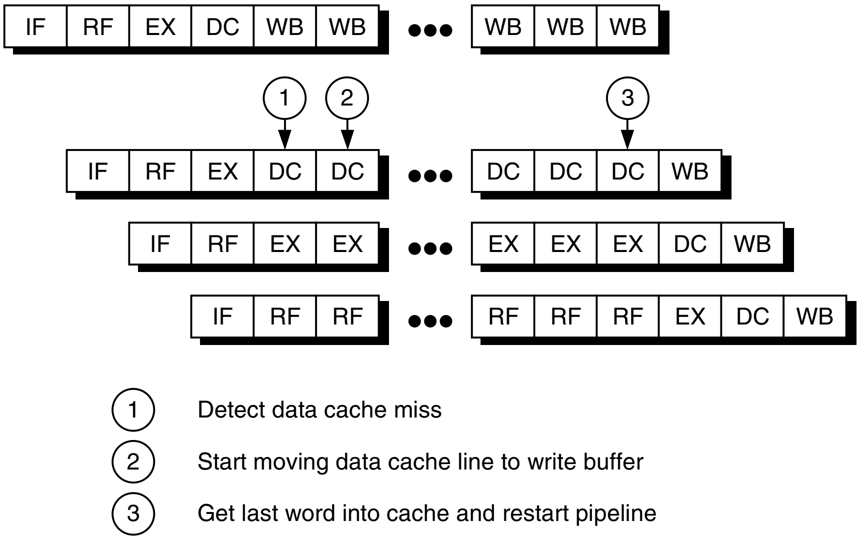
Figure 2-20. Exception Detection


Exception
IF1 IF2 RF1 RF2 EX1 EX2 DC1 DC2 WB1 WB2
1
IF1 IF2 RF1 RF2 EX1 EX2 DC1 DC2 WB1 WB2
2
IF1 IF2 RF1 RF2 EX1 EX2 DC1 DC2 WB1 WB2
Exception vector
IF1 IF2 RF1 RF2 EX1 EX2 DC1 DC2 WB1 WB2
: Killed stage
: Interrupt

2.3.5.2 Stall conditions

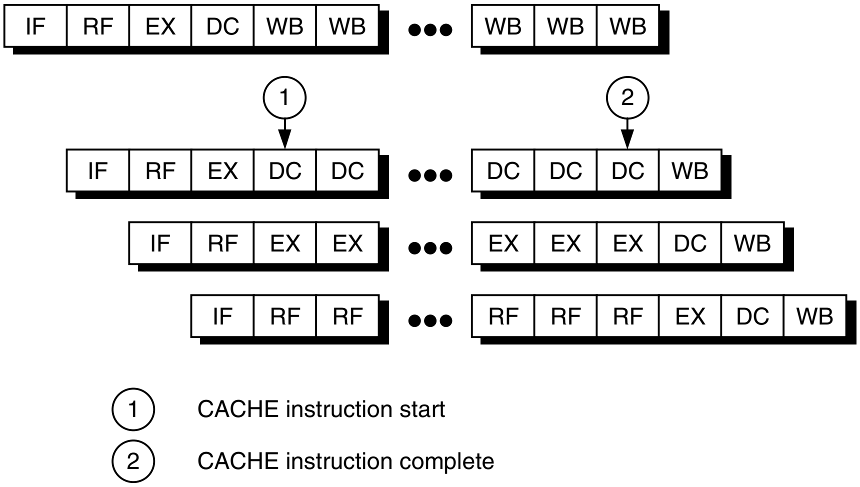
Stalls are used to stop the pipeline for conditions detected after the RF stage. When a stall occurs, the processor will resolve the condition and then the pipeline will continue. Figure 2-21 shows a data cache miss stall, and Figure 2-22 shows a CACHE instruction stall.

Figure 2-21. Data Cache Miss Stall


IF RF EX DC WB WB ••• WB WB WB
1
2
3
IF RF EX DC DC ••• DC DC DC WB
IF RF EX EX ••• EX EX EX DC WB
IF RF RF ••• RF RF RF EX DC WB
1 Detect data cache miss
2 Start moving data cache line to write buffer
3 Get last word into cache and restart pipeline

If the cache line to be replaced is dirty — the W bit is set — the data is moved to the internal write buffer in the next cycle. The write-back data is returned to memory. The last word in the data is returned to the cache at 3, and pipelining restarts.

Figure 2-22. CACHE Instruction Stall

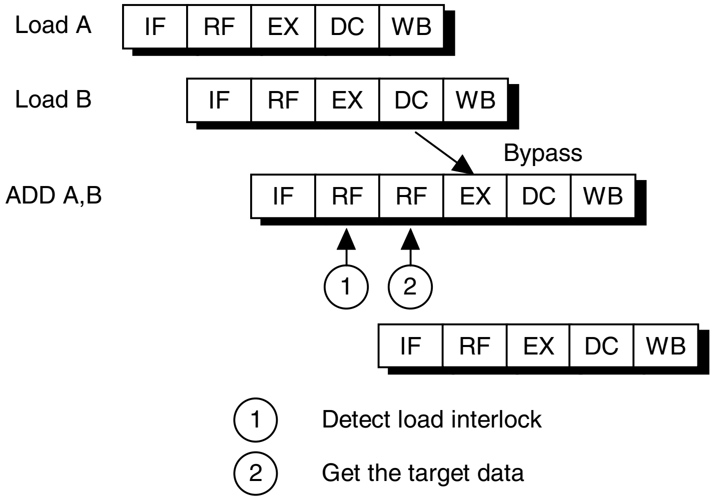

IF RF EX DC WB WB ••• WB WB WB
1
2
IF RF EX DC DC ••• DC DC DC WB
IF RF EX EX ••• EX EX EX DC WB
IF RF RF ••• RF RF RF EX DC WB
1 CACHE instruction start
2 CACHE instruction complete

When the CACHE instruction enters the DC stage, the pipeline stalls while the CACHE instruction is executed. The pipeline begins running again when the CACHE instruction is completed, allowing the instruction fetch to proceed.

2.3.5.3 Slip conditions

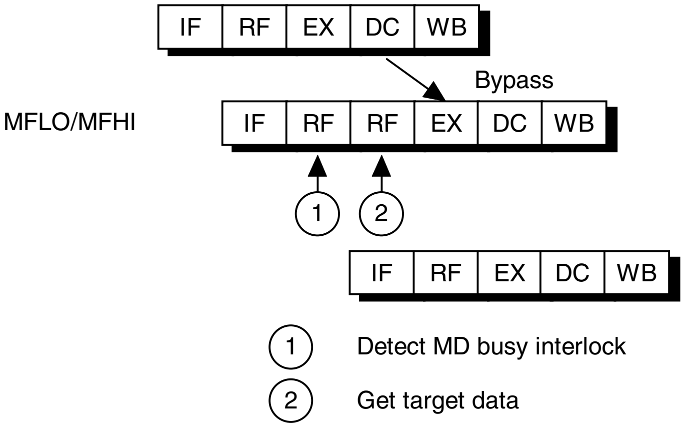
During *Φ*2 of the RF stage and Φ1 of the EX stage, internal logic will determine whether it is possible to start the current instruction in this cycle. If all of the source operands are available (either from the register file or via the internal bypass logic) and all the hardware resources necessary to complete the instruction will be available whenever required, then the instruction “run”; otherwise, the instruction will “slip”. Slipped instructions are retired on subsequent cycles until they issue. The backend of the pipeline (stages DC and WB) will advance normally during slips in an attempt to resolve the conflict. NOPs will be inserted into the bubble in the pipeline. Instructions killed by branch likely instructions, ERET or exceptions will not cause slips.

Figure 2-23. Load Data Interlock



Load Data Interlock is detected in the RF stage shown in as Figure 2-23 and also the pipeline slips in the stage. Load Data Interlock occurs when data fetched by a load instruction and data moved from HI, LO or CP0 registers is required by the next immediate instruction. The pipeline begins running again when the clock after the target of the load is read from the data cache, HI, LO and CP0 registers. The data returned at the end of the DC stage is input into the end of the RF stage, using the bypass multiplexers.

Figure 2-24. MD Busy Interlock



MD Busy Interlock is detected in the RF stage as shown in Figure 2-24 and also the pipeline slips in the stage. MD Busy Interlock occurs when HI/LO register is required by MFHI/MFLO instruction before finishing Mult/Div execution. The pipeline begins running again the clock after finishing Mult/Div execution. The data returned from the HI/LO register at the end of the DC stage is input into the end of the RF stage, using the bypass multiplexers.

Store-Load Interlock is detected in the EX stage and the pipeline slips in the RF stage. Store-Load Interlock occurs when store instruction followed by load instruction is detected. The pipeline begins running again one clock after.

Coprocessor 0 Interlock is detected in the EX stage and the pipeline slips in the RF stage. A coprocessor interlock occurs when an MTC0 instruction for the Configuration or Status register is detected.

The pipeline begins running again one clock after.

2.3.5.4 Bypassing

In some cases, data and conditions produced in the EX, DC and WB stages of the pipeline are made available to the EX stage (only) through the bypass data path.

Operand bypass allows an instruction in the EX stage to continue without having to wait for data or conditions to be written to the register file at the end of the WB stage. Instead, the Bypass Control Unit is responsible for ensuring data and conditions from later pipeline stages are available at the appropriate time for instructions earlier in the pipeline.

The Bypass Control Unit is also responsible for controlling the source and destination register addresses supplied to the register file.

2.3.6 Program compatibility

The VR4120A core is designed taking into consideration program compatibility with other VR-Series processors. However, because the VR4120A differs from other processors in its architecture, it may not be able to run some programs that run on other processors. Likewise, programs that run on the VR4120A will not necessarily run on other processors. Matters which should be paid attention to when porting programs between the VR4120A core and other VR-Series processors are listed below.

- The VR4120A core does not support floating-point instructions since it has no Floating-Point Unit (FPU).
- Multiply-add instructions (DMACC, MACC) are added in the VR4120A.
- Instructions for power modes (HIBERNATE, STANDBY, SUSPEND) are added in the VR4120A to support power modes.
- The VR4120A does not have the LL bit to perform synchronization of multiprocessing. Therefore, the CPU core does not support instructions which manipulate the LL bit (LL, LLD, SC, SCD).
- A 16-bit length MIPS16 instruction set is added in the VR4120A (but the μPD98502 does not support MIPS16 mode).
- The CP0 hazards of the VR4120A are equally or less stringent than those of other processors (for details, see **APPENDIX B VR4120A COPROCESSOR 0 HAZARDS**).
- An instruction for debug has been added for the VR4120A. However, this instruction cannot be used for the VR4120A.

For more information, refer to **APPENDIX A MIPS III INSTRUCTION SET DETAILS**, the VR4100, VR4111™ User's Manual, or the VR4300™ User's Manual.

The list of instructions supported by VR-Series products is shown below.

Table 2-26. VR Series Supported Instructions

Product Instruction		VR4100 VR4102™	VR4111	VR4120A Core	VR4300 VR4305™ VR4310™	VR5000™ VR10000™
MIPS I instruction set		O	O	O	O	O
MIPS II instruction set		O	O	O	O	O
MIPS III instruction set		O	O	O	O	O
	LL bit operation	×	×	×	O	O
MIPS IV instruction set		×	×	×	×	O
MIPS16 instruction set		×	O	O Note	×	×
16-bit multiply-add operation		O	O	O (Use of 32-bit multiply-add operation	×	×
32-bit multiply-add operation		×	×	O	×	×
Floating-point operation		×	×	×	O	O
Power mode transfer		O	O	O	×	×

Note The μPD98502 does not support MIPS16 mode. The MIPD16EN pin (located at D11) should be connected to GND.

2.4 Memory Management System

The VR4120A Core provides a memory management unit (MMU) which uses a translation lookaside buffer (TLB) to translate virtual addresses into physical addresses. This chapter describes the virtual and physical address spaces, the virtual-to-physical address translation, the operation of the TLB in making these translations, and the CP0 registers that provide the software interface to the TLB.

2.4.1 Translation lookaside buffer (TLB)

Virtual addresses are translated into physical addresses using an on-chip TLB[Note]. The on-chip TLB is a fully-associative memory that holds 32 entries, which provide mapping to odd/even page in pairs for one entry. These pages can have five different sizes, 1 K, 4 K, 16 K, 64 K, and 256 K, and can be specified in each entry. If it is supplied with a virtual address, each of the TLB entries is checked simultaneously to see whether they match the virtual addresses that are provided with the ASID field and saved in the EntryHi register.

If there is a virtual address match, or “hit,” in the TLB, the physical page number is extracted from the TLB and concatenated with the offset to form the physical address.

If no match occurs (TLB “miss”), an exception is taken and software refills the TLB from the page table resident in memory. The software writes to an entry selected using the Index register or a random entry indicated in the Random register.

If more than one entry in the TLB matches the virtual address being translated, the operation is undefined and the TLB may be disabled. In this case, the TLB-Shutdown (TS) bit of the status register is set to 1, and the TLB becomes unusable (an attempt to access the TLB results in a TLB Mismatch exception regardless of whether there is an entry that hits). The TS bit can be cleared only by a reset.

Note Depending on the address space, virtual addresses may be converted to physical addresses without using a TLB. For example, address translation for the kseg0 or kseg1 address space does not use mapping. The physical addresses of these address spaces are determined by subtracting the base address of the address space from the virtual addresses.

2.4.2 Virtual address space

This section describes the virtual/physical address space and the manner in which virtual addresses are converted or “translated” into physical addresses in the TLB. The VR4120A virtual address can be either 32 or 64 bits wide, depending on whether the processor is operating in 32-bit or 64-bit mode.

In 32-bit mode, addresses are 32 bits wide. The maximum user process size is 2 Gbytes (2^{31}).
In 64-bit mode, addresses are 64 bits wide. The maximum user process size is 1 Tbyte (2^{40}).

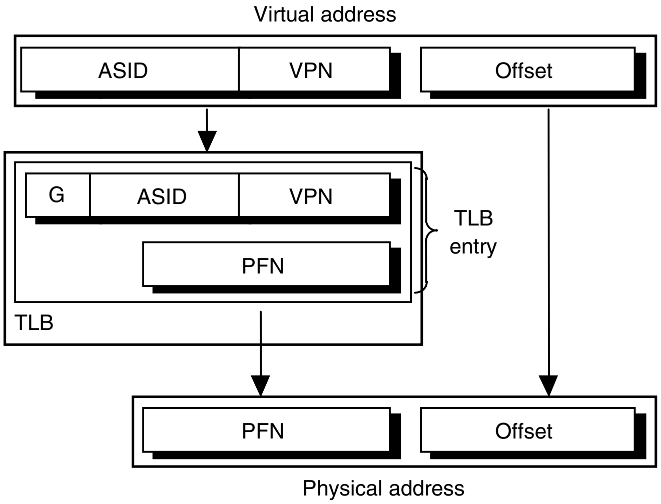
As shown in Figure 2-25, the virtual address is extended with an address space identifier (ASID), which reduces the frequency of TLB flushing when switching contexts. This 8-bit ASID is in the CP0 EntryHi register, described later in this chapter. The Global (G) bit is in the EntryLo0 and EntryLo1 registers, described later in this section.

Figure 2-25. Virtual-to-Physical Address Translation

1 The virtual page number (VPN) in the virtual address (VA) is compared with the VPN in the TLB.

2 If there is a match, the page frame number (PFN) representing the high-order bits of the physical address is output from the TLB.

3 The offset is then added to the PFN passing through the TLB.



2.4.2.1 Virtual-to-physical address translation

Converting a virtual address to a physical address begins by comparing the virtual address from the processor with the virtual addresses in the TLB; there is a match when the virtual page number (VPN) of the address is the same as the VPN field of the entry, and either:

- ✧ the Global (G) bit of the TLB entry is set to 1
- ✧ the ASID field of the virtual address is the same as the ASID field of the TLB entry.

This match is referred to as a TLB hit. If there is no match, a TLB Mismatch exception is taken by the processor and software is allowed to refill the TLB from a page table of virtual/physical addresses in memory.

If there is a virtual address match in the TLB, the physical address is output from the TLB and concatenated with the offset, which represents an address within the page frame space. The offset does not pass through the TLB. Instead, the low-order bits of the virtual address are output without being translated. For details about the physical address, see **Section 2.4.5.11 Virtual-to-physical address translation**.

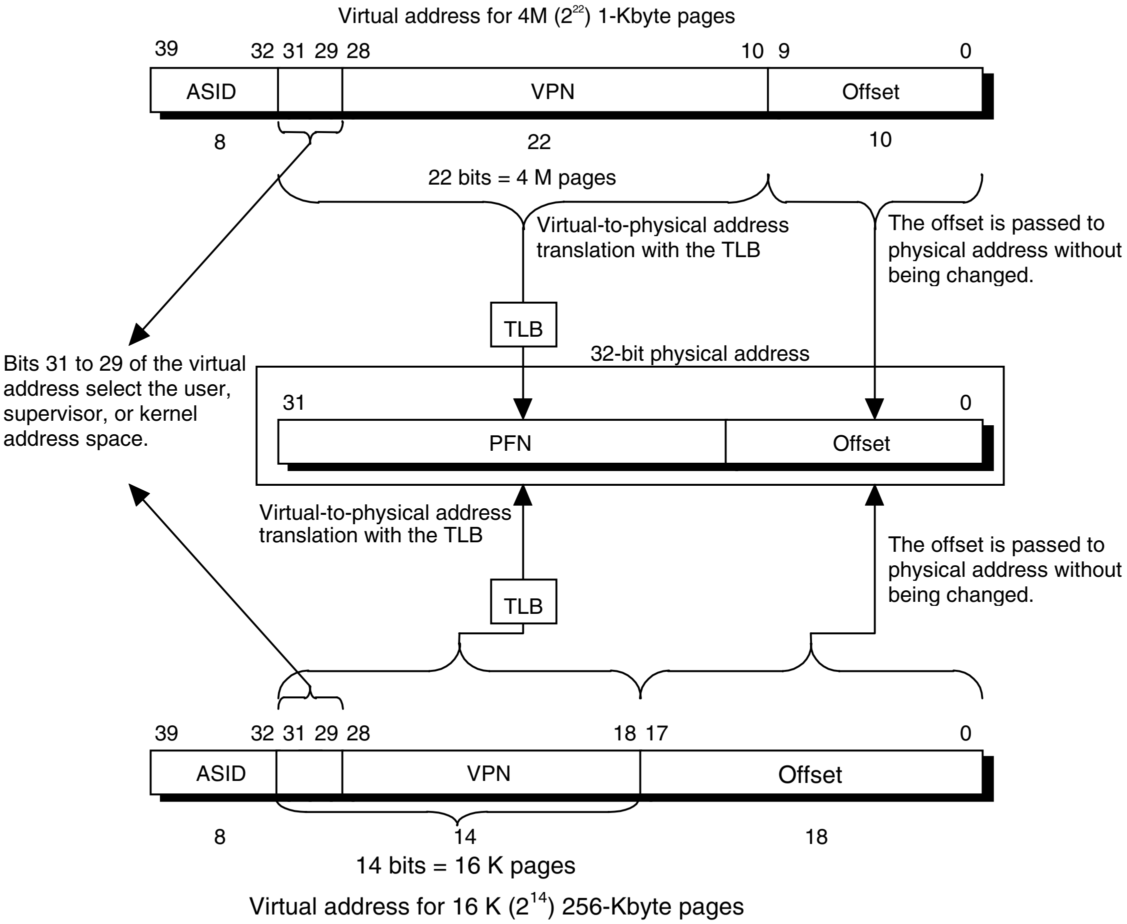
The next two sections describe the 32-bit and 64-bit mode address translations.

2.4.2.2 32-bit mode address translation

Figure 2-26 shows the virtual-to-physical-address translation of a 32-bit mode address. The pages can have five different sizes between 1 Kbyte (10 bits) and 256 Kbytes (18 bits), each being 4 times as large as the preceding one in ascending order, that is 1 K, 4 K, 16 K, 64 K, and 256 K.

- ✧ Shown at the top of Figure 2-26 is the virtual address space in which the page size is 1 Kbyte and the offset is 10 bits. The 22 bits excluding the ASID field represents the virtual page number (VPN), enabling selecting a page table of 4 M entries.
- ✧ Shown at the bottom of Figure 2-26 is the virtual address space in which the page size is 256 Kbytes and the offset is 18 bits. The 14 bits excluding the ASID field represents the VPN, enabling selecting a page table of 16 K entries.

Figure 2-26. 32-bit Mode Virtual Address Translation

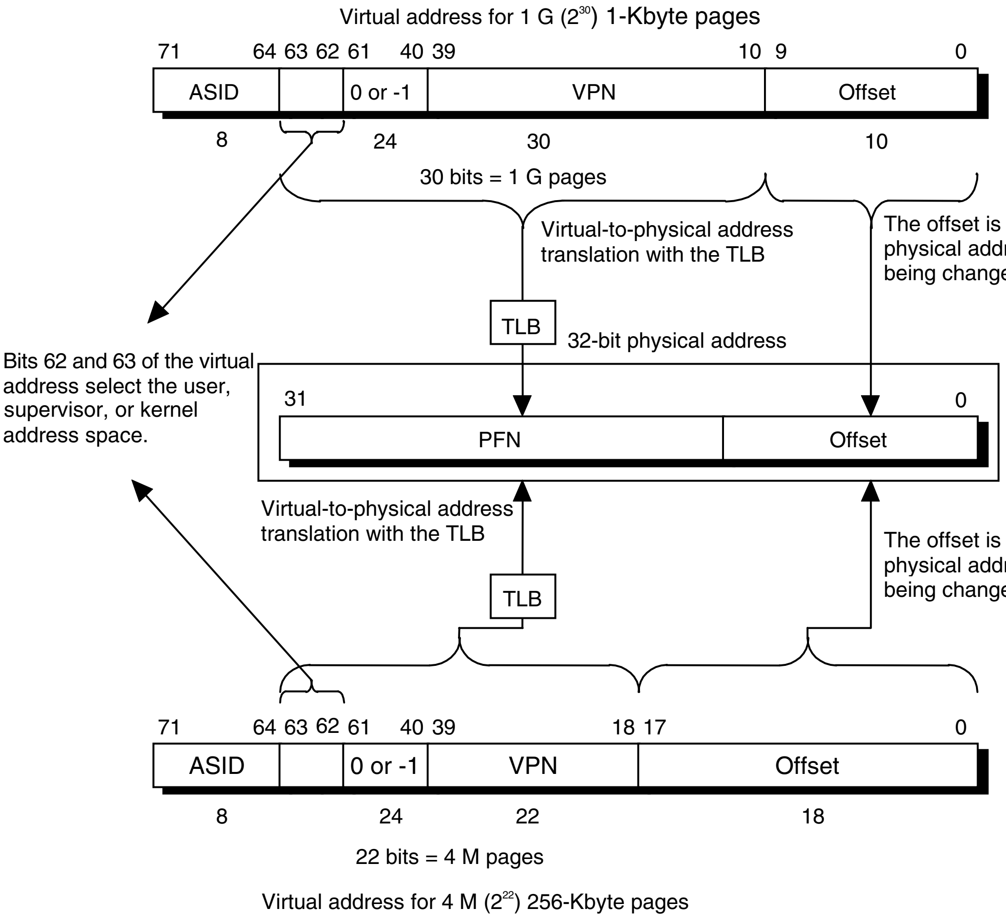

Virtual address for 4M (2^{22}) 1-Kbyte pages
39
32 31 29 28
10 9
0
ASID
VPN
Offset
8
22
10
22 bits = 4 M pages
Virtual-to-physical address translation with the TLB
The offset is passed to physical address without being changed.
TLB
32-bit physical address
Bits 31 to 29 of the virtual address select the user, supervisor, or kernel address space.
31
0
PFN
Offset
Virtual-to-physical address translation with the TLB
The offset is passed to physical address without being changed.
TLB
39
32 31 29 28
18 17
0
ASID
VPN
Offset
8
14
18
14 bits = 16 K pages
Virtual address for 16 K (2^{14}) 256-Kbyte pages

2.4.2.3 64-bit mode address translation

Figure 2-27 shows the virtual-to-physical-address translation of a 64-bit mode address. This figure illustrates the two possible page size; a 1-Kbyte page (10 bits) and a 256-Kbyte page (18 bits).

- ✧ Shown at the top of Figure 2-27 is the virtual address space in which the page size is 1 Kbyte and the offset is 10 bits. The 30 bits excluding the ASID field represents the virtual page number (VPN), enabling selecting a page table of 1 G entry.
- ✧ Shown at the bottom of Figure 2-27 is the virtual address space in which the page size is 256 Kbytes and the offset is 18 bits. The 22 bits excluding the ASID field represents the VPN, enabling selecting a page table of 4 M entries.

Figure 2-27. 64-bit Mode Virtual Address Translation


Virtual address for 1 G (2^30) 1-Kbyte pages
71
64 63 62 61
40 39
10 9
0
ASID
0 or -1
VPN
Offset
8
24
30
10
30 bits = 1 G pages
Virtual-to-physical address translation with the TLB
The offset is passed to physical address without being changed.
TLB
32-bit physical address
Bits 62 and 63 of the virtual address select the user, supervisor, or kernel address space.
31
0
PFN
Offset
Virtual-to-physical address translation with the TLB
The offset is passed to physical address without being changed.
TLB
71
64 63 62 61
40 39
18 17
0
ASID
0 or -1
VPN
Offset
8
24
22
18
22 bits = 4 M pages
Virtual address for 4 M (2^22) 256-Kbyte pages

2.4.2.4 Operating modes

The processor has three operating modes that function in both 32- and 64-bit operations:

- ✧ User mode
- ✧ Supervisor mode
- ✧ Kernel mode

User and Kernel modes are common to all VR-Series processors. Generally, Kernel mode is used to execute the operating system, while User mode is used to run application programs. The VR4000 Series processors have a third mode, which is called Supervisor mode and categorized in between User and Kernel modes. This mode is used to configure a high-security system.

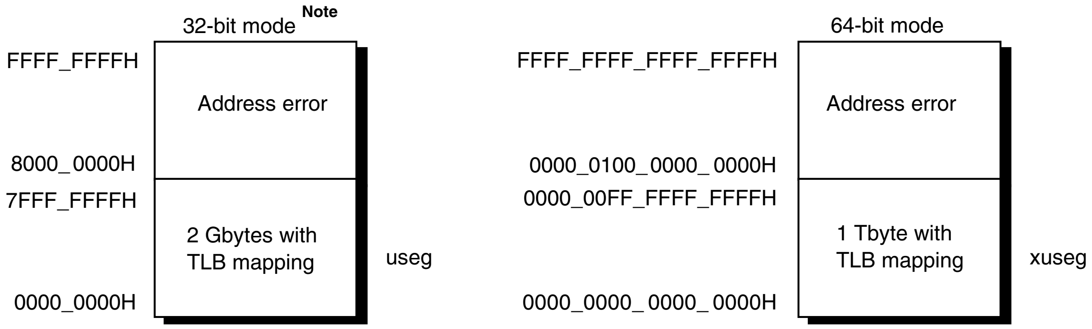
When an exception occurs, the CPU enters Kernel mode, and remains in this mode until an exception return instruction (ERET) is executed. The ERET instruction brings back the processor to the mode in which it was just before the exception occurs.

2.4.2.5 User mode virtual addressing

In user mode, a single virtual address space labeled User segment is available; its size is

- ✧ 2-Gbyte (2^{31} bytes) in 32-bit mode (useg)
- ✧ 1-Tbyte (2^{40} bytes) in 64-bit mode (xuseg)

Figure 2-28. User Mode Address Space



Note The VR4120A uses 64-bit addresses within it. When the processor is running in Kernel mode, it saves the contents of each register or restores their previous contents to initialize them before switching the context. For 32-bit mode addressing, bit 31 is sign-extended to bits 32 to 63, and the resulting 32 bits are used for addressing. Usually, it is impossible for 32-bit mode programs to generate invalid addresses. If context switching occurs and the processor enters Kernel mode, however, an attempt may be made to save an address other than the sign-extended 32-bit address mentioned above to a 64-bit register. In this case, user-mode programs are likely to generate an invalid address.

The User segment starts at address 0 and the current active user process resides in either useg (in 32-bit mode) or xuseg (in 64-bit mode). The TLB identically maps all references to useg/xuseg from all modes, and controls cache accessibility.

The processor operates in User mode when the Status register contains the following bit-values:

✧ KSU = 10
✧ EXL = 0
✧ ERL = 0

In conjunction with these bits, the UX bit in the Status register selects addressing mode as follows:

✧ When UX = 0, 32-bit useg space is selected.
✧ When UX = 1, 64-bit xuseg space is selected.

Table 2-27 lists the characteristics of each user segment (useg and xuseg).

Table 2-27. Comparison of useg and xuseg

Address Bit	Status Register Bit Value				Segment	Address Range	Size
Value	KSU	EXL	ERL	UX	Name		
32-bit A31 = 0	10	0	0	0	useg	0000_0000H to 7FFF_FFFFH	2 Gbytes (2^{31} bytes)
64-bit A(63:40) = 0	10	0	0	1	xuseg	0000_0000_0000_0000H to 0000_00FF_FFFF_FFFFH	1 Tbyte (2^{40} bytes)

(1) useg (32-bit mode)

In User mode, when UX = 0 in the Status register and the most significant bit of the virtual address is 0, this virtual address space is labeled useg.

Any attempt to reference an address with the most-significant bit set while in User mode causes an Address Error exception (see **Section 2.5 Exception Processing**).

The TLB Mismatch exception vector is used for TLB misses.

(2) xuseg (64-bit mode)

In User mode, when UX = 1 in the Status register and bits 63 to 40 of the virtual address are all 0, this virtual address space is labeled xuseg.

Any attempt to reference an address with bits 63 to 40 equal to 1 causes an Address Error exception (see **Section 2.5 Exception Processing**).

The XTLB Mismatch exception vector is used for TLB misses.

2.4.2.6 Supervisor-mode virtual addressing

Supervisor mode shown in Figure 2-29 is designed for layered operating systems in which a true kernel runs in Kernel mode, and the rest of the operating system runs in Supervisor mode.

The processor operates in Supervisor mode when the Status register contains the following bit-values:

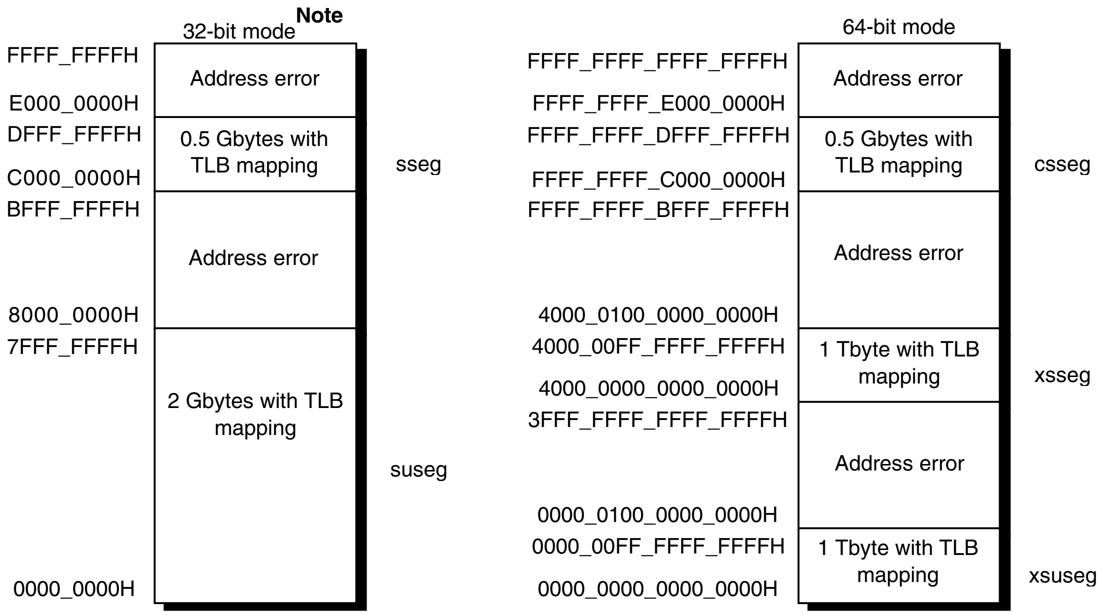
- ✧ KSU = 01
- ✧ EXL = 0
- ✧ ERL = 0

In conjunction with these bits, the SX bit in the Status register selects Supervisor mode addressing:

- ✧ When SX = 0: 32-bit supervisor space is selected.
- ✧ When SX = 1: 64-bit supervisor space is selected.

Figure 2-29 shows the supervisor mode address mapping, and Table 2-28 lists the characteristics of the Supervisor mode segments.

Figure 2-29. Supervisor Mode Address Space



Note The VR4120A uses 64-bit addresses within it. For 32-bit mode addressing, bit 31 is sign-extended to bits 32 to 63, and the resulting 32 bits are used for addressing. Usually, it is impossible for 32-bit mode programs to generate invalid addresses. In an operation of base register + offset for addressing, however, a two's complement overflow may occur, causing an invalid address. Note that the result becomes undefined. Two factors that can cause a two’s complement follow:

- ✧ When offset bit 15 is 0, base register bit 31 is 0, and bit 31 of the operation “base register + offset” is 1
- ✧ When offset bit 15 is 1, base register bit 31 is 1, and bit 31 of the operation “base register + offset” is 0

Table 2-28. 32-bit and 64-bit Supervisor Mode Segments

Address Bit Value	Status Register Bit Value				Segment Name	Address Range	Size
	KSU	EXL	ERL	SX			
32-bit A31 = 0	01	0	0	0	suseg	0000_0000H to 7FFF_FFFFH	2 Gbytes (2^{31} bytes)
32-bit A(31:29) = 110	01	0	0	0	sseg	C000_0000H to DFFF_FFFFH	512 Mbytes (2^{29} bytes)
64-bit A(63:62) = 00	01	0	0	1	xsuseg	0000_0000_0000_0000H to 0000_00FF_FFFF_FFFFH	1 Tbyte (2^{40} bytes)
64-bit A(63:62) = 01	01	0	0	1	xsseg	4000_0000_0000_0000H to 4000_00FF_FFFF_FFFFH	1 Tbyte (2^{40} bytes)
64-bit A(63:62) = 11	01	0	0	1	csseg	FFFF_FFFF_C000_0000H to FFFF_FFFF_DFFF_FFFFH	512 Mbytes (2^{29} bytes)

(1) suseg (32-bit supervisor mode, user space)

When SX = 0 in the Status register and the most-significant bit of the virtual address space is set to 0, the suseg virtual address space is selected; it covers 2 Gbytes (2^{31} bytes) of the current user address space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address. This mapped space starts at virtual address 0000_0000H and runs through 7FFF_FFFFH.

(2) sseg (32-bit supervisor mode, supervisor space)

When SX = 0 in the Status register and the most-significant three bits of the virtual address space are 110, the sseg virtual address space is selected; it covers 512 Mbytes (2^{29} bytes) of the current supervisor virtual address space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address. This mapped space begins at virtual address C000_0000H and runs through DFFF_FFFFH.

(3) xsuseg (64-bit supervisor mode, user space)

When SX = 1 in the Status register and bits 63 and 62 of the virtual address space are set to 00, the xsuseg virtual address space is selected; it covers 1 Tbyte (2^{40} bytes) of the current user address space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address. This mapped space starts at virtual address 0000_0000_0000_0000H and runs through 0000_00FF_FFFF_FFFFH.

(4) xsseg (64-bit supervisor mode, current supervisor space)

When SX = 1 in the Status register and bits 63 and 62 of the virtual address space are set to 01, the xsseg virtual address space is selected; it covers 1 Tbyte (2^{40} bytes) of the current supervisor virtual address space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address. This mapped space begins at virtual address 4000_0000_0000_0000H and runs through 4000_00FF_FFFF_FFFFH.

(5) csseg (64-bit supervisor mode, separate supervisor space)

When SX = 1 in the Status register and bits 63 and 62 of the virtual address space are set to 11, the csseg virtual address space is selected; it covers 512 Mbytes (2^{29} bytes) of the separate supervisor virtual address space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address. This mapped space begins at virtual address FFFF_FFFFH_C000_0000 and runs through FFFF_FFFF_DFFF_FFFFH.

2.4.2.7 Kernel-mode virtual addressing

If the Status register satisfies any of the following conditions, the processor runs in Kernel mode.

- ✧ KSU = 00
- ✧ EXL = 1
- ✧ ERL = 1

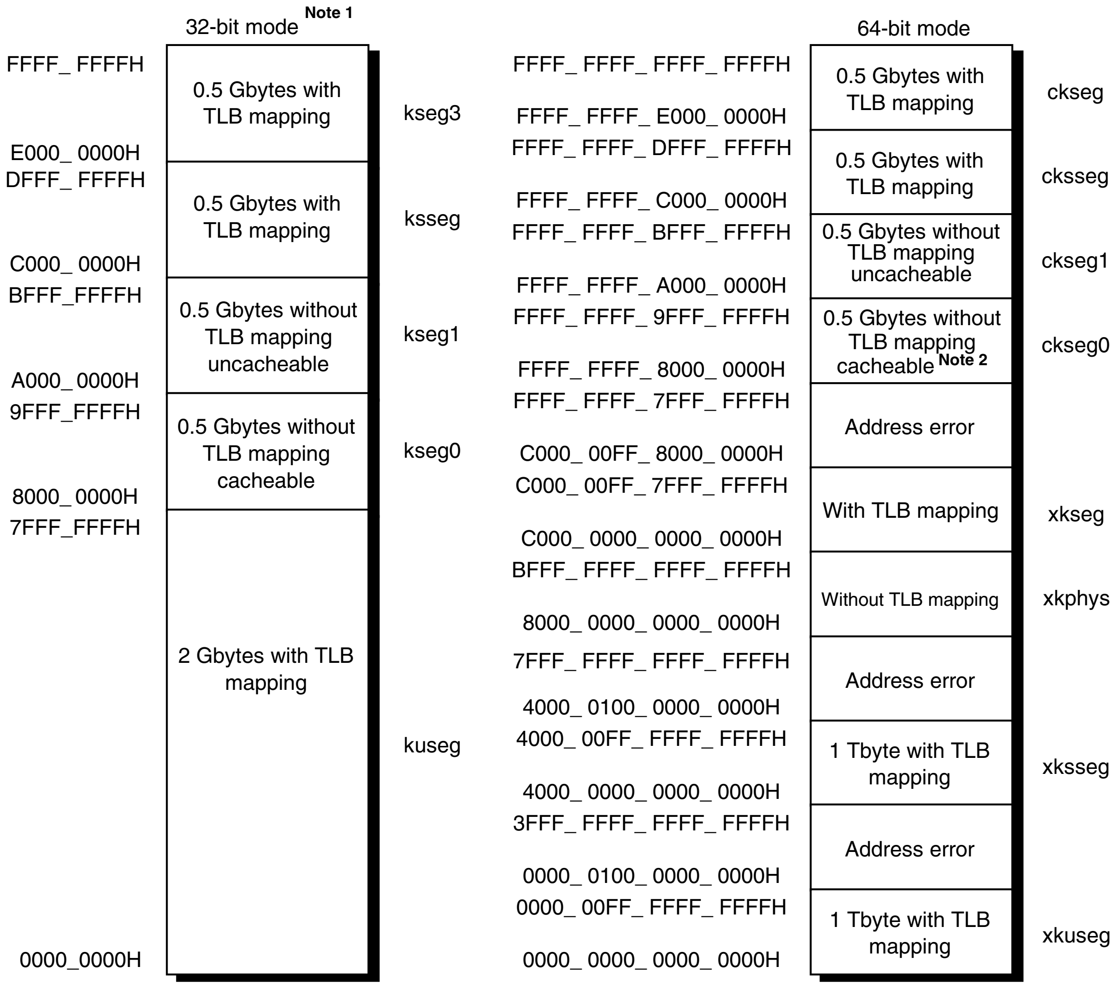
The addressing width in Kernel mode varies according to the state of the KX bit of the Status register, as follows:

- ✧ When KX = 0: 32-bit kernel space is selected.
- ✧ When KX = 1: 64-bit kernel space is selected.

The processor enters Kernel mode whenever an exception is detected and it remains in Kernel mode until an exception return (ERET) instruction is executed and results in ERL and/or EXL = 0. The ERET instruction restores the processor to the mode existing prior to the exception.

Kernel mode virtual address space is divided into regions differentiated by the high-order bits of the virtual address, as shown in Figure 2-30. Table 2-29 lists the characteristics of the 32-bit Kernel mode segments, and Table 2-30 lists the characteristics of the 64-bit Kernel mode segments.

Figure 2-30. Kernel Mode Address Space



Notes 1. The VR4120A uses 64-bit addresses within it. For 32-bit mode addressing, bit 31 is sign-extended to bits 32 to 63, and the resulting 32 bits are used for addressing. Usually, a 64-bit instruction is used for the program in 32-bit mode.

2. The K0 field of the Config register controls cacheability of kseg0 and ckseg0.

Table 2-29. 32-bit Kernel Mode Segments

Address Bit	Status Register Bit Value				Segment	Virtual	Physical	Size
Value	KSU	EXL	ERL	KX	Name	Address	Address	
A31 = 0	KSU = 00 or EXL = 1 or ERL = 1			0	kuseg	0000_0000H to 7FFF_FFFFH	TLB map	2 Gbytes (2^{31} bytes)
A(31:29) = 100				0	kseg0	8000_0000H to 9FFF_FFFFH	0000_0000H to 1FFF_FFFFH	512 Mbytes (2^{29} bytes)
A(31:29) = 101				0	kseg1	A000_0000H to BFFF_FFFFH	0000_0000H to 1FFF_FFFFH	512 Mbytes (2^{29} bytes)
A(31:29) = 110				0	ksseg	C000_0000H to DFFF_FFFFH	TLB map	512 Mbytes (2^{29} bytes)
A(31:29) = 111				0	kseg3	E000_0000H to FFFF_FFFFH	TLB map	512 Mbytes (2^{29} bytes)

(1) kuseg (32-bit kernel mode, user space)

When KX = 0 in the Status register, and the most-significant bit of the virtual address space is 0, the kuseg virtual address space is selected; it is the current 2-Gbyte (2^{31}-byte) user address space.

The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address.

If the ERL bit of the Status register is 1, the user address space is assigned 2 Gbytes (2^{31} bytes) without TLB mapping and becomes unmapped (with virtual addresses being used as physical addresses) and uncached so that the cache error handler can use it. This allows the Cache Error exception code to operate uncached using r0 as a base register.

(2) kseg0 (32-bit kernel mode, kernel space 0)

When KX = 0 in the Status register and the most-significant three bits of the virtual address space are 100, the kseg0 virtual address space is selected; it is the current 512-Mbyte (2^{29}-byte) physical space.

References to kseg0 are not mapped through TLB; the physical address selected is defined by subtracting 8000 0000H from the virtual address.

The K0 field of the Config register controls cacheability (see **Section 2.5 Exception Processing**).

(3) kseg1 (32-bit kernel mode, kernel space 1)

When KX = 0 in the Status register and the most-significant three bits of the virtual address space are 101, the kseg1 virtual address space is selected; it is the current 512-Mbyte (2^{29}-byte) physical space.

References to kseg1 are not mapped through TLB; the physical address selected is defined by subtracting A000 0000H from the virtual address.

Caches are disabled for accesses to these addresses, and main memory (or memory-mapped I/O device registers) is accessed directly.

(4) ksseg (32-bit kernel mode, supervisor space)

When KX = 0 in the Status register and the most-significant three bits of the virtual address space are 110, the ksseg virtual address space is selected; it is the current 512-Mbyte (2^{29}-byte) virtual address space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address.

(5) kseg3 (32-bit kernel mode, kernel space 3)

When KX = 0 in the Status register and the most-significant three bits of the virtual address space are 111, the kseg3 virtual address space is selected; it is the current 512-Mbyte (2^{29}-byte) kernel virtual space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address.

Table 2-30. 64-bit Kernel Mode Segments

Address Bit	Status Register Bit Value				Segment	Virtual Address	Physical	Size
Value	KSU	EXL	ERL	KX	Name		Address	
A(63:62) = 00	KSU = 00 or EXL = 1 or ERL = 1			1	xkuseg	0000_0000_0000_0000H to 0000_00FF_FFFF_FFFFH	TLB map	1 Tbyte (2^{40} bytes)
A(63:62) = 01				1	xksseg	4000_0000_0000_0000H to 4000_00FF_FFFF_FFFFH	TLB map	1 Tbyte (2^{40} bytes)
A(63:62) = 10				1	xkphys	8000_0000_0000_0000H to BFFF_FFFF_FFFF_FFFFH	0000_0000H to FFFF_FFFFH	4 Gbytes (2^{32} bytes)
A(63:62) = 11				1	xkseg	C000_0000_0000_0000H to C000_00FF_7FFF_FFFFH	TLB map	2^{40} to 2^{31} bytes
A(63:62) = 11 A(63:31) = -1				1	ckseg0	FFFF_FFFF_8000_0000H to FFFF_FFFF_9FFF_FFFFH	0000_0000H to 1FFF_FFFFH	512 Mbytes (2^{29} bytes)
A(63:62) = 11 A(63:31) = -1				1	ckseg1	FFFF_FFFF_A000_0000H to FFFF_FFFF_BFFF_FFFFH	0000_0000H to 1FFF_FFFFH	512 Mbytes (2^{29} bytes)
A(63:62) = 11 A(63:31) = -1				1	cksseg	FFFF_FFFF_C000_0000H to FFFF_FFFF_DFFF_FFFFH	TLB map	512 Mbytes (2^{29} bytes)
A(63:62) = 11 A(63:31) = -1				1	ckseg3	FFFF_FFFF_E000_0000H to FFFF_FFFF_FFFF_FFFFH	TLB map	512 Mbytes (2^{29} bytes)

(6) xkuseg (64-bit kernel mode, user space)

When KX = 1 in the Status register and bits 63 and 62 of the virtual address space are 00, the xkuseg virtual address space is selected; it is the 1-Tbyte (2^{40} bytes) current user address space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address.

If the ERL bit of the Status register is 1, the user address space is assigned 2 Gbytes (2^{31} bytes) without TLB mapping and becomes unmapped (with virtual addresses being used as physical addresses) and uncached so that the cache error handler can use it. This allows the Cache Error exception code to operate uncached using r0 as a base register.

(7) xksseg (64-bit kernel mode, current supervisor space)

When KX = 1 in the Status register and bits 63 and 62 of the virtual address space are 01, the xksseg address space is selected; it is the 1-Tbyte (2^{40} bytes) current supervisor address space. The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address.

(8) xkphys (64-bit kernel mode, physical spaces)

When the KX = 1 in the Status register and bits 63 and 62 of the virtual address space are 10, the virtual address space is called xkphys and selected as either cached or uncached. If any of bits 58 to 32 of the address is 1, an attempt to access that address results in an address error.

Table 2-31. Cacheability and xkphys Address Space

Bits 61-59	Cacheability	Start Address
0	Cached	8000_0000_0000_0000H to 8000_0000_FFFF_FFFFH
1	Cached	8800_0000_0000_0000H to 8800_0000_FFFF_FFFFH
2	Uncached	9000_0000_0000_0000H to 9000_0000_FFFF_FFFFH
3	Cached	9800_0000_0000_0000H to 9800_0000_FFFF_FFFFH
4	Cached	A000_0000_0000_0000H to A000_0000_FFFF_FFFFH
5	Cached	A800_0000_0000_0000H to A800_0000_FFFF_FFFFH
6	Cached	B000_0000_0000_0000H to B000_0000_FFFF_FFFFH
7	Cached	B800_0000_0000_0000H to B800_0000_FFFF_FFFFH

(9) xkseg (64-bit kernel mode, physical spaces)

When the KX = 1 in the Status register and bits 63 and 62 of the virtual address space are 11, the virtual address space is called xkseg and selected as either of the following:

- Kernel virtual space xkseg, the current kernel virtual space; the virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address
  This space is referenced via TLB. Whether cache can be used or not is determined by bit C of each page's TLB entry.
- One of the four 32-bit kernel compatibility spaces, as described in the next section.

(10) 64-bit kernel mode compatible spaces (ckseg0, ckseg1, cksseg, and ckseg3)

If the conditions listed below are satisfied in Kernel mode, ckseg0, ckseg1, cksseg, or ckseg3 (each having 512 Mbytes) is selected as a compatible space according to the state of the bits 30 and 29 (two low-order bits) of the address.

✧ The KX bit of the Status register is 1.
✧ Bits 63 and 62 of the 64-bit virtual address are 11.
✧ Bits 61 to 31 of the virtual address are all 1.

(a) ckseg0

This space is an unmapped region, compatible with the 32-bit mode kseg0 space. The K0 field of the Config register controls cacheability and coherency.

(b) ckseg1

This space is an unmapped and uncached region, compatible with the 32-bit mode kseg1 space.

(c) cksseg

This space is the current supervisor virtual space, compatible with the 32-bit mode ksseg space.
References to cksseg are mapped through TLB. Whether cache can be used or not is determined by bit C of each page's TLB entry.

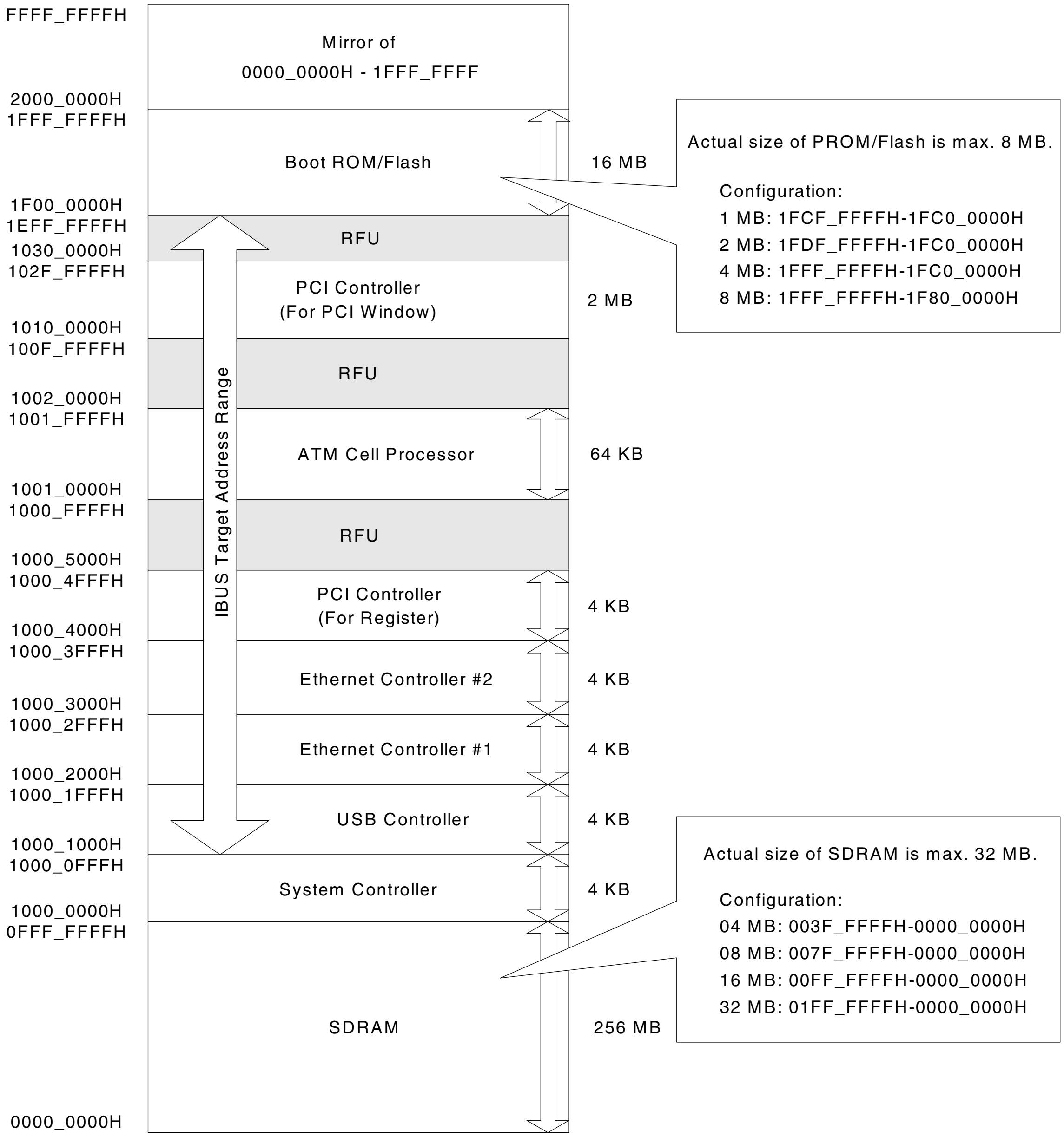
(d) ckseg3

This space is the current supervisor virtual space, compatible with the 32-bit mode kseg3 space.
References to ckseg3 are mapped through TLB. Whether cache can be used or not is determined by bit C of each page's TLB entry.

2.4.3 Physical address space

So VR4120A core uses a 32-bit address, that the processor physical address space encompasses 4 Gbytes. The VR4120A uses this 4-Gbyte physical address space as shown in Figure 2-31.

Figure 2-31. μPD98502 Physical Address Space

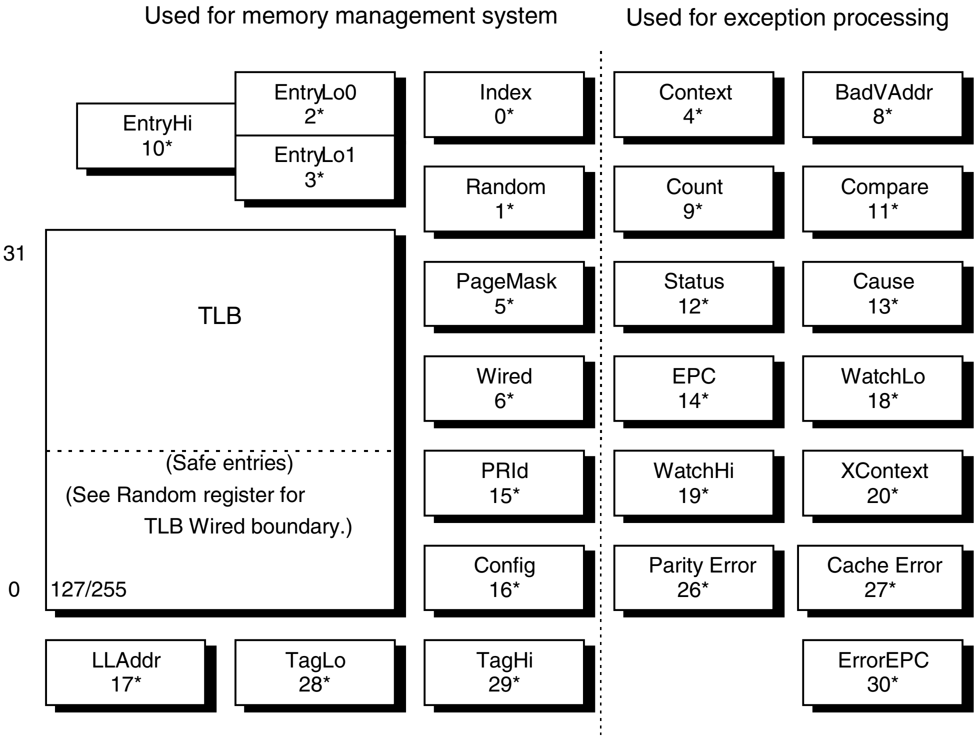


2.4.4 System control coprocessor

The System Control Coprocessor (CP0) is implemented as an integral part of the CPU, and supports memory management, address translation, exception processing, and other privileged operations. The CP0 contains the registers and a 32-entry TLB shown in Figure 2-32. The sections that follow describe how the processor uses each of the memory management-related registers.

Remark Each CP0 register has a unique number that identifies it; this number is referred to as the register number.

Figure 2-32. CP0 Registers and TLB



Remark *: Register number

2.4.4.1 Format of a TLB entry

Figure 2-33 shows the TLB entry formats for both 32- and 64-bit modes. Each field of an entry has a corresponding field in the EntryHi, EntryLo0, EntryLo1, or PageMask registers.

Figure 2-33. Format of a TLB Entry

(a) 32-bit mode

127 115	114 107	106 96
0	MASK	0
13	8	11

95 75	74	73 72	71 64
VPN2	G	0	ASID
21	1	2	8

63 60	59 38	37 35	34	33	32
0	PFN	C	D	V	0
4	22	3	1	1	1

31 28	27 6	5 3	2	1	0
0	PFN	C	D	V	0
4	22	3	1	1	1

(b) 64-bit mode

255 211	210 203	202 192
0	MASK	0
45	8	11

191 190	189 168	167 139	138	137 136	135 128
R	0	VPN2	G	0	ASID
2	22	29	1	2	8

127 92	91 70	69 67	66	65	64
0	PFN	C	D	V	0
36	22	3	1	1	1

63 28	27 6	5 3	2	1	0
0	PFN	C	D	V	0
36	22	3	1	1	1

The format of the EntryHi, EntryLo0, EntryLo1, and PageMask registers are nearly the same as the TLB entry. However, it is unknown what bit of the EntryHi register corresponds to the TLB G bit.

2.4.5 CP0 registers

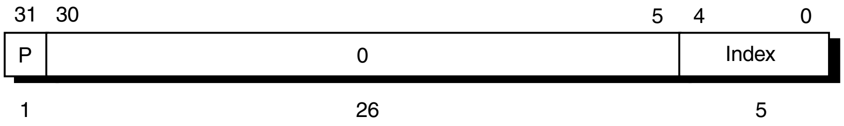
The CP0 registers explained below are accessed by the memory management system and software. The parenthesized number that follows each register name is the register number.

2.4.5.1 Index register (0)

The Index register is a 32-bit, read/write register containing five low-order bits to index an entry in the TLB. The most-significant bit of the register shows the success or failure of a TLB probe (TLBP) instruction.

The Index register also specifies the TLB entry affected by TLB read (TLBR) or TLB write index (TLBWI) instructions.

Figure 2-34. Index Register



P : Indicates whether probing is successful or not. It is set to 1 if the latest TLBP instruction fails. It is cleared to 0 when the TLBP instruction is successful.
Index : Specifies an index to a TLB entry that is a target of the TLBR or TLBWI instruction.
0 : RFU. Write 0 in a write operation. When this field is read, 0 is read.

2.4.5.2 Random register (1)

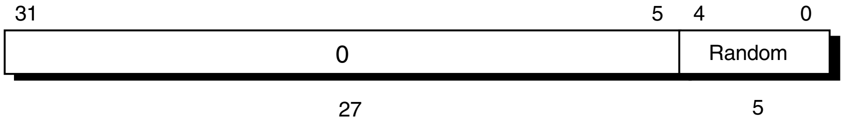
The Random register is a read-only register. The low-order 5 bits are used in referencing a TLB entry. This register is decremented each time an instruction is executed. The values that can be set in the register are as follows:

- ✧ The lower bound is the content of the Wired register.
- ✧ The upper bound is 31.

The Random register specifies the entry in the TLB that is affected by the TLBWR instruction. The register is readable to verify proper operation of the processor.

The Random register is set to the value of the upper bound upon Cold Reset. This register is also set to the upper bound when the Wired register is written. Figure 2-35 shows the format of the Random register.

Figure 2-35. Random Register

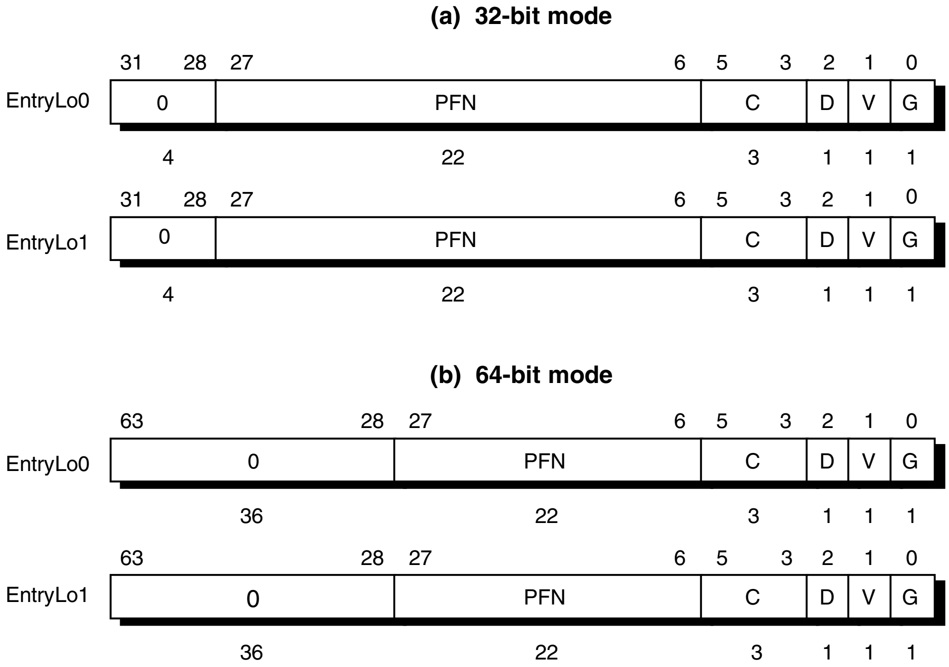


Random : TLB random index
0 : RFU. Write 0 in a write operation. When this field is read, 0 is read.

2.4.5.3 EntryLo0 (2) and EntryLo1 (3) registers

The EntryLo register consists of two registers that have identical formats: EntryLo0, used for even virtual pages and EntryLo1, used for odd virtual pages. The EntryLo0 and EntryLo1 registers are both read-/write-accessible. They are used to access the on-chip TLB. When a TLB read/write operation is carried out, the EntryLo0 and EntryLo1 registers hold the contents of the low-order 32 bits of TLB entries at even and odd addresses, respectively.

Figure 2-36. EntryLo0 and EntryLo1 Registers



(a) 32-bit mode

(b) 64-bit mode

PFN : Page frame number; high-order bits of the physical address.
C : Specifies the TLB page attribute (see Table 2-33).
D : Dirty. If this bit is set to 1, the page is marked as dirty and, therefore, writeable. This bit is actually a write-protect bit that software can use to prevent alteration of data.
V : Valid. If this bit is set to 1, it indicates that the TLB entry is valid; otherwise, a TLB Invalid exception (TLBL or TLBS) occurs.
G : Global. If this bit is set in both EntryLo0 and EntryLo1, then the processor ignores the ASID during TLB lookup.
0 : RFU. Write 0 in a write operation. When this field is read, 0 is read.

The coherency attribute (C) bits are used to specify whether to use the cache in referencing a page. When the cache is used, whether the page attribute is “cached” or “uncached” is selected by algorithm.

Table 2-32 lists the page attributes selected according to the value in the C bits.

Table 2-32. Cache Algorithm

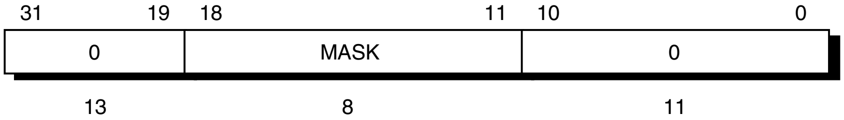
C Bit Value	Cache Algorithm
0	Cached
1	Cached
2	Uncached
3	Cached
4	Cached
5	Cached
6	Cached
7	Cached

2.4.5.4 PageMask register (5)

The PageMask register is a read/write register used for reading from or writing to the TLB; it holds a comparison mask that sets the page size for each TLB entry, as shown in Table 2-33. Page sizes must be from 1 Kbyte to 256 Kbytes.

TLB read and write instructions use this register as either a source or a destination; Bits 18 to 11 that are targets of comparison are masked during address translation.

Figure 2-37. Page Mask Register



MASK : Page comparison mask, which determines the virtual page size for the corresponding entry.
0 : RFU. Write 0 in a write operation. When this field is read, 0 is read.

Table 2-33 lists the mask pattern for each page size. If the mask pattern is one not listed below, the TLB behaves unexpectedly.

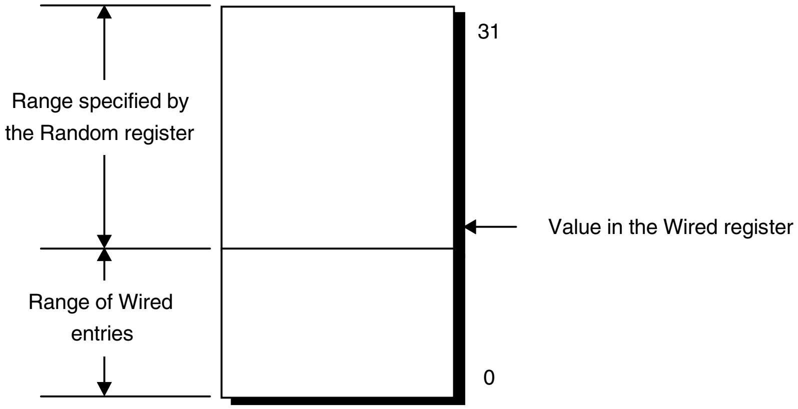
Table 2-33. Mask Values and Page Sizes

Page Size	Bit							
	18	17	16	15	14	13	12	11
1 Kbyte	0	0	0	0	0	0	0	0
4 Kbytes	0	0	0	0	0	0	1	1
16 Kbytes	0	0	0	0	1	1	1	1
64 Kbytes	0	0	1	1	1	1	1	1
256 Kbytes	1	1	1	1	1	1	1	1

2.4.5.5 Wired register (6)

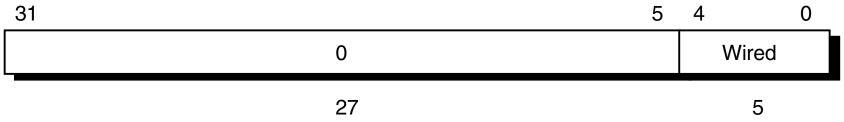
The Wired register is a read/write register that specifies the lower boundary of the random entry of the TLB as shown in Figure 2-38. Wired entries cannot be overwritten by a TLBWR instruction. They can, however, be overwritten by a TLBWI instruction. Random entries can be overwritten by both instructions.

Figure 2-38. Positions Indicated by Wired Register


31
Range specified by the Random register
Value in the Wired register
Range of Wired entries
0

The Wired register is set to 0 upon Cold Reset. Writing this register also sets the Random register to the value of its upper bound (see **Section 2.4.5.2 Random register (1)**). Figure 2-39 shows the format of the Wired register.

Figure 2-39. Wired Register



Wired : TLB wired boundary
0 : RFU. Write 0 in a write operation. When this field is read, 0 is read.

2.4.5.6 EntryHi register (10)

The EntryHi register is write-accessible. It is used to access the on-chip TLB. The EntryHi register holds the high-order bits of a TLB entry for TLB read and write operations. If a TLB Mismatch, TLB Invalid, or TLB Modified exception occurs, the EntryHi register holds the high-order bit of the TLB entry. The EntryHi register is also set with the virtual page number (VPN2) for a virtual address where an exception occurred and the ASID. See **Section 2.5 Exception Processing** for details of the TLB exception.

The ASID is used to read from or write to the ASID field of the TLB entry. It is also checked with the ASID of the TLB entry as the ASID of the virtual address during address translation.

The EntryHi register is accessed by the TLBP, TLBWR, TLBWI, and TLBR instructions.

Figure 2-40. EntryHi Register

(a) 32-bit mode

31 11	10 8	7 0
VPN2	0	ASID
21	3	8

(b) 64-bit mode

63 62	61 40	39 11	10 8	7 0
R	Fill	VPN2	0	ASID
2	22	29	3	8

VPN2: Virtual page number divided by two (mapping to two pages)
ASID : Address space ID. An 8-bit ASID field that lets multiple processes share the TLB; each process has a distinct mapping of otherwise identical virtual page numbers.
R : Space type (00 → user, 01 → supervisor, 11 → kernel). Matches bits 63 and 62 of the virtual address.
Fill : RFU. Ignored on write. When read, returns zero.
0 : RFU. Write 0 in a write operation. When this field is read, 0 is read.

2.4.5.7 Processor revision identifier (PRId) register (15)

The 32-bit, read-only Processor Revision Identifier (PRId) register contains information identifying the implementation and revision level of the CPU and CP0. Figure 2-41 shows the format of the PRId register.

Figure 2-41. PRId Register

31 16	15 8	7 0
0	Imp	Rev
16	8	8

Imp : CPU core processor ID number (0CH for the VR4120A)
Rev : CPU core processor revision number
0 : RFU. Write 0 in a write operation. When this field is read, 0 is read.

The processor revision number is stored as a value in the form y.x, where y is a major revision number in bits 7 to 4 and x is a minor revision number in bits 3 to 0.

The processor revision number can distinguish some CPU core revisions, however there is no guarantee that changes to the CPU core will necessarily be reflected in the PRId register, or that changes to the revision number necessarily reflect real CPU core changes. Therefore, create a program that does not depend on the processor revision number area.

2.4.5.8 Config register (16)

The Config register specifies various configuration options selected on VR4120A processors.

Some configuration options, as defined by the EC and BE fields, are set by the hardware during Cold Reset and are included in the Config register as read-only status bits for the software to access. Other configuration options are read/write (AD, EP, and K0 fields) and controlled by software; on Cold Reset these fields are undefined. Since only a subset of the VR4000 Series options are available in the VR4120A, some bits are set to constants (e.g., bits 14:13) that were variable in the VR4000 Series. The Config register should be initialized by software before caches are used. Figure 2-42 shows the format of the Config register.

Figure 2-42. Config Register Format

31	30 28	27 24	23	22 21	20	19 18	17	16	15	14 13	12	11 9	8 6	5	4 3	2 0
0	EC	EP	AD	0	M16	0	1	0	BE	10	CS	IC	DC	IB	0	K0
1	3	4	1	2	1	2	1	1	1	2	1	3	3	1	2	3

EC : Frequency ratio of system interface clock (VTCLK) (read only)
0 to 6 → RFU
7 → Pipeline clock (ACLK) frequency /1

EP : Transfer data pattern (cache write-back pattern) setting
0 → DD: 1 Word/1 Cycle
Others → RFU: Do not set

AD : Accelerate data mode
0 → VR4000 Series compatible mode
1 → RFU

M16: MIPS16 ISA mode enable/disable indication (read only)
0 → MIPS16 instruction cannot be executed (The μPD98502 sets “0” in this bit because it does not support MIPS16 mode).
1 → MIPS16 instruction can be executed.

BE : BigEndianMem. Endian mode of memory and a kernel.
0 → Little endian
1 → Big endian

CS : Cache size mode indication (fixed to 1 in the VR4120A)
0 → IC = $2^{(n+12)}$ Bytes, DC = $2^{(n+12)}$ Bytes
1 → IC = $2^{(n+10)}$ Bytes, DC = $2^{(n+10)}$ Bytes

IC : Instruction cache size indication. In the VR4120A, $2^{(IC+10)}$ bytes.
4 → 16 Kbytes
Others → RFU

DC : Data cache size indication. In the VR4120A, $2^{(DC+10)}$ bytes.
3 → 8 Kbytes
Others → RFU

IB : Select refill size (In this LSI, 8-word mode is not supported)
0 → 4 words (16 bytes)
1 → Do not set. (8 words (32 bytes))

K0 : kseg0 cache coherency algorithm
010 → Uncached
Others → Cached

1 : 1 is returned when read.

0 : 0 is returned when read.

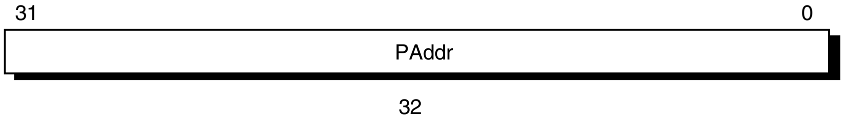
Caution Be sure to set the EP field, the AD bit and the IB bit to 0. If they are set with any other values, the processor may behave unexpectedly.

2.4.5.9 Load linked address (LLAddr) register (17)

The read/write Load Linked Address (LLAddr) register is not used with the VR4120A processor except for diagnostic purpose, and serves no function during normal operation.

LLAddr register is implemented just for compatibility between the VR4120A and VR4000/VR4400™.

Figure 2-43. LLAddr Register



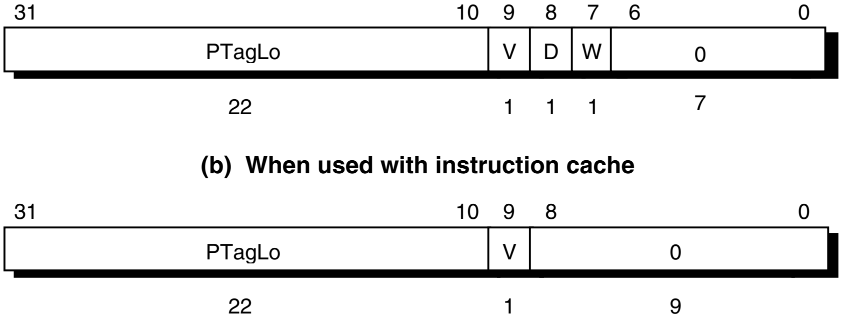
PAddr: 32-bit physical address

2.4.5.10 Cache tag registers (TagLo (28) and TagHi (29))

The TagLo and TagHi registers are 32-bit read/write registers that hold the primary cache tag during cache initialization, cache diagnostics, or cache error processing. The Tag registers are written by the CACHE and MTC0 instructions.

Figures 2-44 and 2-45 show the format of these registers.

Figure 2-44. TagLo Register



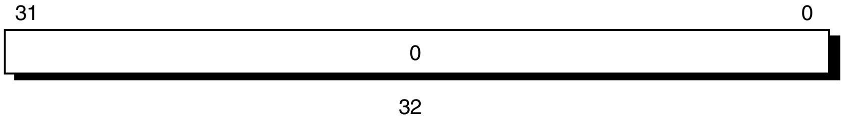
(a) When used with data cache

(b) When used with instruction cache

PTagLo: Specifies physical address bits 31 to 10.
V : Valid bit
D : Dirty bit. However, this bit is defined only for the compatibility with the VR4000 Series processors, and does not indicate the status of cache memory in spite of its readability and writability. This bit cannot change the status of cache memory.

W : Write-back bit (set if cache line has been updated)
0 : RFU. Write 0 in a write operation. When this field is read, 0 is read.

Figure 2-45. TagHi Register



0: RFU. Write 0 in a write operation. When this field is read, 0 is read.

2.4.5.11 Virtual-to-physical address translation

During virtual-to-physical address translation, the CPU compares the 8-bit ASID (when the Global bit, G, is not set to 1) of the virtual address to the ASID of the TLB entry to see if there is a match. One of the following comparisons are also made:

- ✧ In 32-bit mode, the high-order bits [Note 1] of the 32-bit virtual address are compared to the contents of the VPN2 (virtual page number divided by two) of each TLB entry.
- ✧ In 64-bit mode, the high-order bits [Note 2] of the 64-bit virtual address are compared to the contents of the VPN2 (virtual page number divided by two) of each TLB entry.

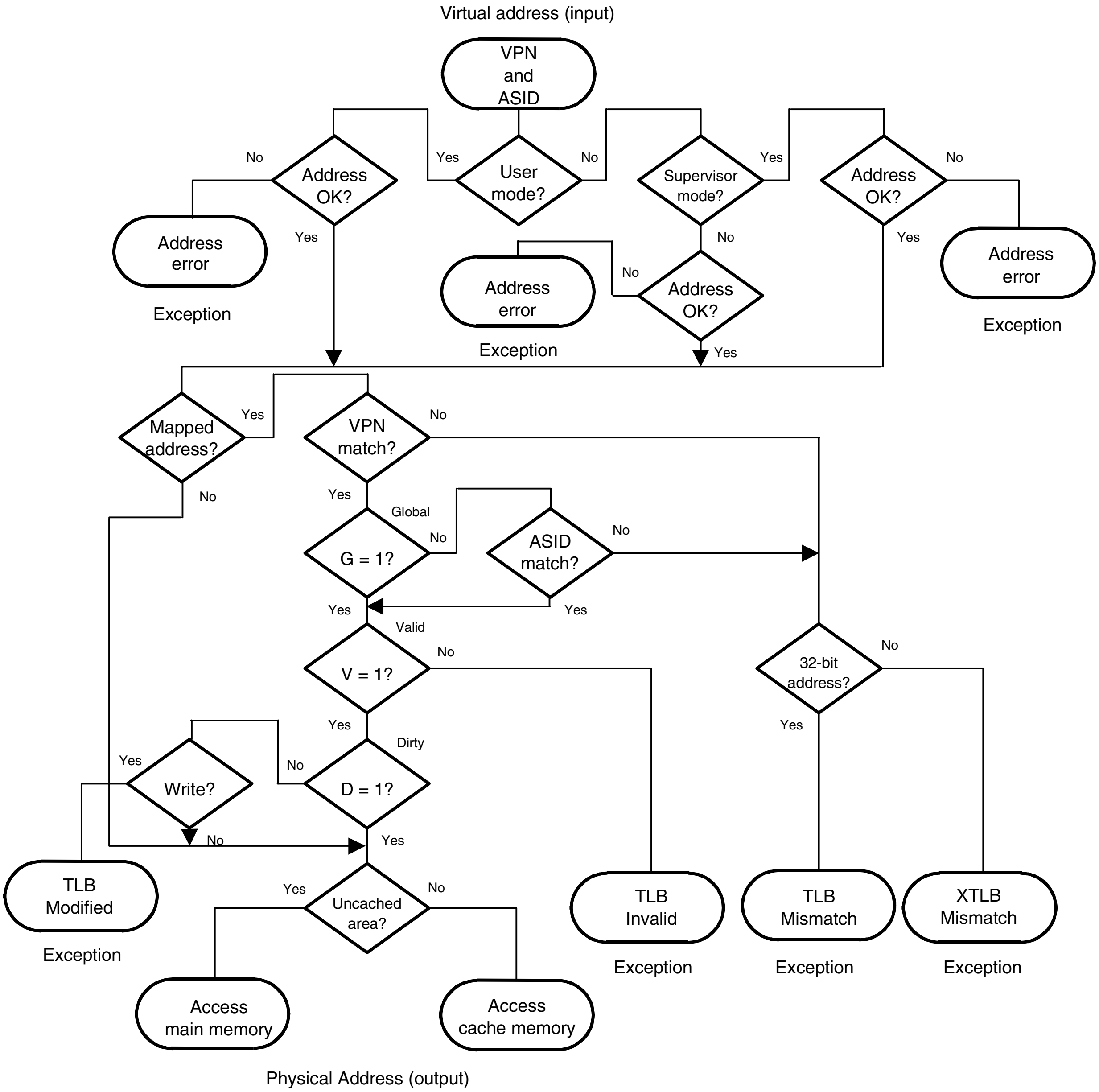
If a TLB entry matches, the physical address and access control bits (C, D, and V) are retrieved from the matching TLB entry. While the V bit of the entry must be set to 1 for a valid address translation to take place, it is not involved in the determination of a matching TLB entry.

Figure 2-46 illustrates the TLB address translation flow.

Notes 1. Up to bit 28. Number of bits depends on the TBL page size

2. Up to bit 29. Number of bits depends on the TBL page size

Figure 2-46. TLB Address Translation



2.4.5.12 TLB misses

If there is no TLB entry that matches the virtual address, a TLB Refill (miss) exception occurs[Note]. If the access control bits (D and V) indicate that the access is not valid, a TLB Modified or TLB Invalid exception occurs. If the C bit is 010, the retrieved physical address directly accesses main memory, bypassing the cache.

Note See **Section 2.5 Exception Processing** for details of the TLB Miss exception.

2.4.5.13 TLB instructions

The instructions used for TLB control are described below.

(1) Translation lookaside buffer probe (TLBP)

The translation lookaside buffer probe (TLBP) instruction loads the Index register with a TLB number that matches the content of the EntryHi register. If there is no TLB number that matches the TLB entry, the highest-order bit of the Index register is set.

(1) Translation lookaside buffer read (TLBR)

The translation lookaside buffer read (TLBR) instruction loads the EntryHi, EntryLo0, EntryLo1, and PageMask registers with the content of the TLB entry indicated by the content of the Index register.

(2) Translation lookaside buffer write index (TLBWI)

The translation lookaside buffer write index (TLBWI) instruction writes the contents of the EntryHi, EntryLo0, EntryLo1, and PageMask registers to the TLB entry indicated by the content of the Index register.

(3) Translation lookaside buffer write random (TLBWR)

The translation lookaside buffer write random (TLBWR) instruction writes the contents of the EntryHi, EntryLo0, EntryLo1, and PageMask registers to the TLB entry indicated by the content of the Random register.

2.5 Exception Processing

This chapter describes VR4120A CPU exception processing, including an explanation of hardware that processes exceptions.

2.5.1 Exception processing operation

The processor receives exceptions from a number of sources, including translation lookaside buffer (TLB) misses, arithmetic overflows, I/O interrupts, and system calls. When the CPU detects an exception, the normal sequence of instruction execution is suspended and the processor enters Kernel mode (see **Section 2.4 Memory Management System** for a description of system operating modes).

The processor then disables interrupts and transfers control for execution to the exception handler (located at a specific address as an exception handling routine implemented by software). The exception handler saves the context of the processor, including the contents of the program counter, the current operating mode (User or Supervisor), statuses, and interrupt enabling. This context is saved so it can be restored when the exception has been serviced.

When an exception occurs, the CPU loads the Exception Program Counter (EPC) register with a location where execution can restart after the exception has been serviced. The restart location in the EPC register is the address of the instruction that caused the exception or, if the instruction was executing in a branch delay slot, the address of the branch instruction immediately preceding the delay slot.

The VR4120A processor supports a Supervisor mode and high-speed TLB refill for all address spaces. The VR4120A CPU also provides the following functions:

- ✧ Interrupt enable (IE) bit
- ✧ Operating mode (User, Supervisor, or Kernel)
- ✧ Exception level (normal or exception is indicated by the EXL bit in the Status register)
- ✧ Error level (normal or error is indicated by the ERL bit in the Status register).

Interrupts are enabled when the following conditions are satisfied:

2.5.1.1 Interrupt enable

An interrupt is enabled when the following conditions are satisfied.

- Interrupt enable bit (IE) = 1
- EXL bit = 0, ERL bit = 0
- Corresponding IM field bits in the Status register = 1

2.5.1.2 Operating mode

The operating mode is specified by KSU bit in the Status register when both the exception level and error level are normal (0). The operation enters Kernel mode when either EXL bit or ERL bit in the Status register is set to 1.

2.5.1.3 Exception/error levels

Returning from an exception resets the exception level to normal (0) (for details, see **APPENDIX A MIPS III INSTRUCTION SET DETAILS**).

The registers that retain address, cause, and status information during exception processing are described in **Section 2.5.3 Exception processing registers**. For a description of the exception process, see **Section 2.5.4 Details of exceptions**.

2.5.2 Precision of exceptions

VR4120A CPU exceptions are logically precise; the instruction that causes an exception and all those that follow it are aborted and can be re-executed after servicing the exception. When succeeding instructions are discarded, exceptions associated with those instructions are also discarded. Exceptions are not taken in the order detected, but in instruction fetch order.

The exception handler can determine the cause of an exception and the address. The program can be restarted by rewriting the destination register - not automatically, however, as in the case of all the other precise exceptions where no status change occurs.

2.5.3 Exception processing registers

This section describes the CP0 registers that are used in exception processing. Table 2-34 lists these registers, along with their number-each register has a unique identification number that is referred to as its register number. The CP0 registers not listed in the table are used in memory management (for details, see **Section 2.4 Memory Management System**).

The exception handler examines the CP0 registers during exception processing to determine the cause of the exception and the state of the CPU at the time the exception occurred.

The registers in Table 2-34 are used in exception processing, and are described in the sections that follow.

Table 2-34. CP0 Exception Processing Registers

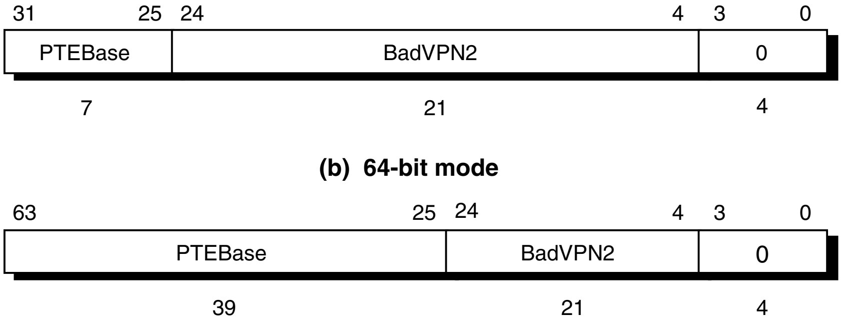
Register Name	Register Number
Context register	4
BadVAddr register	8
Count register	9
Compare register	11
Status register	12
Cause register	13
EPC register	14
WatchLo register	18
WatchHi register	19
XContext register	20
Parity Error register**Note**	26
Cache Error register**Note**	27
Error EPC register	30

Note This register is prepared to maintain compatibility with the VR4100. This register is not used in the μPD98502 hardware.

2.5.3.1 Context register (4)

The Context register is a read/write register containing the pointer to an entry in the page table entry (PTE) array on the memory; this array is a table that stores virtual-to-physical address translations. When there is a TLB miss, the operating system loads the unsuccessfully translated entry from the PTE array to the TLB. The Context register is used by the TLB Refill exception handler for loading TLB entries. The Context register duplicates some of the information provided in the BadVAddr register, but the information is arranged in a form that is more useful for a software TLB exception handler. Figure 2-47 shows the format of the Context register.

Figure 2-47. Context Register Format



(a) 32-bit mode

(b) 64-bit mode

PTEBase : The PTEBase field is a base address of the PTE entry table.
BadVPN2 : This field holds the value (VPN2) obtained by halving the virtual page number of the most recent virtual address for which translation failed.
0 : RFU. Write 0 in a write operation. When this field is read, 0 is read.

The PTEBase field is used by software as the pointer to the base address of the PTE table in the current user address space.

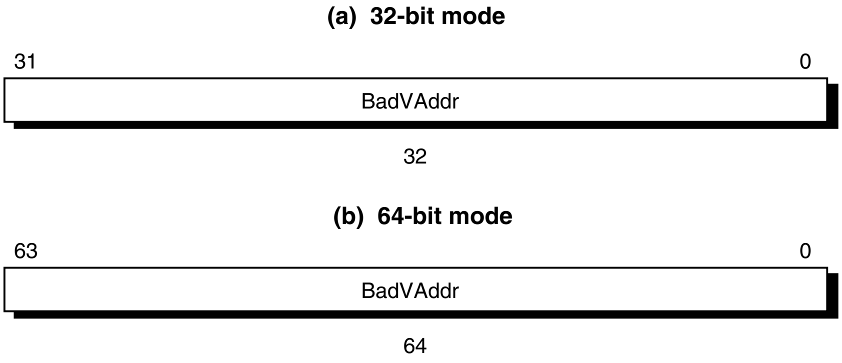
The 21-bit BadVPN2 field contains bits 31 to 11 of the virtual address that caused the TLB miss; bit 10 is excluded because a single TLB entry maps to an even-odd page pair. For a 1-Kbyte page size, this format can directly address the pair-table of 8-byte PTEs. When the page size is 4 Kbytes or more, shifting or masking this value produces the correct PTE reference address.

2.5.3.2 BadVAddr register (8)

The Bad Virtual Address (BadVAddr) register is a read-only register that saves the most recent virtual address that failed to have a valid translation, or that had an addressing error. Figure 2-48 shows the format of the BadVAddr register.

Caution This register saves no information after a bus error exception, because it is not an address error exception.

Figure 2-48. BadVAddr Register Format



(a) 32-bit mode

(b) 64-bit mode

BadVAddr: Most recent virtual address for which an addressing error occurred, or for which address translation failed.

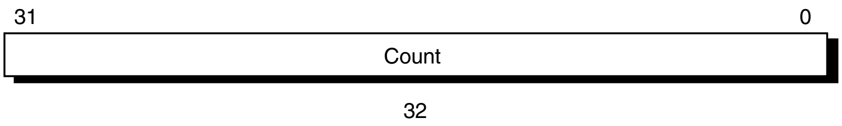
2.5.3.3 Count register (9)

The read/write Count register acts as a timer. It is incremented in synchronization with the frequencies of MasterOut clock, regardless of whether instructions are being executed, retired, or any forward progress is actually made through the pipeline.

This register is a free-running type. When the register reaches all ones, it rolls over to zero and continues counting. This register is used for self-diagnostic test, system initialization, or the establishment of inter-process synchronization.

Figure 2-49 shows the format of the Count register.

Figure 2-49. Count Register Format



Count: 32-bit up-date count value that is compared with the value of the Compare register.

2.5.3.4 Compare register (11)

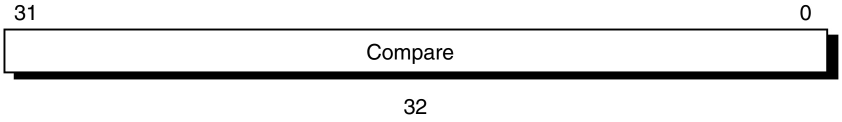
The Compare register causes a timer interrupt; it maintains a stable value that does not change on its own.

When the value of the Count register (see **Section 2.5.3.3 Count register (9)**) equals the value of the Compare register, the IP7 bit in the Cause register is set. This causes an interrupt as soon as the interrupt is enabled.

Writing a value to the Compare register, as a side effect, clears the timer interrupt request.

For diagnostic purposes, the Compare register is a read/write register. Normally, this register should be only used for a write. Figure 2-50 shows the format of the Compare register.

Figure 2-50. Compare Register Format

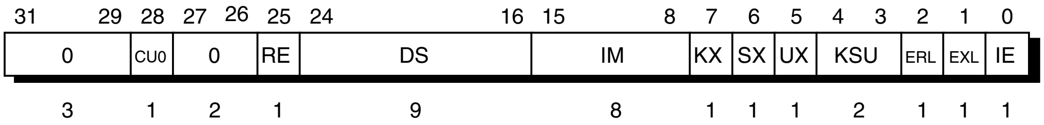


Compare: Value that is compared with the count value of the Count register.

2.5.3.5 Status register (12)

The Status register is a read/write register that contains the operating mode, interrupt enabling, and the diagnostic states of the processor. Figure 2-51 shows the format of the Status register.

Figure 2-51. Status Register Format



CU0 : Enables/disables the use of the coprocessor (1 → Enabled, 0 → Disabled).
CP0 can be used by the kernel at all times.

RE : Enables/disables reversing of the endian setting in User mode (0 → Disabled, 1 → Enabled).

Caution This bit must be set to 0.

DS : Diagnostic Status field (see Figure 2-52).

IM : Interrupt Mask field used to enable/disable external/internal and software interrupts (0 → Disabled, 1 → Enabled). This field consists of 8 bits that are used to control eight interrupts. The bits are assigned to interrupts as follows:

IM7 : Masks a timer interrupt.
IM (6:2) : Mask ordinary interrupts (Int (4:0)[Note]). However, Int4[Note] never occur in the VR4120A CPU.
IM (1:0) : Software interrupts.

Note Int (4:0) are internal signals of the CPU core. For details about connection to the on-chip peripheral units.

KX : Enables 64-bit addressing in Kernel mode (0 → 32-bit, 1 → 64-bit). If this bit is set, an XTLB Refill exception occurs if a TLB miss occurs in the Kernel mode address space.
In addition, 64-bit operations are always valid in kernel mode.

SX : Enables 64-bit addressing and operation in Supervisor mode (0 → 32-bit, 1 → 64-bit). If this bit is set, an XTLB Refill exception occurs if a TLB miss occurs in the Supervisor mode address space.

UX: : Enables 64-bit addressing and operation in User mode (0 → 32-bit, 1 → 64-bit). If this bit is set, an XTLB Refill exception occurs if a TLB miss occurs in the User mode address space.

KSU : Sets and indicates the operating mode (10 → User, 01 → Supervisor, 00 → Kernel).

ERL : Sets and indicates the error level (0 → Normal, 1 → Error).

EXL : Sets and indicates the exception level (0 → Normal, 1 → Exception).

IE : Sets and indicates interrupt enabling/disabling (0 → Disabled, 1 → Enabled).

0 : RFU. Write 0 in a write operation. When this bit is read, 0 is read.

Figure 2-52 shows the details of the Diagnostic Status (DS) field. All DS field bits other than the TS bit are writeable.

Figure 2-52. Status Register Diagnostic Status Field

24 23	22	21	20	19	18	17	16
0	BEV	TS	SR	0	CH	CE	DE
2	1	1	1	1	1	1	1

BEV : Specifies the base address of a TLB Refill exception vector and common exception vector (0 → Normal, 1 → Bootstrap).
TS : Occurs the TLB to be shut down (read-only) (0 → Not shut down, 1 → Shut down). This bit is used to avoid any problems that may occur when multiple TLB entries match the same virtual address. After the TLB has been shut down, reset the processor to enable restart. Note that the TLB is shut down even if a TLB entry matching a virtual address is marked as being invalid (with the V bit cleared).
SR : Occurs a Soft Reset or NMI exception (0 → Not occurred, 1 → Occurred).
CH : CP0 condition bit (0 → False, 1 → True). This bit can be read and written by software only; it cannot be accessed by hardware.
CE, DE: These are prepared to maintain compatibility with the VR4100, and are not used in the VR4120A Core hardware.
0 : RFU. Write 0 in a write operation. When this field is read, 0 is read.

The status register has the following fields where the modes and access status are set.

(1) Interrupt enable

Interrupts are enabled when all of the following conditions are true:

- IE is set to 1.
- EXL is cleared to 0.
- ERL is cleared to 0.
- The appropriate bit of the IM is set to 1.

(2) Operating modes

The following Status register bit settings are required for User, Kernel, and Supervisor modes.

- The processor is in User mode when KSU = 10, EXL = 0, and ERL = 0.
- The processor is in Supervisor mode when KSU = 01, EXL = 0, and ERL = 0.
- The processor is in Kernel mode when KSU = 00, EXL = 1, or ERL = 1.

(3) 32- and 64-bit modes

The following Status register bit settings select 32- or 64-bit operation for User, Kernel, and Supervisor operating modes. Enabling 64-bit operation permits the execution of 64-bit opcodes and translation of 64-bit addresses. 64-bit operation for User, Kernel and Supervisor modes can be set independently.

- 64-bit addressing for Kernel mode is enabled when KX bit = 1. 64-bit operations are always valid in Kernel mode.
- 64-bit addressing and operations are enabled for Supervisor mode when SX bit = 1.
- 64-bit addressing and operations are enabled for User mode when UX bit = 1.

(4) Kernel address space accesses

Access to the kernel address space is allowed when the processor is in Kernel mode.

(5) Supervisor address space accesses

Access to the supervisor address space is allowed when the processor is in Supervisor or Kernel mode.

(6) User address space accesses

Access to the user address space is allowed in any of the three operating modes.

(7) Status after reset

The contents of the Status register are undefined after Cold resets, except for the following bits in the diagnostic status field.

- TS and SR are cleared to 0.
- ERL and BEV are set to 1.
- SR is 0 after Cold reset, and is 1 after Soft reset or NMI interrupt.

Remark Cold reset and Soft reset are CPU core reset (see **Section 2.6 Initialization Interface**).

2.5.3.6 Cause register (13)

The 32-bit read/write Cause register holds the cause of the most recent exception. A 5-bit exception code indicates one of the causes (see Table 2-35). Other bits hold the detailed information of the specific exception. All bits in the Cause register, with the exception of the IP1 and IP0 bits, are read-only; IP1 and IP0 are used for software interrupts. Figure 2-53 shows the fields of this register; Table 2-35 describes the Cause register codes.

Figure 2-53. Cause Register Format

31	30	29 28	27 16	15 8	7	6 2	1 0
BD	0	CE	0	IP(7:0)	0	ExcCode	0
1	1	2	12	8	1	5	2

BD : Indicates whether the most recent exception occurred in the branch delay slot (1 → In delay slot, 0 → Normal).

CE : Indicates the coprocessor number in which a Coprocessor Unusable exception occurred. This field will remain undefined for as long as no exception occurs.

IP : Indicates whether an interrupt is pending (1 → Interrupt pending, 0 → No interrupt pending).

IP7 : A timer interrupt.

IP(6:2) : Ordinary interrupts (Int(4:0)[Note]). However, Int4[Note] never occurs in the VR4120A CPU.

IP(1:0) : Software interrupts. Only these bits cause an interrupt exception, when they are set to 1 by means of software.

Note Int (4:0) are internal signals of the CPU core. For details about connection to the on-chip peripheral units.

ExcCode: Exception code field (refer to Table 2-35 for details).

0 : RFU. Write 0 in a write operation. When this field is read, 0 is read.

Table 2-35. Cause Register Exception Code Field

Exception Code	Mnemonic	Description
0	Int	Interrupt exception
1	Mod	TLB Modified exception
2	TLBL	TLB Refill exception (load or fetch)
3	TLBS	TLB Refill exception (store)
4	AdEL	Address Error exception (load or fetch)
5	AdES	Address Error exception (store)
6	IBE	Bus Error exception (instruction fetch)
7	DBE	Bus Error exception (data load or store)
8	Sys	System Call exception
9	Bp	Breakpoint exception
10	RI	Reserved Instruction exception
11	CpU	Coprocessor Unusable exception
12	Ov	Integer Overflow exception
13	Tr	Trap exception
14 to 22	—	RFU
23	WATCH	Watch exception
24 to 31	—	RFU

The VR4120A CPU has eight interrupt request sources, IP7 to IP0. For the detailed description of interrupts, refer to **Section 2.8 CPU Core Interrupts**.

(1) IP7

This bit indicates whether there is a timer interrupt request.

It is set when the values of Count register and Compare register match.

(2) IP6 to IP2

IP6 to IP2 reflect the state of the interrupt request signal of the CPU core.

(3) IP1 and IP0

These bits are used to set/clear a software interrupt request.

2.5.3.7 Exception program counter (EPC) register (14)

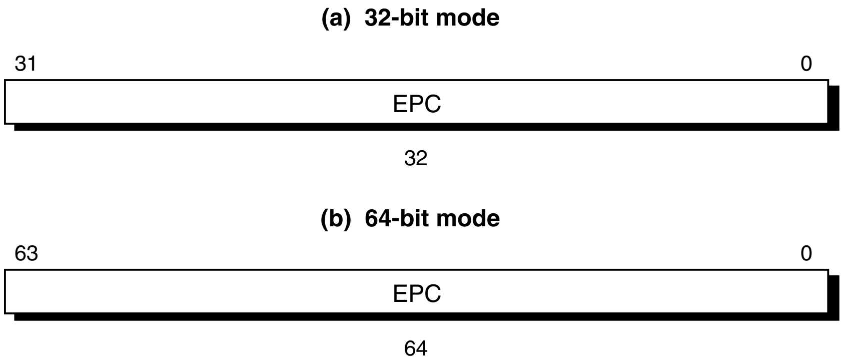
The Exception Program Counter (EPC) is a read/write register that contains the address at which processing resumes after an exception has been serviced. Because the μPD98502 does not support the MIPS16 instruction mode, the EPC register contains either:

- Virtual address of the instruction that caused the exception.
- Virtual address of the immediately preceding branch or jump instruction (when the instruction associated with the exception is in a branch delay slot, and the BD bit in the Cause register is set to 1).

The EXL bit in the Status register is set to 1 to keep the processor from overwriting the address of the exception-causing instruction contained in the EPC register in the event of another exception.

Figure 2-54 shows the EPC register format.

Figure 2-54. EPC Register Format



(a) 32-bit mode

(b) 64-bit mode

EPC: Restart address after exception processing.

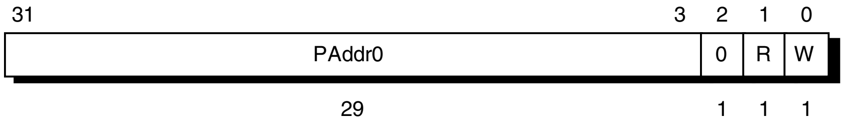
2.5.3.8 WatchLo (18) and WatchHi (19) registers

The VR4120A processor provides a debugging feature to detect references to a selected physical address; load and store instructions to the location specified by the WatchLo and WatchHi registers cause a Watch exception.

Figures 2-55 and 2-56 show the format of the WatchLo and WatchHi registers.

Figure 2-55. WatchLo Register Format

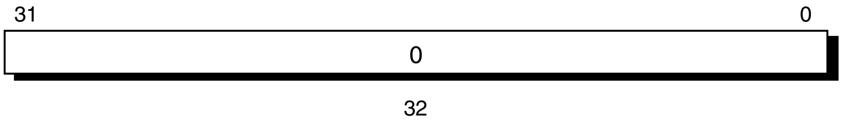
WatchLo Register



PAddr0 : Specifies physical address bits 31 to 3.
R : If this bit is set to 1, an exception will occur when a load instruction is executed.
W : If this bit is set to 1, an exception will occur when a store instruction is executed.
0 : RFU. Write 0 in a write operation. When this field is read, 0 is read.

Figure 2-56. WatchHi Register Format

WatchHi Register



0 : RFU. Write 0 in a write operation. When this field is read, 0 is read.

2.5.3.9 XContext register (20)

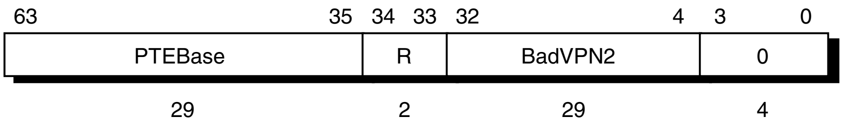
The read/write XContext register contains a pointer to an entry in the page table entry (PTE) array, an operating system data structure that stores virtual-to-physical address translations. If a TLB miss occurs, the operating system loads the untranslated data from the PTE into the TLB to handle the software error.

The XContext register is used by the XTLB Refill exception handler to load TLB entries in 64-bit addressing mode.

The XContext register duplicates some of the information provided in the BadVAddr register, and puts it in a form useful for the XTLB exception handler.

This register is included solely for operating system use. The operating system sets the PTEBase field in the register, as needed. Figure 2-57 shows the format of the XContext register.

Figure 2-57. XContext Register Format



PTEBase : The PTEBase field is a base address of the PTE entry table.
R : Space type (00 → User, 01→ Supervisor, 11 → Kernel). The setting of this field matches virtual address bits 63 and 62.
BadVPN2 : This field holds the value (VPN2) obtained by halving the virtual page number of the most recent virtual address for which translation failed.
0 : RFU. Write 0 in a write operation. When this field is read, 0 is read.

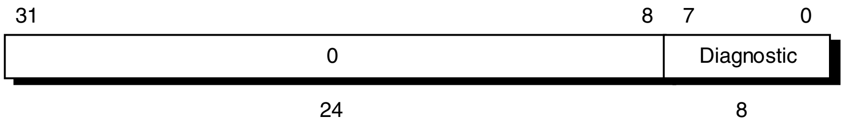
The 29-bit BadVPN2 field has bits 39 to 11 of the virtual address that caused the TLB miss; bit 10 is excluded because a single TLB entry maps to an even-odd page pair. For a 1-Kbyte page size, this format may be used directly to address the pair-table of 8-byte PTEs. For 4-Kbyte or more page and PTE sizes, shifting or masking this value produces the appropriate address.

2.5.3.10 Parity error register (26)

The Parity Error (PErr) register is a readable/writeable register. This register is defined to maintain software-compatibility with the VR4100, and is not used in hardware because the VR4120A CPU has no parity.

Figure 2-58 shows the format of the PErr register.

Figure 2-58. Parity Error Register Format



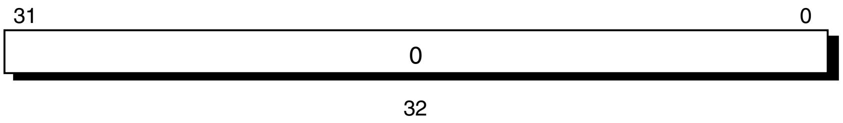
Diagnostic : 8-bit self diagnostic field.
0 : RFU. Write 0 in a write operation. When this field is read, 0 is read.

2.5.3.11 Cache error register (27)

The Cache Error register is a readable/writeable register. This register is defined to maintain software-compatibility with the VR4100, and is not used in hardware because the VR4120A CPU has no parity.

Figure 2-59 shows the format of the Cache Error register.

Figure 2-59. Cache Error Register Format



0 : RFU. Write 0 in a write operation. When this field is read, 0 is read.

2.5.3.12 ErrorEPC register (30)

The Error Exception Program Counter (ErrorEPC) register is similar to the EPC register. It is used to store the Program Counter value at which the Cache Error, Cold Reset, Soft Reset, or NMI exception has been serviced.

The read/write ErrorEPC register contains the virtual address at which instruction processing can resume after servicing an error.

This address can be:

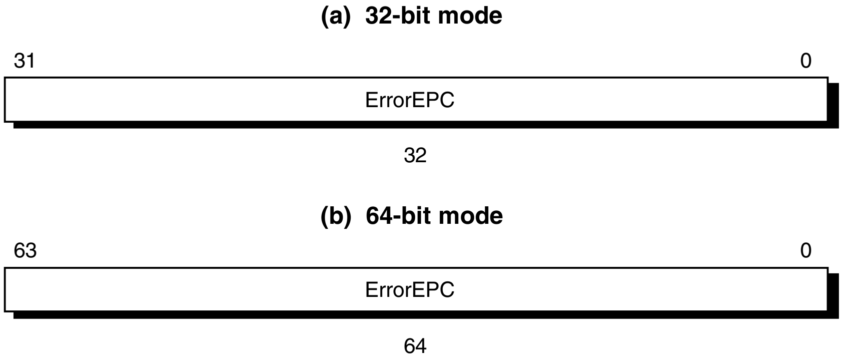
- Virtual address of the instruction that caused the exception.
- Virtual address of the immediately preceding branch or jump instruction, when the instruction associated with the error exception is in a branch delay slot.

The contents of the ErrorEPC register do not change when the ERL bit of the Status register is set to 1. This prevents the processor when other exceptions occur from overwriting the address of the instruction in this register which causes an error exception.

There is no branch delay slot indication for the ErrorEPC register.

Figure 2-60 shows the format of the ErrorEPC register.

Figure 2-60. ErrorEPC Register Format


(a) 32-bit mode

(b) 64-bit mode

ErrorEPC: Program counter that indicates the restart address after Cold reset, Soft reset, or NMI exception.

2.5.4 Details of exceptions

This section describes causes, processes, and services of the VR4120A's exceptions.

2.5.4.1 Exception types

This section gives sample exception handler operations for the following exception types:

- Cold Reset
- Soft Reset
- NMI
- Remaining processor exceptions

When the EXL and ERL bits in the Status register are 0, either User, Supervisor, or Kernel operating mode is specified by the KSU bits in the Status register. When either the EXL or ERL bit is set to 1, the processor is in Kernel mode.

When the processor takes an exception, the EXL bit is set to 1, meaning the system is in Kernel mode. After saving the appropriate state, the exception handler typically resets the EXL bit back to 0. The exception handler sets the EXL bit to 1 so that the saved state is not lost upon the occurrence of another exception while the saved state is being restored.

Returning from an exception also resets the EXL bit to 0. For details, see **APPENDIX A MIPS III INSTRUCTION SET DETAILS**.

2.5.4.2 Exception vector address

The Cold Reset, Soft Reset, and NMI exceptions are always branched to the following reset exception vector address. This address is in an uncached, unmapped space.

- BFC0_0000H in 32-bit mode (virtual address)
- FFFF_FFFF_BFC0_0000H in 64-bit mode (virtual address)

Vector addresses for the remaining exceptions are a combination of a vector offset and a base address.

64-/32-bit mode exception vectors and their offsets are shown below.

Table 2-36. 64-Bit Mode Exception Vector Base Addresses

	Vector Base Address (Virtual)	Vector Offset
Cold Reset Soft Reset NMI	FFFF_FFFF_BFC0_0000H (BEV bit is automatically set to 1)	0000H
TLB Refill (EXL = 0)	FFFF_FFFF_8000_0000H (BEV = 0) FFFF_FFFF_BFC0_0200H (BEV = 1)	0000H
XTLB Refill (EXL = 0)		0080H
Other exceptions		0180H

Table 2-37. 32-Bit Mode Exception Vector Base Addresses

	Vector Base Address (Virtual)	Vector Offset
Cold Reset Soft Reset NMI	BFC0_0000H (BEV bit is automatically set to 1)	0000H
TLB Refill (EXL = 0)	8000_0000H (BEV = 0) BFC0_0200H (BEV = 1)	0000H
XTLB Refill (EXL = 0)		0080H
Other exceptions		0180H

(1) TLB refill exception vector

When BEV bit = 0, the vector base address (virtual) for the TLB Refill exception is in kseg0 (unmapped) space.

✧ 8000_0000H in 32-bit mode
✧ FFFF_FFFF_8000_0000H in 64-bit mode

When BEV bit = 1, the vector base address (virtual) for the TLB Refill exception is in kseg1 (uncached, unmapped) space.

✧ BFC0_0200H in 32-bit mode
✧ FFFF_FFFF_BFC0_0200H in 64-bit mode

This is an uncached, non-TLB-mapped space, allowing the exception handler to bypass the cache and TLB.

2.5.4.3 Priority of exceptions

While more than one exception can occur for a single instruction, only the exception with the highest priority is reported. Table 2-38 lists the priorities.

Table 2-38. Exception Priority Order

High	Cold Reset
↑	Soft Reset
│	NMI
│	Address Error (instruction fetch)
│	TLB/XTLB Refill (instruction fetch)
│	TLB Invalid (instruction fetch)
│	Bus Error (instruction fetch)
│	System Call
│	Breakpoint
│	Coprocessor Unusable
│	Reserved Instruction
│	Trap
│	Integer Overflow
│	Address Error (data access)
│	TLB/XTLB Refill (data access)
│	TLB Invalid (data access)
│	TLB Modified (data write)
│	Watch
↓	Bus Error (data access)
Low	Interrupt (other than NMI)

Hereafter, handling exceptions by hardware is referred to as "process", and handling exception by software is referred to as "service".

2.5.4.4 Cold reset exception

(1) Cause

The Cold Reset exception occurs when the ColdReset_B signal (internal) is asserted and then deasserted. This exception is not maskable. The Reset_B signal (internal) must be asserted along with the ColdReset_B signal (for details, see **Section 2.6 Initialization Interface**).

(2) Processing

The CPU provides a special interrupt vector for this exception:

- ✧ BFC0_0000H (virtual address) in 32-bit mode
- ✧ FFFF_FFFF_BFC0_0000H (virtual address) in 64-bit mode

The Cold Reset vector resides in unmapped and uncached CPU address space, so the hardware need not initialize the TLB or the cache to process this exception. It also means the processor can fetch and execute instructions while the caches and virtual memory are in an undefined state.

The contents of all registers in the CPU are undefined when this exception occurs, except for the following register fields:

- While the ERL of Status register is 0, the PC value at which an exception occurs is set to the ErrorEPC register.
- TS and SR bits of the Status register are cleared to 0.
- ERL and BEV bits of the Status register are set to 1.
- The Random register is initialized to the value of its upper bound (31).
- The Wired register is initialized to 0.
- Bits 31 to 28 and bits 22 to 3 of the Config register are set to fixed values.
- All other bits are undefined.

(3) Servicing

The Cold Reset exception is serviced by:

- Initializing all processor registers, coprocessor registers, TLB, caches, and the memory system
- Performing diagnostic tests
- Bootstrapping the operating system

2.5.4.5 Soft reset exception

(1) Cause

A Soft Reset (sometimes called Warm Reset) occurs when the ColdReset_B signal (internal) remains deasserted while the Reset_B signal (internal) goes from assertion to deassertion (for details, see **Section 2.6 Initialization Interface**).

A Soft Reset immediately resets all state machines, and sets the SR bit of the Status register. Execution begins at the reset vector when the reset is deasserted.

This exception is not maskable.

Caution In the μPD98502, a soft reset never occurs.

(2) Processing

The CPU provides a special interrupt vector for this exception (same location as Cold Reset):

✧ BFC0_0000H (virtual) in 32-bit mode
✧ FFFF_FFFF_BFC0_0000H (virtual) in 64-bit mode

This vector is located within unmapped and uncached address space, so that the cache and TLB need not be initialized to process this exception. The SR bit of the Status register is set to 1 to distinguish this exception from a Cold Reset exception.

When this exception occurs, the contents of all registers are preserved except for the following registers:

- The PC value at which an exception occurs is set to the ErrorEPC register.
- TS bit of the Status register is cleared to 0.
- ERL, SR, and BEV bits of the Status register are set to 1.

During a soft reset, access to the operating cache or system interface may be aborted. This means that the contents of the cache and memory will be undefined if a Soft Reset occurs.

(3) Servicing

The Soft Reset exception is serviced by:

- Preserving the current processor states for diagnostic tests
- Reinitializing the system in the same way as for a Cold Reset exception

2.5.4.6 NMI exception

(1) Cause

The Nonmaskable Interrupt (NMI) exception occurs when the NMI signal (internal) becomes active. This interrupt is not maskable; it occurs regardless of the settings of the EXL, ERL, and IE bits in the Status register (for details, see **Section 2.8 CPU Core Interrupts**).

(2) Processing

The CPU provides a special interrupt vector for this exception:

✧ BFC0_0000H (virtual) in 32-bit mode
✧ FFFF_FFFF_BFC0_0000H (virtual) in 64-bit mode

This vector is located within unmapped and uncached address space so that the cache and TLB need not be initialized to process an NMI exception. The SR bit of the Status register is set to 1 to distinguish this exception from a Cold Reset exception.

Unlike Cold Reset and Soft Reset, but like other exceptions, NMI is taken only at instruction boundaries. The states of the caches and memory system are preserved by this exception.

When this exception occurs, the contents of all registers are preserved except for the following registers:

- The PC value at which an exception occurs is set to the ErrorEPC register.
- The TS bit of the Status register is cleared to 0.
- The ERL, SR, and BEV bits of the Status register are set to 1.

(3) Servicing

The NMI exception is serviced by:

- Preserving the current processor states for diagnostic tests
- Reinitializing the system in the same way as for a Cold Reset exception

2.5.4.7 Address error exception

(1) Cause

The Address Error exception occurs when an attempt is made to execute one of the following. This exception is not maskable.

- Execution of the LW, LWU, SW, or CACHE instruction for word data that is not located on a word boundary
- Execution of the LH, LHU, or SH instruction for half-word data that is not located on a half-word boundary
- Execution of the LD or SD instruction for double-word data that is not located on a double-word boundary
- Referencing the kernel address space in User or Supervisor mode
- Referencing the supervisor space in User mode
- Referencing an address that does not exist in the kernel, user, or supervisor address space in 64-bit Kernel, User, or Supervisor mode
- Branching to an address that was not located on a word boundary.

(2) Processing

The common exception vector is used for this exception. The AdEL or AdES code in the Cause register is set. If this exception has been caused by an instruction reference or load operation, AdEL is set. If it has been caused by a store operation, AdES is set.

When this exception occurs, the BadVAddr register stores the virtual address that was not properly aligned or was referenced in protected address space. The contents of the VPN field of the Context and EntryHi registers are undefined, as are the contents of the EntryLo register.

The EPC register contains the address of the instruction that caused the exception. However, if this instruction is in a branch delay slot, the EPC register contains the address of the preceding jump or branch instruction, and the BD bit of the Cause register is set to 1.

(3) Servicing

The kernel reports the UNIX™ SIGSEGV (segmentation violation) signal to the current process, and this exception is usually fatal.

2.5.4.8 TLB exceptions

Three types of TLB exceptions can occur:

- TLB Refill exception occurs when there is no TLB entry that matches a referenced address.
- A TLB Invalid exception occurs when a TLB entry that matches a referenced virtual address is marked as being invalid (with the V bit set to 0).
- The TLB Modified exception occurs when a TLB entry that matches a virtual address referenced by the store instruction is marked as being valid (with the V bit set to 1).

The following three sections describe these TLB exceptions.

(1) TLB refill exception (32-bit space mode)/XTLB refill exception (64-bit space mode)

(a) Cause

The TLB Refill exception occurs when there is no TLB entry to match a reference to a mapped address space. This exception is not maskable.

(b) Processing

There are two special exception vectors for this exception; one for references to 32-bit address spaces, and one for references to 64-bit address spaces. The UX, SX, and KX bits of the Status register determine whether the user, supervisor or kernel address spaces referenced are 32-bit or 64-bit spaces. When the EXL bit of the Status register is set to 0, either of these two special vectors is referenced. When the EXL bit is set to 1, the common exception vector is referenced.

This exception sets the TLBL or TLBS code in the ExcCode field of the Cause register. If this exception has been caused by an instruction reference or load operation, TLBL is set. If it has been caused by a store operation, TLBS is set.

When this exception occurs, the BadVAddr, Context, XContext and EntryHi registers hold the virtual address that failed address translation. The EntryHi register also contains the ASID from which the translation fault occurred. The Random register normally contains a valid location in which to place the replacement TLB entry. The contents of the EntryLo register are undefined.

The EPC register contains the address of the instruction that caused the exception. However, if this instruction is in a branch delay slot, the EPC register contains the address of the preceding jump or branch instruction, and the BD bit of the Cause register is set to 1.

(c) Servicing

To service this exception, the contents of the Context or XContext register are used as a virtual address to fetch memory words containing the physical page frame and access control bits for a pair of TLB entries. The memory word is written into the TLB entry by using the EntryLo0, EntryLo1, or EntryHi register.

It is possible that the physical page frame and access control bits are placed in a page where the virtual address is not resident in the TLB. This condition is processed by allowing a TLB Refill exception in the TLB Refill exception handler. In this case, the common exception vector is used because the EXL bit of the Status register is set to 1.

(2) TLB invalid exception

(a) Cause

The TLB Invalid exception occurs when the TLB entry that matches with the virtual address to be referenced is invalid (the V bit is set to 0). This exception is not maskable.

(b) Processing

The common exception vector is used for this exception. The TLBL or TLBS code in the ExcCode field of the Cause register is set. If this exception has been caused by an instruction reference or load operation, TLBL is set. If it has been caused by a store operation, TLBS is set.

When this exception occurs, the BadVAddr, Context, Xcontext, and EntryHi registers contain the virtual address that failed address translation. The EntryHi register also contains the ASID from which the translation fault occurred. The Random register normally stores a valid location in which to place the replacement TLB entry. The contents of the EntryLo register are undefined.

The EPC register contains the address of the instruction that caused the exception. However, if this instruction is in a branch delay slot, the EPC register contains the address of the preceding jump or branch instruction, and the BD bit of the Cause register is set to 1.

(c) Servicing

Usually, the V bit of a TLB entry is cleared in the following cases:

- When a virtual address does not exist
- When the virtual address exists, but is not in main memory (a page fault)
- When a trap is required on any reference to the page (for example, to maintain a reference bit)

After servicing the cause of a TLB Invalid exception, the TLB entry is located with a TLBP (TLB Probe) instruction, and replaced by an entry with its V bit set to 1.

(3) TLB modified exception

(a) Cause

The TLB Modified exception occurs when the TLB entry that matches with the virtual address referenced by the store instruction is valid (V bit is 1) but is not writeable (D bit is 0). This exception is not maskable.

(b) Processing

The common exception vector is used for this exception, and the Mod code in the ExcCode field of the Cause register is set.

When this exception occurs, the BadVAddr, Context, Xcontext, and EntryHi registers contain the virtual address that failed address translation. The EntryHi register also contains the ASID from which the translation fault occurred. The contents of the EntryLo register are undefined.

The EPC register contains the address of the instruction that caused the exception. However, if this instruction is in a branch delay slot, the EPC register contains the address of the preceding jump or branch instruction, and the BD bit of the Cause register is set to 1.

(c) Servicing

The kernel uses the failed virtual address or virtual page number to identify the corresponding access control bits. The page identified may or may not permit write accesses; if writes are not permitted, a write protection violation occurs.

If write accesses are permitted, the page frame is marked dirty (/writeable) by the kernel in its own data structures. The TLBP instruction places the index of the TLB entry that must be altered into the Index register. The word data containing the physical page frame and access control bits (with the D bit set to 1) is loaded to the EntryLo register, and the contents of the EntryHi and EntryLo registers are written into the TLB.

2.5.4.9 Bus error exception

(1) Cause

A Bus Error exception is raised by board-level circuitry for events such as bus time-out, local bus parity errors, and invalid physical memory addresses or access types. This exception is not maskable.

A Bus Error exception occurs only when a cache miss refill, uncached reference, or unbuffered write occurs simultaneously. In other words, it occurs when an illegal access is detected during BCU read.

For details of illegal accesses.

(2) Processing

The common interrupt vector is used for a Bus Error exception. The IBE or DBE code in the ExcCode field of the Cause register is set, signifying whether the instruction caused the exception by an instruction reference, load operation, or store operation.

The EPC register contains the address of the instruction that caused the exception. However, if this instruction is in a branch delay slot, the EPC register contains the address of the preceding jump or branch instruction, and the BD bit of the Cause register is set to 1.

(3) Servicing

The physical address at which the fault occurred can be computed from information available in the System Control Coprocessor (CP0) registers.

- If the IBE code in the Cause register is set (indicating an instruction fetch), the virtual address is contained in the EPC register.
- If the DBE code is set (indicating a load or store), the virtual address of the instruction that caused the exception is saved to the EPC register.

The virtual address of the load and store instruction can then be obtained by interpreting the instruction. The physical address can be obtained by using the TLBP instruction and reading the EntryLo register to compute the physical page number.

At the time of this exception, the kernel reports the UNIX SIGBUS (bus error) signal to the current process, but the exception is usually fatal.

2.5.4.10 System call exception

(1) Cause

A System Call exception occurs during an attempt to execute the SYSCALL instruction. This exception is not maskable.

(2) Processing

The common exception vector is used for this exception, and the Sys code in the ExcCode field of the Cause register is set.

The EPC register contains the address of the SYSCALL instruction unless it is in a branch delay slot, in which case the EPC register contains the address of the preceding branch instruction.

If the SYSCALL instruction is in a branch delay slot, the BD bit of the Status register is set to 1; otherwise this bit is cleared.

(3) Servicing

When this exception occurs, control is transferred to the applicable system routine.

To resume execution, the EPC register must be altered so that the SYSCALL instruction does not re-execute; this is accomplished by adding a value of 4 to the EPC register before returning.

If a SYSCALL instruction is in a branch delay slot, interpretation (decoding) of the branch instruction is required to resume execution.

2.5.4.11 Breakpoint exception

(1) Cause

A Breakpoint exception occurs when an attempt is made to execute the BREAK instruction. This exception is not maskable.

(2) Processing

The common exception vector is used for this exception, and the Bp code in the ExcCode field of the Cause register is set.

The EPC register contains the address of the instruction that caused the exception. However, if this instruction is in a branch delay slot, the EPC register contains the address of the preceding jump or branch instruction, and the BD bit of the Cause register is set to 1.

(3) Servicing

When the Breakpoint exception occurs, control is transferred to the applicable system routine. Additional distinctions can be made by analyzing the unused bits of the BREAK instruction (bits 25 to 6), and loading the contents of the instruction whose address the EPC register contains.

To resume execution, the EPC register must be altered so that the BREAK instruction does not re-execute; this is accomplished by adding a value of 4 to the EPC register before returning.

If a BREAK instruction is in a branch delay slot, interpretation (decoding) of the branch instruction is required to resume execution.

2.5.4.12 Coprocessor unusable exception

(1) Cause

The Coprocessor Unusable exception occurs when an attempt is made to execute a coprocessor instruction for either:

- ✧ a corresponding coprocessor unit that has not been marked usable (Status register bit, CU0 = 0), or
- ✧ CP0 instructions, when the unit has not been marked usable (Status register bit, CU0 = 0) and the process executes in User or Supervisor mode.

This exception is not maskable.

(2) Processing

The common exception vector is used for this exception, and the CPU code in the ExcCode field of the Cause register is set. The CE bit of the Cause register indicates which of the four coprocessors was referenced.
The EPC register contains the address of the instruction that caused the exception. However, if this instruction is in a branch delay slot, the EPC register contains the address of the preceding jump or branch instruction, and the BD bit of the Cause register is set to 1.

(3) Servicing

The coprocessor unit to which an attempted reference was made is identified by the CE bit of the Cause register.
One of the following processing is performed by the handler:

- ✧ If the process is entitled access to the coprocessor, the coprocessor is marked usable and the corresponding state is restored to the coprocessor.
- ✧ If the process is entitled access to the coprocessor, but the coprocessor does not exist or has failed, interpretation of the coprocessor instruction is possible.
- ✧ If the BD bit in the Cause register is set to 1, the branch instruction must be interpreted; then the coprocessor instruction can be emulated and execution resumed with the EPC register advanced past the coprocessor instruction.
- ✧ If the process is not entitled access to the coprocessor, the kernel reports UNIX SIGILL/ILL_PRIVIN_FAULT (illegal instruction/privileged instruction fault) signal to the current process, and this exception is fatal.

2.5.4.13 Reserved instruction exception

(1) Cause

The Reserved Instruction exception occurs when an attempt is made to execute one of the following instructions:

- Instruction with an undefined major opcode (bits 31 to 26)
- SPECIAL instruction with an undefined minor opcode (bits 5 to 0)
- REGIMM instruction with an undefined minor opcode (bits 20 to 16)
- 64-bit instructions in 32-bit User or Supervisor mode
- RR instruction with an undefined minor op code (bits 4 to 0)

64-bit operations are always valid in Kernel mode regardless of the value of the KX bit in the Status register. This exception is not maskable.

(2) Processing

The common exception vector is used for this exception, and the RI code in the ExcCode field of the Cause register is set.

The EPC register contains the address of the instruction that caused the exception. However, if this instruction is in a branch delay slot, the EPC register contains the address of the preceding jump or branch instruction, and the BD bit of the Cause register is set to 1.

(3) Servicing

All currently defined MIPS ISA instructions can be executed. The process executing at the time of this exception is handled by a UNIX SIGILL/ILL_RESOP_FAULT (illegal instruction/reserved operand fault) signal. This error is usually fatal.

2.5.4.14 Trap exception

(1) Cause

The Trap exception occurs when a TGE, TGEU, TLT, TLTU, TEQ, TNE, TGEI, TGEUI, TLTI, TLTUI, TEQI, or TNEI instruction results in a TRUE condition. This exception is not maskable.

(2) Processing

The common exception vector is used for this exception, and the Tr code in the ExcCode field of the Cause register is set.

The EPC register contains the address of the trap instruction causing the exception unless the instruction is in a branch delay slot, in which case the EPC register contains the address of the preceding branch instruction and the BD bit of the Cause register is set to 1.

(3) Servicing

At the time of a Trap exception, the kernel reports the UNIX SIGFPE/FPE_INTOVF_TRAP (floating-point exception/integer overflow) signal to the current process, but the exception is usually fatal.

2.5.4.15 Integer overflow exception

(1) Cause

An Integer Overflow exception occurs when an ADD, ADDI, SUB, DADD, DADDI, or DSUB instruction results in a 2’s complement overflow. This exception is not maskable.

(2) Processing

The common exception vector is used for this exception, and the Ov code in the ExcCode field of the Cause register is set.

The EPC register contains the address of the instruction that caused the exception unless the instruction is in a branch delay slot, in which case the EPC register contains the address of the preceding branch instruction and the BD bit of the Cause register is set to 1.

(3) Servicing

At the time of the exception, the kernel reports the UNIX SIGFPE/FPE_INTOVF_TRAP (floating-point exception/integer overflow) signal to the current process, and this exception is usually fatal.

2.5.4.16 Watch exception

(1) Cause

A Watch exception occurs when a load or store instruction references the physical address specified by the WatchLo/WatchHi registers. The WatchLo/WatchHi registers specify whether a load or store or both could have initiated this exception.

- When the R bit of the WatchLo register is set to 1: Load instruction
- When the W bit of the WatchLo register is set to 1: Store instruction
- When both the R bit and W bit of the WatchLo register are set to 1: Load instruction or store instruction

The CACHE instruction never causes a Watch exception.

The Watch exception is postponed while the EXL bit in the Status register is set to 1, and Watch exception is only maskable by setting the EXL bit in the Status register to 1.

(2) Processing

The common exception vector is used for this exception, and the Watch code in the ExcCode field of the Cause register is set.

The EPC register contains the address of the instruction that caused the exception. However, if this instruction is in a branch delay slot, the EPC register contains the address of the preceding jump or branch instruction, and the BD bit of the Cause register is set to 1.

(3) Servicing

The Watch exception is a debugging aid; typically the exception handler transfers control to a debugger, allowing the user to examine the situation. To continue, once the Watch exception must be disabled to execute the faulting instruction. The Watch exception must then be reenabled. The faulting instruction can be executed either by the debugger or by setting breakpoints.

2.5.4.17 Interrupt exception

(1) Cause

The Interrupt exception occurs when one of the eight interrupt conditions[Note] is asserted. In the VR4120A CPU, interrupt requests from internal peripheral units first enter the ICU and are then notified to the CPU core via one of four interrupt sources (Int(3:0)) or NMI.

Each of the eight interrupts can be masked by clearing the corresponding bit in the IM field of the Status register, and all of the eight interrupts can be masked at once by clearing the IE bit of the Status register or setting the EXL/ERL bit.

Note They are 1 timer interrupt, 5 ordinary interrupts, and 2 software interrupts.

Of the five ordinary interrupts, Int4 is never asserted active in the VR4120A CPU.

(2) Processing

The common exception vector is used for this exception, and the Int code in the ExcCode field of the Cause register is set.

The IP field of the Cause register indicates current interrupt requests. It is possible that more than one of the bits can be simultaneously set (or cleared) if the interrupt request signal is asserted and then deasserted before this register is read.

The EPC register contains the address of the instruction that caused the exception. However, if this instruction is in a branch delay slot, the EPC register contains the address of the preceding jump or branch instruction, and the BD bit of the Cause register is set to 1.

(3) Servicing

If the interrupt is caused by one of the two software-generated exceptions (SW0 or SW1), the interrupt condition is cleared by setting the corresponding Cause register bit to 0.

If the interrupt is caused by hardware, the interrupt condition is cleared by deactivating the corresponding interrupt request signal.

2.5.5 Exception processing and servicing flowcharts

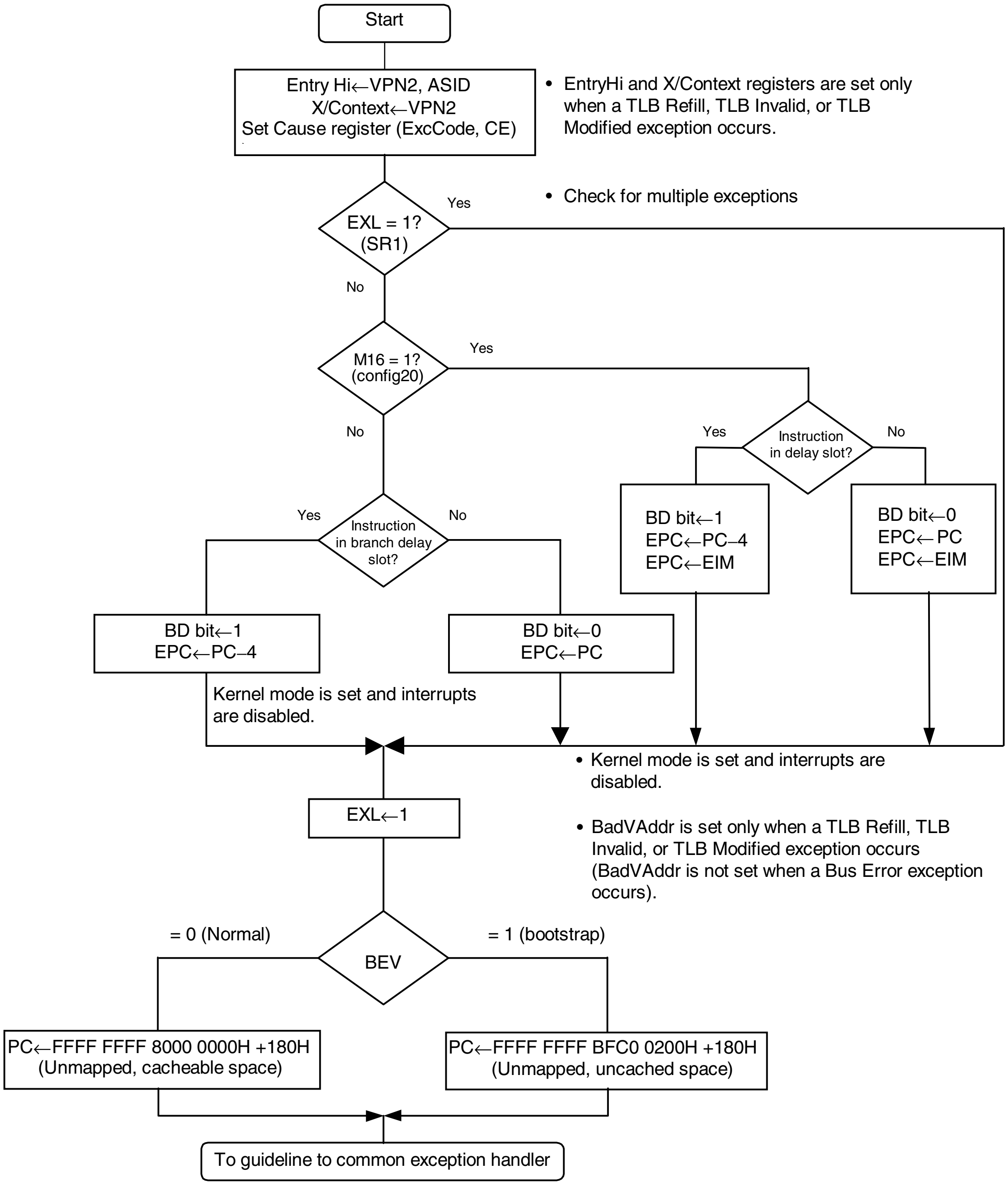
The remainder of this chapter contains flowcharts for the following exceptions and guidelines for their handlers:

- ✧ Common exceptions and a guideline to their exception handler
- ✧ TLB/XTLB Refill exception and a guideline to their exception handler
- ✧ Cold Reset, Soft Reset and NMI exceptions, and a guideline to their handler.

Generally speaking, the exceptions are "processed" by hardware (HW); the exceptions are then "serviced" by software (SW).

Figure 2-61. Common Exception Handling (1/2)

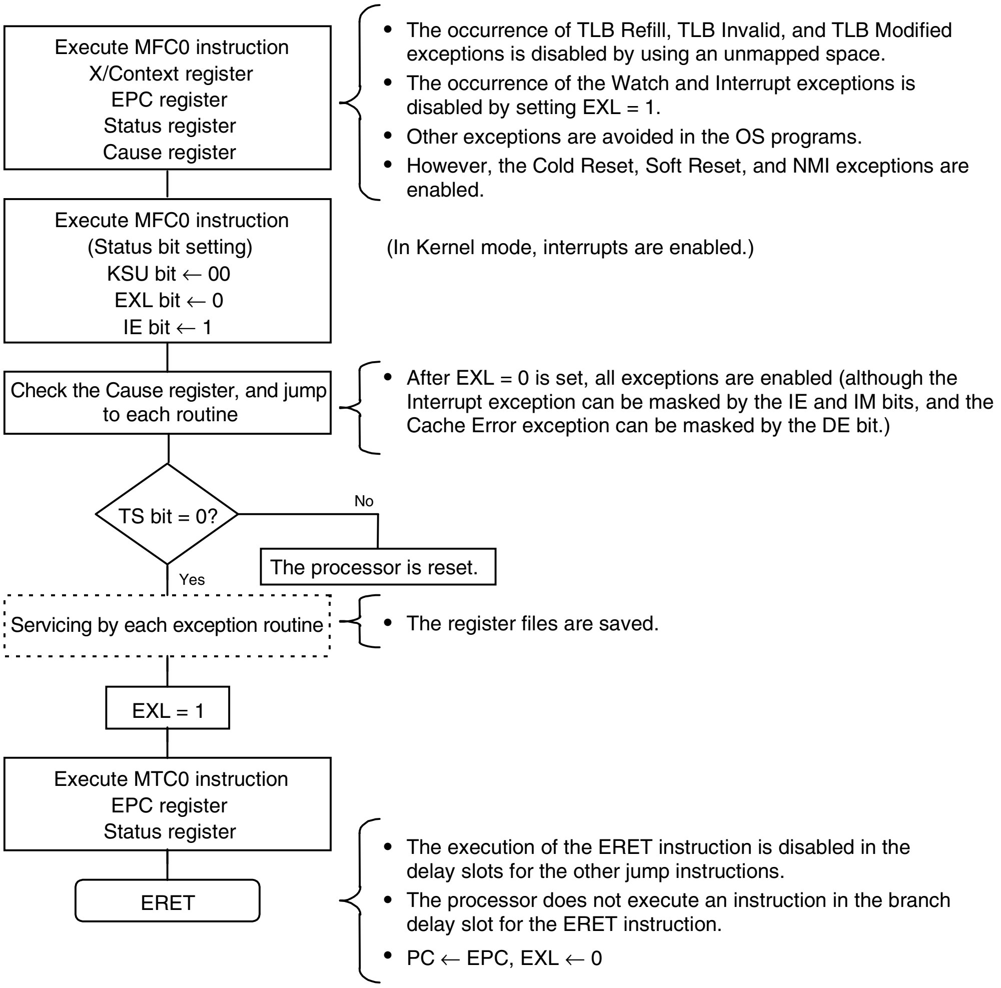
(a) Handling Exceptions other than Cold Reset, Soft Reset, NMI, and TLB/XTLB Refill (Hardware)


Start
Entry Hi←VPN2, ASID
X/Context←VPN2
Set Cause register (ExcCode, CE)
• EntryHi and X/Context registers are set only when a TLB Refill, TLB Invalid, or TLB Modified exception occurs.
• Check for multiple exceptions
EXL = 1?
(SR1)
Yes
No
M16 = 1?
(config20)
Yes
No
Instruction in delay slot?
Yes
No
BD bit←1
EPC←PC–4
EPC←EIM
BD bit←0
EPC←PC
EPC←EIM
Instruction in branch delay slot?
Yes
No
BD bit←1
EPC←PC–4
BD bit←0
EPC←PC
Kernel mode is set and interrupts are disabled.
• Kernel mode is set and interrupts are disabled.
• BadVAddr is set only when a TLB Refill, TLB Invalid, or TLB Modified exception occurs (BadVAddr is not set when a Bus Error exception occurs).
EXL←1
BEV
= 0 (Normal)
= 1 (bootstrap)
PC←FFFF FFFF 8000 0000H +180H
(Unmapped, cacheable space)
PC←FFFF FFFF BFC0 0200H +180H
(Unmapped, uncached space)
To guideline to common exception handler

Remark The interrupts can be masked by setting the IE or IM bit.
The Watch exception can be set to pending state by setting the EXL bit to 1.

Figure 2-61. Common Exception Handling (2/2)

(b) Servicing Common Exceptions (Software)



- The occurrence of TLB Refill, TLB Invalid, and TLB Modified exceptions is disabled by using an unmapped space.
- The occurrence of the Watch and Interrupt exceptions is disabled by setting EXL = 1.
- Other exceptions are avoided in the OS programs.
- However, the Cold Reset, Soft Reset, and NMI exceptions are enabled.

(In Kernel mode, interrupts are enabled.)

- After EXL = 0 is set, all exceptions are enabled (although the Interrupt exception can be masked by the IE and IM bits, and the Cache Error exception can be masked by the DE bit.)

- The register files are saved.

- The execution of the ERET instruction is disabled in the delay slots for the other jump instructions.
- The processor does not execute an instruction in the branch delay slot for the ERET instruction.
- PC ← EPC, EXL ← 0

Figure 2-62. TLB/XTLB Refill Exception Handling (1/2)

(a) Handling TLB/XTLB Refill Exceptions (Hardware)

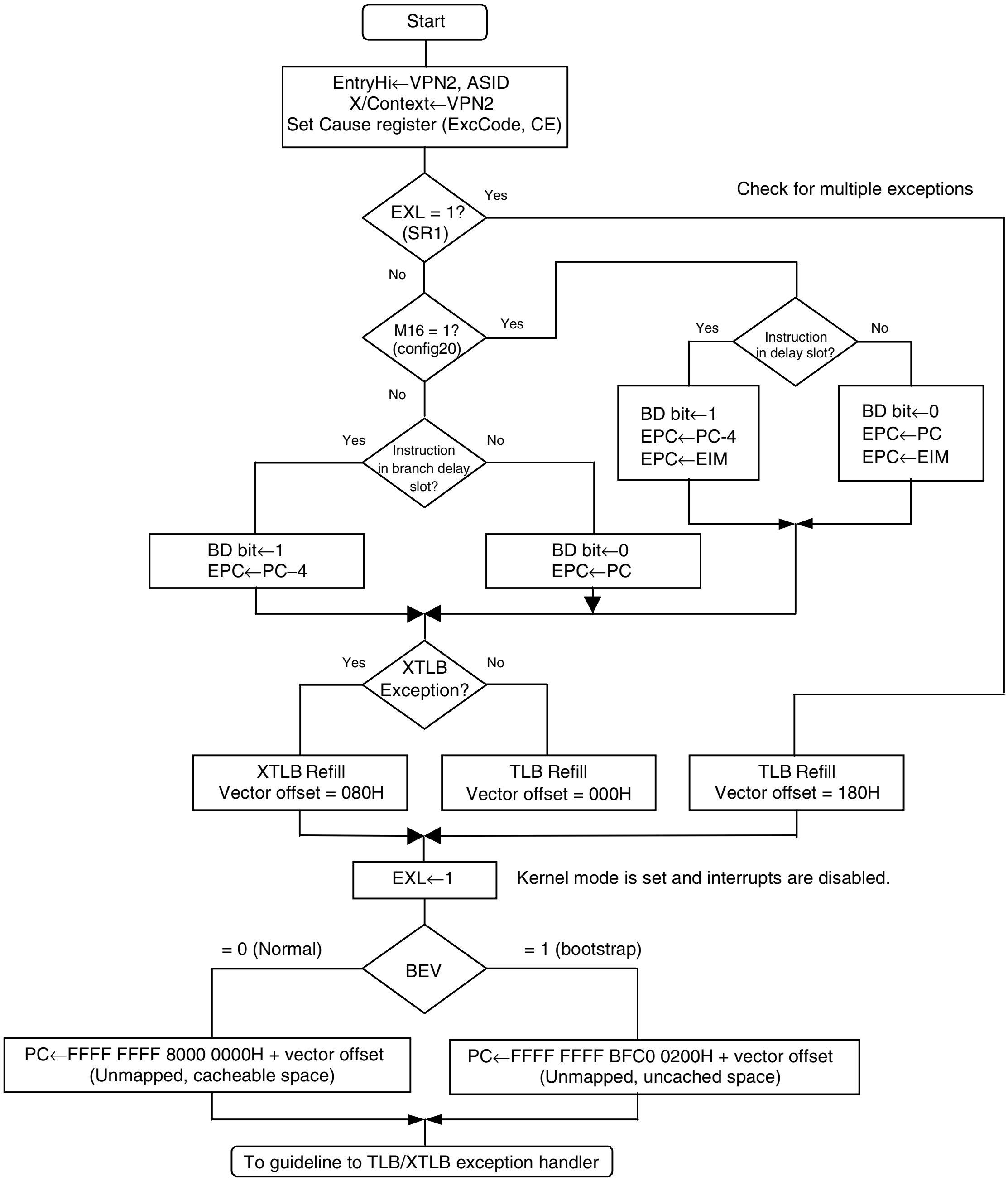
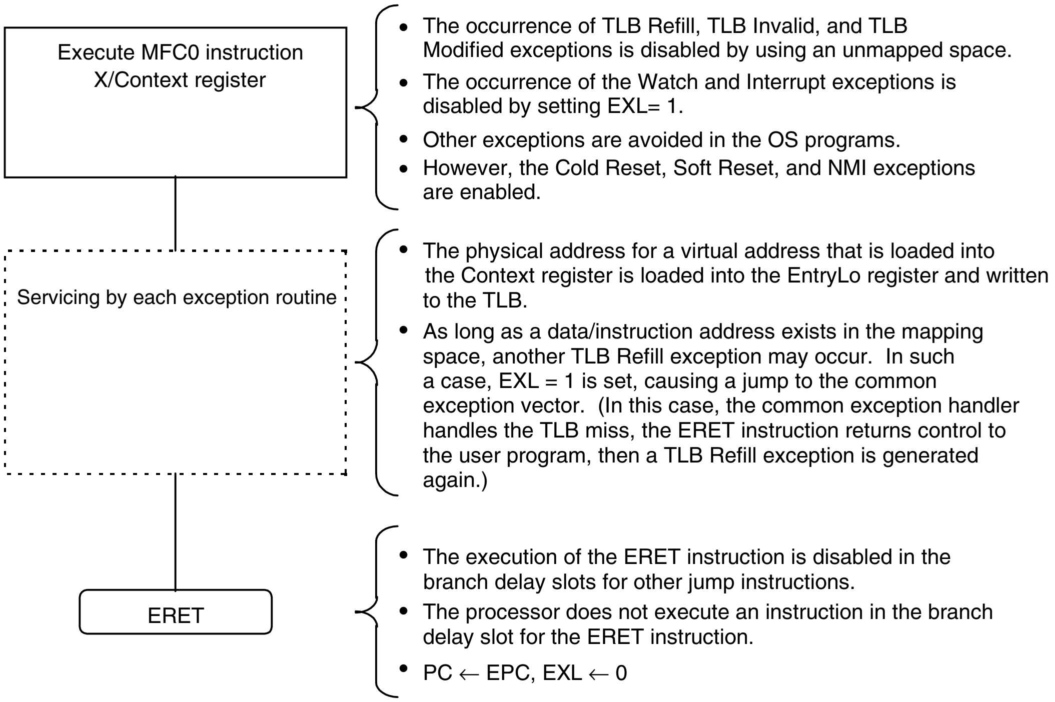


Figure 2-62. TLB/XTLB Refill Exception Handling (2/2)

(b) Servicing TLB/XTLB Refill Exceptions (Software)

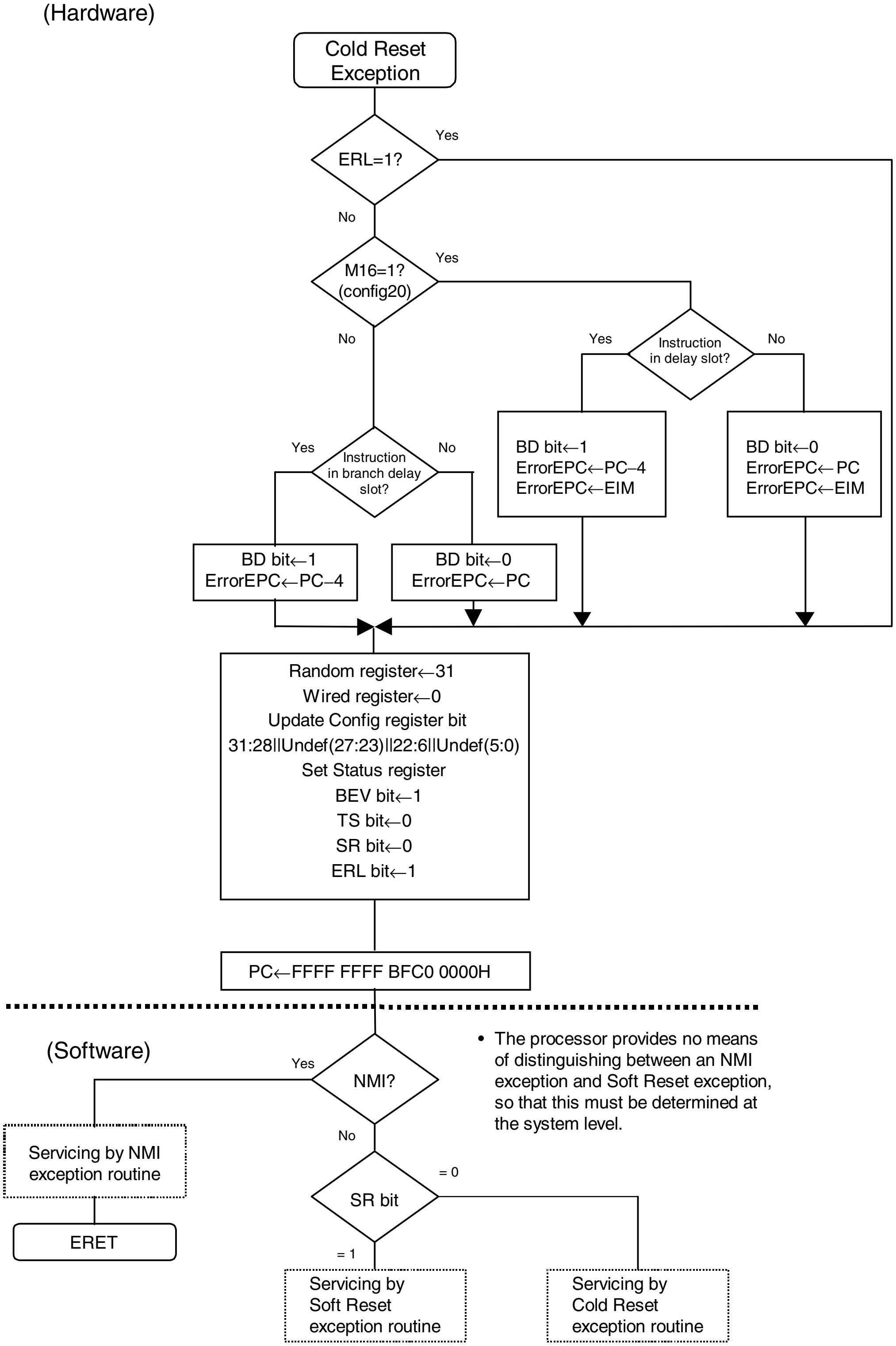


- The occurrence of TLB Refill, TLB Invalid, and TLB Modified exceptions is disabled by using an unmapped space.
- The occurrence of the Watch and Interrupt exceptions is disabled by setting EXL= 1.
- Other exceptions are avoided in the OS programs.
- However, the Cold Reset, Soft Reset, and NMI exceptions are enabled.

- The physical address for a virtual address that is loaded into the Context register is loaded into the EntryLo register and written to the TLB.
- As long as a data/instruction address exists in the mapping space, another TLB Refill exception may occur. In such a case, EXL = 1 is set, causing a jump to the common exception vector. (In this case, the common exception handler handles the TLB miss, the ERET instruction returns control to the user program, then a TLB Refill exception is generated again.)

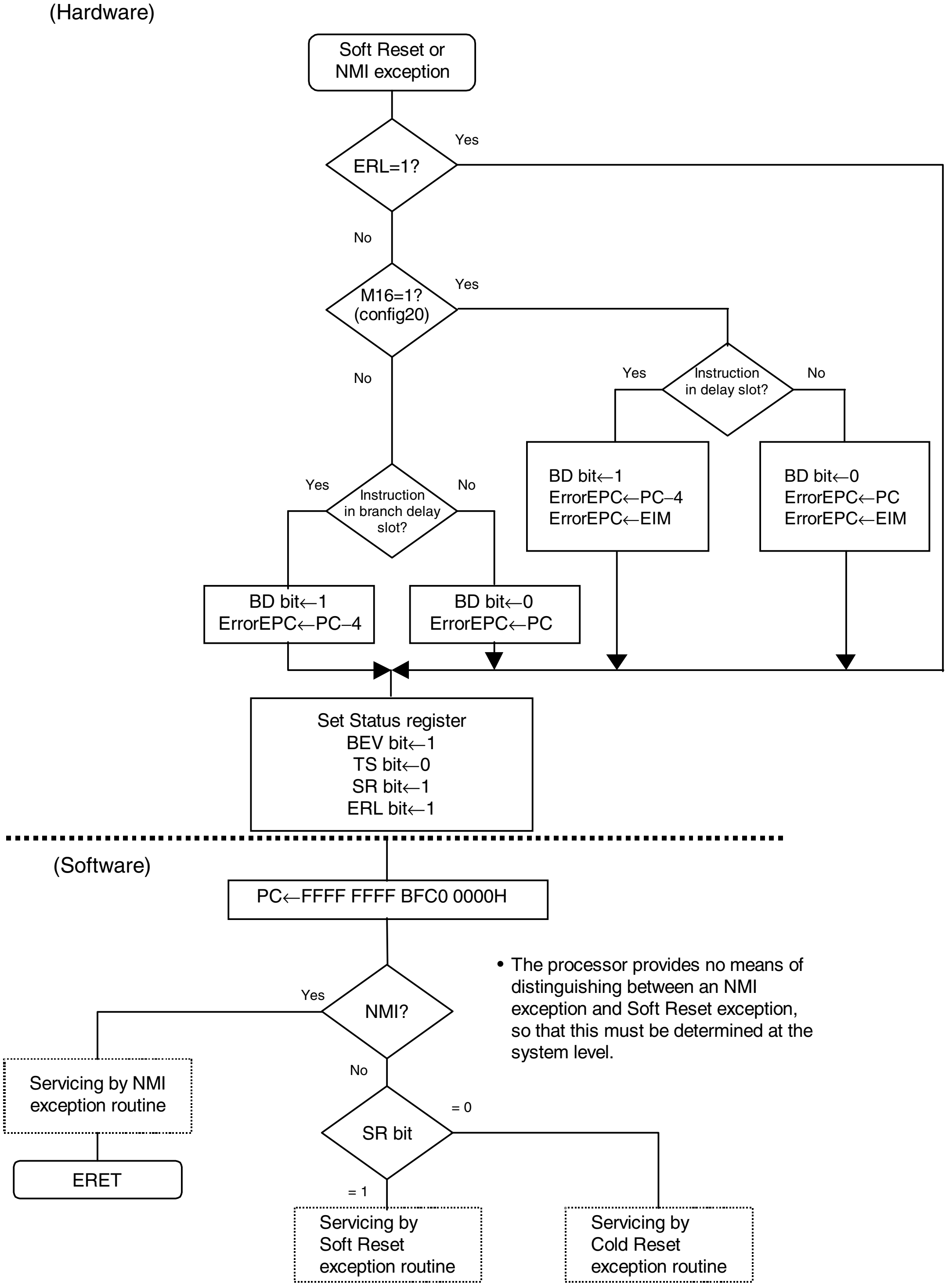
- The execution of the ERET instruction is disabled in the branch delay slots for other jump instructions.
- The processor does not execute an instruction in the branch delay slot for the ERET instruction.
- PC ← EPC, EXL ← 0

Figure 2-63. Cold Reset Exception Handling



- The processor provides no means of distinguishing between an NMI exception and Soft Reset exception, so that this must be determined at the system level.

Figure 2-64. Soft Reset and NMI Exception Handling



- The processor provides no means of distinguishing between an NMI exception and Soft Reset exception, so that this must be determined at the system level.

2.6 Initialization Interface

This section describes the reset sequence of the VR4120A Core. For details about factors of reset or reset of the whole VR4120A Core.

2.6.1 Cold reset

In the VR4120A Core, a cold reset sequence is executed in the CPU core in the following cases:

- Hardware reset
- Deadman's SW shutdown
- Software shutdown
- HAL Timer shutdown

A Cold Reset completely initializes the CPU core, except for the following register bits.

- The TS and SR bits of the Status register are cleared to 0.
- The ERL and BEV bits of the Status register are set to 1.
- The upper limit value (31) is set in the Random register.
- The Wired register is initialized to 0.
- Bits 31 to 28 of the Config register are set to 0 and bits 22 to 3 to 04800H; the other bits are undefined.
- The values of the other registers are undefined.

2.6.2 Soft reset

A Soft Reset initializes the CPU core without affecting the clocks; in other words, a Soft Reset is a logic reset. In a Soft Reset, the CPU core retains as much state information as possible; all state information except for the following is retained:

- The TS bit of the Status register is cleared to 0.
- The SR, ERL and BEV bits of the Status register are set to 1.
- The Count register is initialized to 0.
- The IP7 bit of the Cause register is cleared to 0.
- Any Interrupts generated on the SysAD bus are cleared.
- NMI is cleared.
- The Config register is initialized.

2.6.3 VR4120A processor modes

The VR4120A supports various modes, which can be selected by the user. The CPU core mode is set each time a write occurs in the Status register and Config register. The on-chip peripheral unit mode is set by writing to the I/O register.

This section describes the CPU core's operation modes. For operation modes of on-chip peripheral units, see the chapters describing the various units.

2.6.3.1 Power modes

The VR4120A supports four power modes: Fullspeed mode, Standby mode, Suspend mode, and Hibernate mode.

(1) Fullspeed mode

This is the normal operation mode.

The VR4120A's default status sets operation under Fullspeed mode. After the processor is reset, the VR4120A returns to Fullspeed mode.

(2) Standby mode

When a STANDBY instruction has been executed, the processor can be set to Standby mode. During Standby mode, all of the internal clocks in the CPU core except for the timer and interrupt clocks are held at high level. The peripheral units all operate as they do during Fullspeed mode. This means that DMA operations are enabled during Standby mode.

When the STANDBY instruction completes the WB stage, the VR4120A remains idle until the SysAD internal bus enters the idle state. Next, the clocks in the CPU core are shut down and pipeline operation is stopped. However, the PLL, timer, and interrupt clocks continue to operate, as do the internal bus clocks (TClock and MasterOut).

During Standby mode, the processor is returned to Fullspeed mode if any interrupt occurs, including a timer interrupt that occurs internally.

(3) Suspend mode

When the SUSPEND instruction has been executed, the processor can be set to Suspend mode. During Suspend mode, the processor stalls the pipeline and supplying all of the internal clocks in the CPU core except for PLL timer and interrupt clocks are stopped. The VR4120A stops supplying TClock to peripheral units. Accordingly, during Suspend mode peripheral units can only be activated by a special interrupt unit (DCD_B control, etc.). While in this mode, the register and cache contents are retained.

When the SUSPEND instruction completes the WB stage, the VR4120A switches the DRAM to self refresh mode and then waits for the SysAD internal bus to enter the idle state. Next, the clocks in the CPU core are shut down and pipeline operation is stopped. The VR4120A then stops supplying TClock to peripheral units. However, the PLL, timer, and interrupt clocks continue to operate, as do the MasterOut.

The processor remains in Suspend mode until an interrupt is received, at which time it returns to Fullspeed mode.

(4) Hibernate mode

When the HIBERNATE instruction has been executed, the processor can be set to Hibernate mode. During Hibernate mode, the processor stops supplying clocks to all units. The register and cache contents are retained and output of TClock and MasterOut is stopped. The processor remains in Hibernate mode until the POWER pin is asserted or a WakeUpTimer interrupt occurs at which the processor returns to Fullspeed mode.

In this mode, supplying voltage to the 2.5-V power-supply systems (VDD2, VDDP, VDDPD) can be stopped. When the voltage of the 2.5-V power supplies becomes 0 V, the power dissipation becomes almost 0 W (it is not exactly 0 V because there are a 32.768-kHz oscillator and on-chip peripheral circuits operating at 32.768 kHz).

2.6.3.2 Privilege mode

The VR4120A supports three system modes: kernel expanded addressing mode, supervisor expanded addressing mode, and user expanded addressing mode. These three modes are described below.

(1) Kernel expanded addressing mode

When the Status register's KX bit has been set, an expanded TLB miss exception vector is used when a TLB miss occurs for the kernel address. While in kernel mode, the MIPS III operation code can always be used, regardless of the KX bit.

(2) Supervisor expanded addressing mode

When the Status register's SX bit has been set, the MIPS III operation code can be used when in supervisor mode and an expanded TLB miss exception vector is used when a TLB miss occurs for the supervisor address.

(3) User expanded addressing mode

When the Status register's UX bit has been set, the MIPS III operation code can be used when in user mode, and an expanded TLB miss exception vector is used when a TLB miss occurs for the user address. When this bit is cleared, the MIPS I and II operation codes can be used, as can 32-bit virtual addresses.

2.6.3.3 Reverse endian

When the Status register's RE bit has been set, the endian ordering is reversed to adopt the user software's perspective. However, the RE bit of the Status register must be set to 0 since the VR4120A supports the little-endian order only.

2.6.3.4 Bootstrap exception vector (BEV)

The BEV bit is used to generate an exception during operation testing (diagnostic testing) of the cache and main memory system. This bit is automatically set to 1 after reset or NMI exception.

When the Status register's BEV bit has been set, the address of the TLB miss exception vector is changed to the virtual address FFFF_FFFF_BFC0_0200H and the ordinary execution vector is changed to address FFFF_FFFF_BFC0_0380H.

When the BEV bit is cleared, the TLB miss exception vector's address is changed to FFFF_FFFF_8000_0000H and the ordinary execution vector is changed to address FFFF_FFFF_8000_0180H.

2.6.3.5 Cache error check

The Status register's CE bit has no meaning because the VR4120A does not support cash parity.

2.6.3.6 Parity error prohibit

When the Status register's DE bit has been set, the processor does not issue any cache parity error exceptions.

2.6.3.7 Interrupt enable (IE)

When the Status register's IE bit has been cleared, no interrupts can be received except for reset interrupts and nonmaskable interrupts.

2.7 Cache Memory

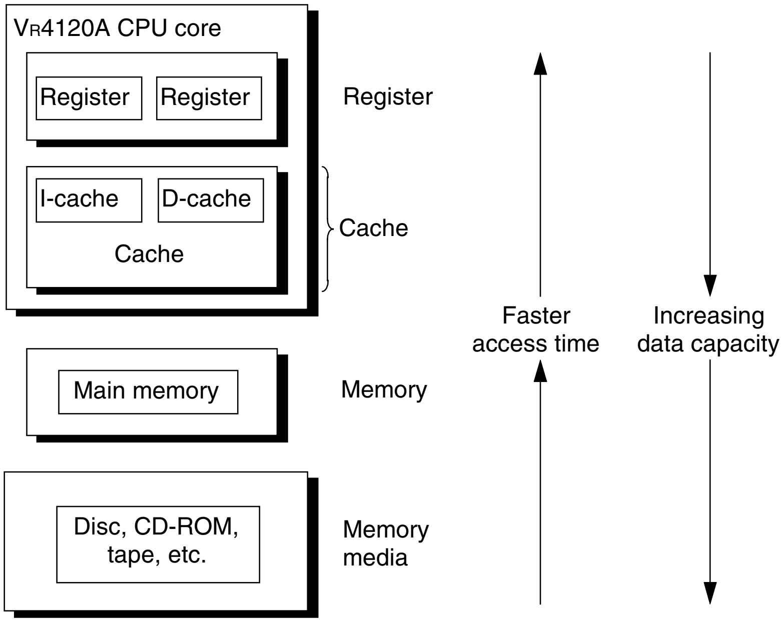
This section describes in detail the cache memory: its place in the VR4120A Core memory organization, and individual organization of the caches.

2.7.1 Memory organization

Figure 2-65 shows the VR4120A Core system memory hierarchy. In the logical memory hierarchy, the caches lie between the CPU and main memory. They are designed to make the speedup of memory accesses transparent to the user.

Each functional block in Figure 2-65 has the capacity to hold more data than the block above it. For instance, main memory (physical memory) has a larger capacity than the caches. At the same time, each functional block takes longer to access than any block above it. For instance, it takes longer to access data in main memory than in the CPU on-chip registers.

Figure 2-65. Logical Hierarchy of Memory



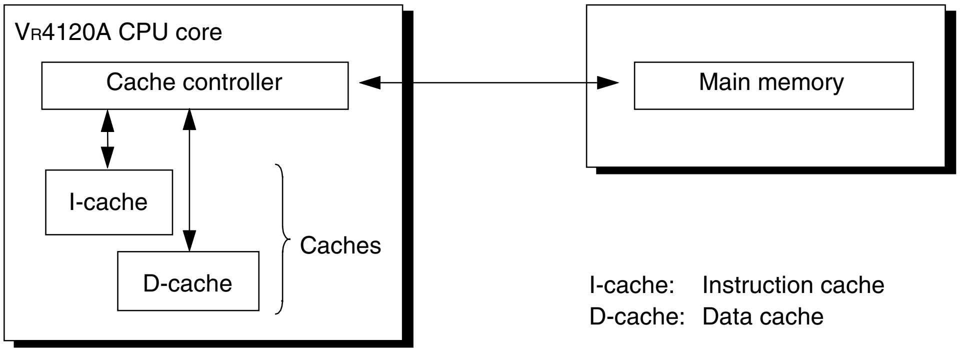
The VR4120A has two on-chip caches: one holds instructions (the instruction cache), the other holds data (the data cache). The instruction and data caches can be read in one PClock cycle.

2 PCycles are needed to write data. However, data writes are pipelined and can complete at a rate of one per PClock cycle. In the first stage of the cycle, the store address is translated and the tag is checked; in the second stage, the data is written into the data RAM.

2.7.2 Cache organization

This section describes the organization of the on-chip data and instruction caches. Figure 2-66 provides a block diagram of the VR4120A Core cache and memory model.

Figure 2-66. Cache Support



(1) Cache line lengths

A cache line is the smallest unit of information that can be fetched from main memory for the cache, and that is represented by a single tag.

The line size for the instruction/data cache is 4 words (16 bytes).

For the cache tag, see **2.7.2.1 Organization of the instruction cache (I-cache)** and **2.7.2.2 Organization of the data cache (D-cache)**.

(2) Cache sizes

The instruction cache in the VR4120A Core is 16 Kbytes; the data cache is 8 Kbytes.

2.7.2.1 Organization of the instruction cache (I-cache)

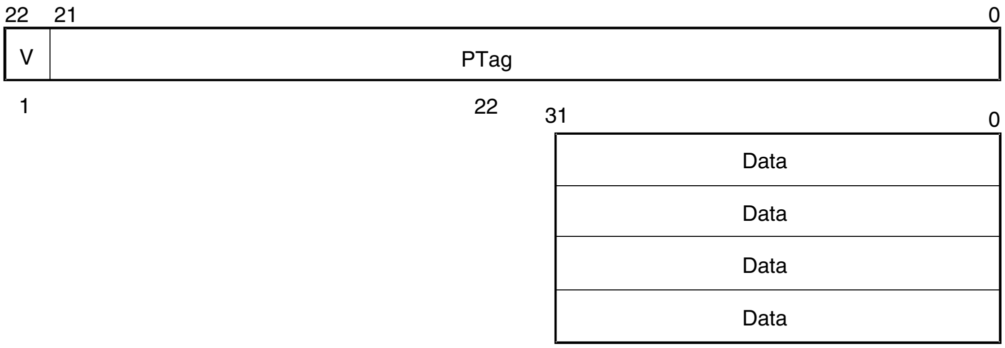
Each line of I-cache data (although it is actually an instruction, it is referred to as data to distinguish it from its tag) has an associated 23-bit tag that contains a 22-bit physical address, and a single valid bit.

The VR4120A Core I-cache has the following characteristics:

- direct-mapped
- indexed with a virtual address
- checked with a physical tag
- organized with a 4-word (16-byte) cache line

Figure 2-67 shows the format of a 4-word (16-byte) I-cache line.

Figure 2-67. Instruction Cache Line Format



PTag : Physical tag (bits 31 to 10 of physical address)
V : Valid bit
Data : Cache data

2.7.2.2 Organization of the data cache (D-cache)

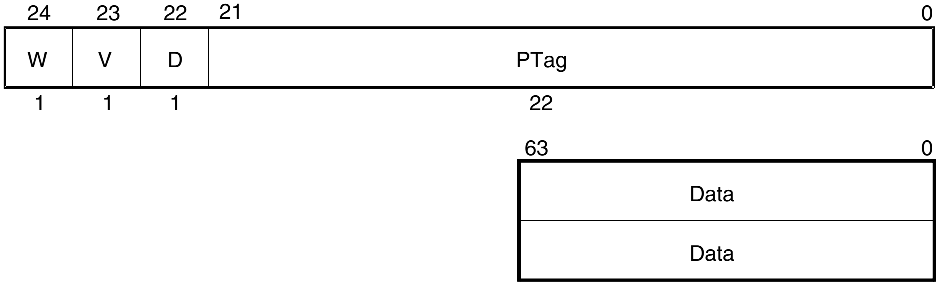
Each line of D-cache data has an associated 25-bit tag that contains a 22-bit physical address, a Valid bit, a Dirty bit, and a Write-back bit.

The VR4120A Core D-cache has the following characteristics :

- ✧ write-back
- ✧ direct-mapped
- ✧ indexed with a virtual address
- ✧ checked with a physical tag
- ✧ organized with a 4-word (16-byte) cache line.

Figure 2-68 shows the format of a 4-word (16-byte) D-cache line.

Figure 2-68. Data Cache Line Format



W : Write-back bit (set if cache line has been written)
D : Dirty bit
V : Valid bit
PTag : Physical tag (bits 31 to 10 of physical address)
Data : D-cache data

2.7.2.3 Accessing the caches

Figure 2-69 shows the virtual address (VA) index into the caches. The number of virtual address bits used to index the instruction and data caches depends on the cache size.

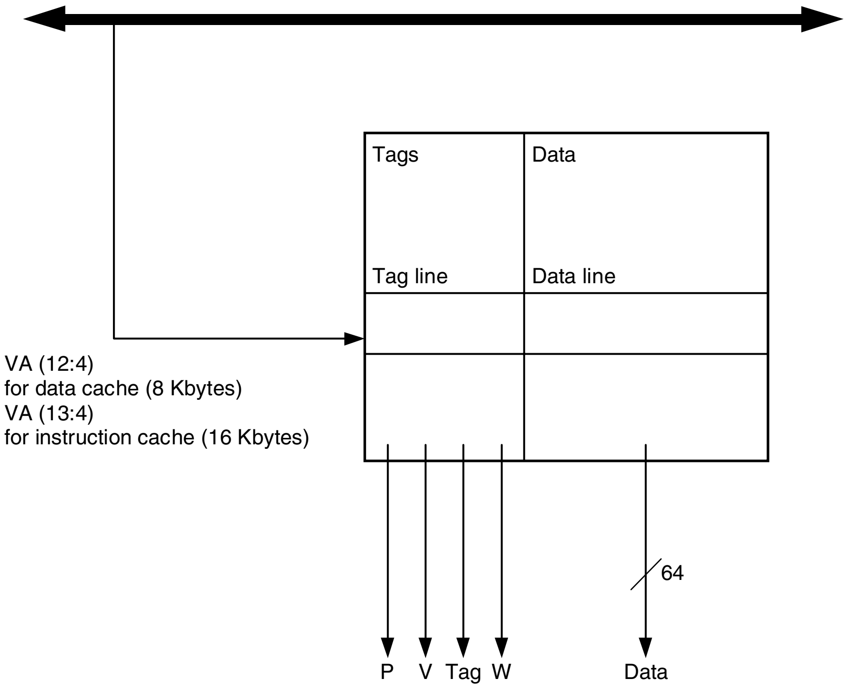
(1) Data cache addressing

Using VA (12:4). The most-significant bit is VA12 because the cache size is 8 Kbytes.
The least-significant bit is VA4 because the line size is 4 words (16 bytes).

(2) Instruction cache addressing

Using VA (13:4). The most-significant bit is VA13 because the cache size is 16 Kbytes.
The least-significant bit is VA4 because the line size is 4 words (16 bytes).

Figure 2-69. Cache Data and Tag Organization



2.7.3 Cache operations

As described earlier, caches provide temporary data storage, and they make the speedup of memory accesses transparent to the user. In general, the CPU core accesses cache-resident instructions or data through the following procedure:

1. The CPU core, through the on-chip cache controller, attempts to access the next instruction or data in the appropriate cache.
2. The cache controller checks to see if this instruction or data is present in the cache.
   - ✧ If the instruction/data is present, the CPU core retrieves it. This is called a cache hit.
   - ✧ If the instruction/data is not present in the cache, the cache controller must retrieve it from memory. This is called a cache miss.
3. The CPU core retrieves the instruction/data from the cache and operation continues.

It is possible for the same data to be in two places simultaneously: main memory and cache. This data is kept consistent through the use of a write-back methodology; that is, modified data is not written back to memory until the cache line is to be replaced.

Instruction and data cache line replacement operations are described in the following sections.

2.7.3.1 Cache write policy

The VR4120A Core manages its data cache by using a write-back policy; that is, it stores write data into the cache, instead of writing it directly to memory[Note]. Some time later this data is independently written into memory. In the VR4120A implementation, a modified cache line is not written back to memory until the cache line is to be replaced either in the course of satisfying a cache miss, or during the execution of a write-back CACHE instruction.

When the CPU core writes a cache line back to memory, it does not ordinarily retain a copy of the cache line, and the state of the cache line is changed to invalid.

Note Contrary to the write-back, the write-through cache policy stores write data into the memory and cache simultaneously.

2.7.4 Cache states

(1) Cache line

The three terms below are used to describe the state of a cache line:

✧ Dirty: a cache line containing data that has changed since it was loaded from memory.
✧ Clean: a cache line that contains data that has not changed since it was loaded from memory.
✧ Invalid: a cache line that does not contain valid information must be marked invalid, and cannot be used. For example, after a Soft Reset, software sets all cache lines to invalid. A cache line in any other state than invalid is assumed to contain valid information. Neither Cold reset nor Soft reset makes the cache state invalid. Software makes the cache state invalid.

(2) Data cache

The data cache supports three cache states:

✧ Invalid
✧ Valid clean
✧ Valid dirty

(3) Instruction cache

The instruction cache supports two cache states:

✧ Invalid
✧ Valid

The state of a valid cache line may be modified when the processor executes a CACHE operation. CACHE operations are described in **APPENDIX A MIPS III INSTRUCTION SET DETAILS**.

2.7.5 Cache state transition diagrams

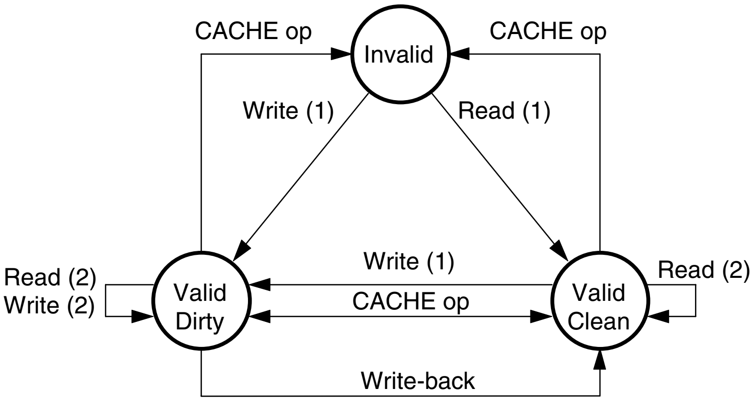
The following section describes the cache state diagrams for the data and instruction cache lines. These state diagrams do not cover the initial state of the system, since the initial state is system-dependent.

2.7.5.1 Data cache state transition

The following diagram illustrates the data cache state transition sequence. A load or store operation may include one or more of the atomic read and write operations shown in the state diagram below, which may cause cache state transitions.

- ✧ Read (1) indicates a read operation from main memory to cache, inducing a cache state transition.
- ✧ Read (2) indicates a read operation from cache to the CPU core, which induces no cache state transition.
- ✧ Write (1) indicates a write operation from CPU core to cache, inducing a cache state transition.
- ✧ Write (2) indicates a write operation from CPU core to cache, which induces no cache state transition.

Figure 2-70. Data Cache State Diagram



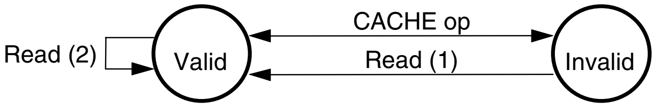
2.7.5.2 Instruction cache state transition

The following diagram illustrates the instruction cache state transition sequence.

Read (1) indicates a read operation from main memory to cache, inducing a cache state transition.

Read (2) indicates a read operation from cache to the CPU core, which induces no cache state transition.

Figure 2-71. Instruction Cache State Diagram



2.7.6 Cache data integrity

Figures 2-72 to 2-86 shows checking operations for various cache accesses.

Figure 2-72. Data Check Flow on Instruction Fetch

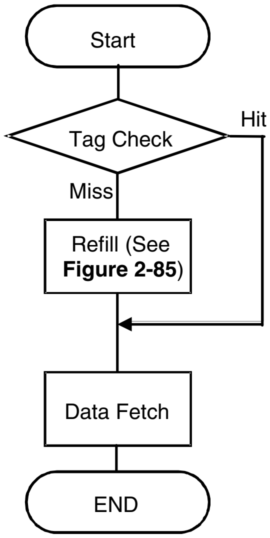


Figure 2-73. Data Check Flow on Load Operations

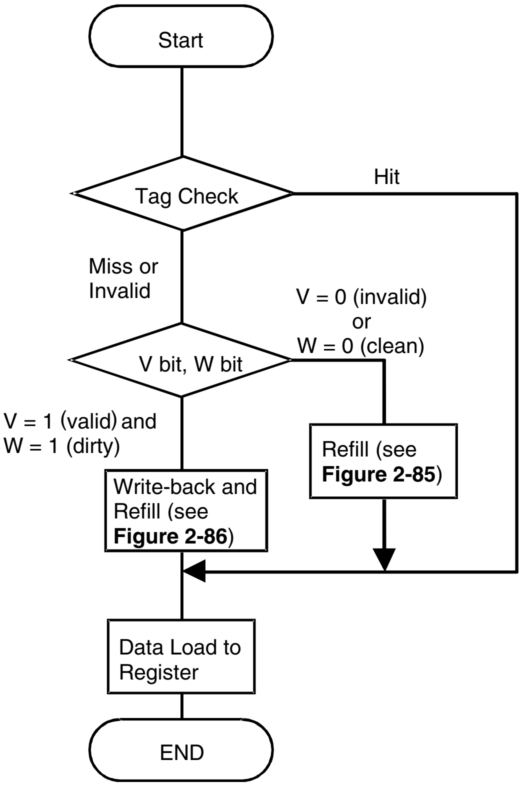

Start
Tag Check
Hit
Miss or Invalid
V = 0 (invalid) or W = 0 (clean)
V bit, W bit
V = 1 (valid) and W = 1 (dirty)
Refill (see **Figure 2-85**)
Write-back and Refill (see **Figure 2-86**)
Data Load to Register
END

Figure 2-74. Data Check Flow on Store Operations

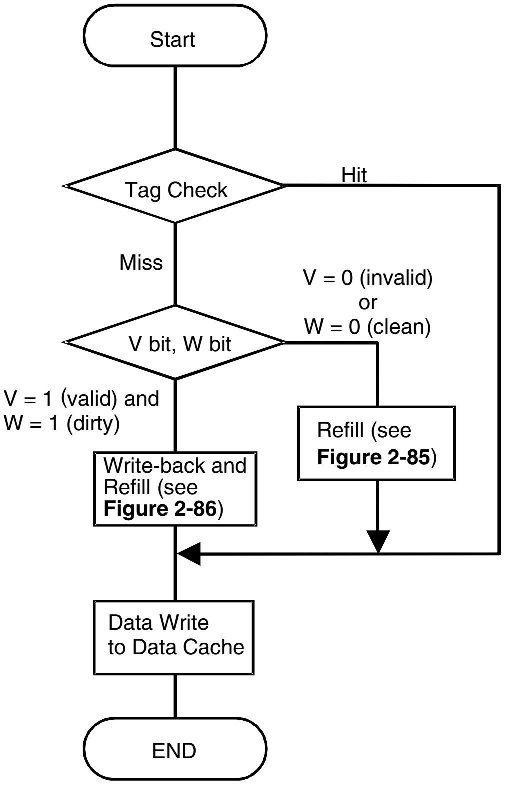


Figure 2-75. Data Check Flow on Index_Invalidate Operations

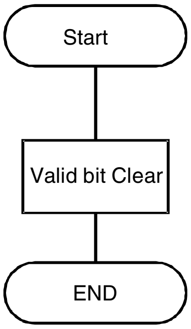


Figure 2-76. Data Check Flow on Index_Writeback_Invalidate Operations

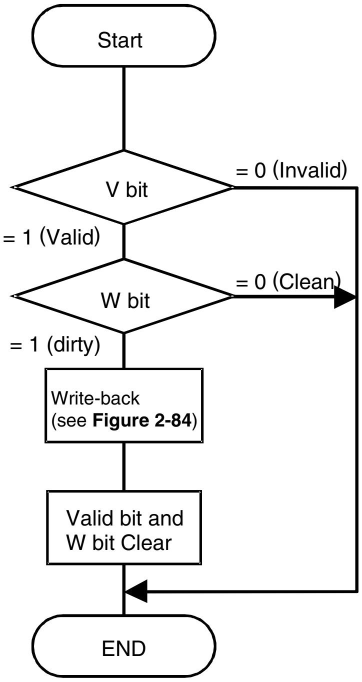


Figure 2-77. Data Check Flow on Index_Load_Tag Operations

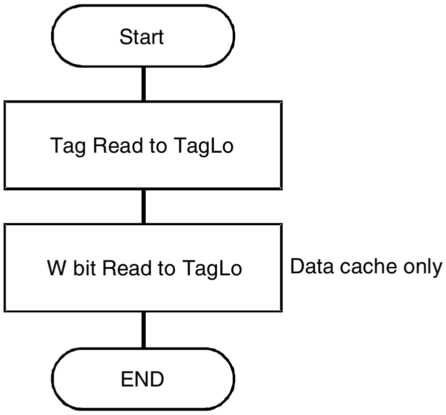


Figure 2-78. Data Check Flow on Index_Store_Tag Operations

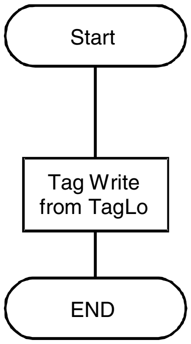


Figure 2-79. Data Check Flow on Create_Dirty Operations

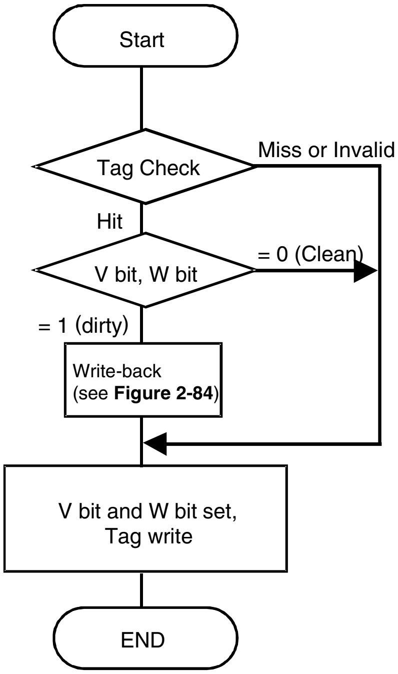


Figure 2-80. Data Check Flow on Hit_Invalidate Operations

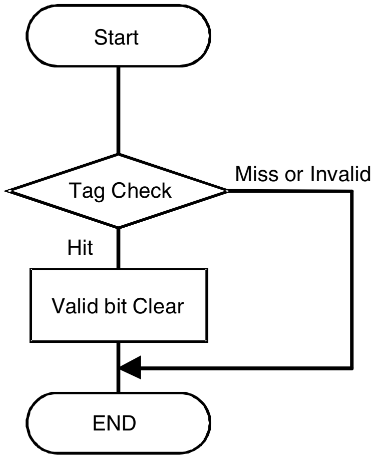


Figure 2-81. Data Check Flow on Hit_Writeback_Invalidate Operations

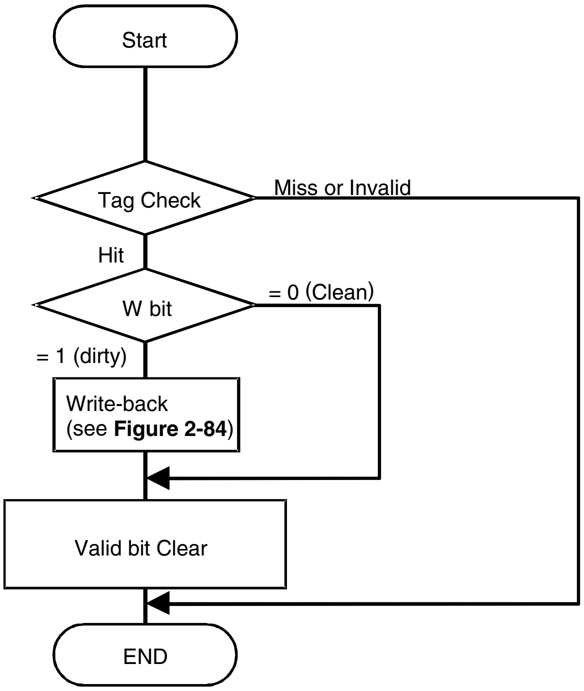


Figure 2-82. Data Check Flow on Fill Operations

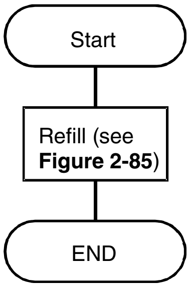


Figure 2-83. Data Check Flow on Hit_Writeback Operations

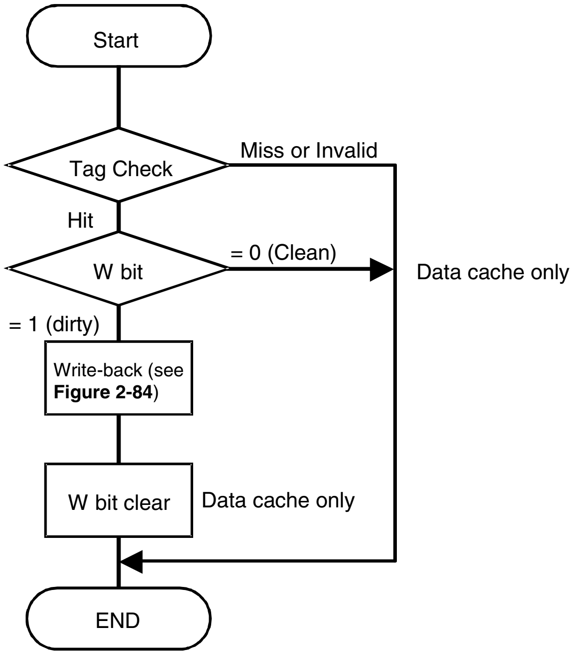


Figure 2-84. Writeback Flow

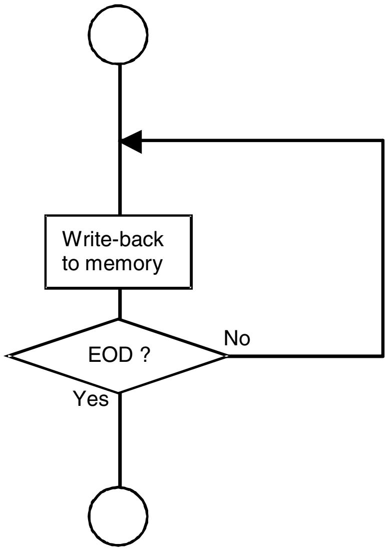


Figure 2-85. Refill Flow

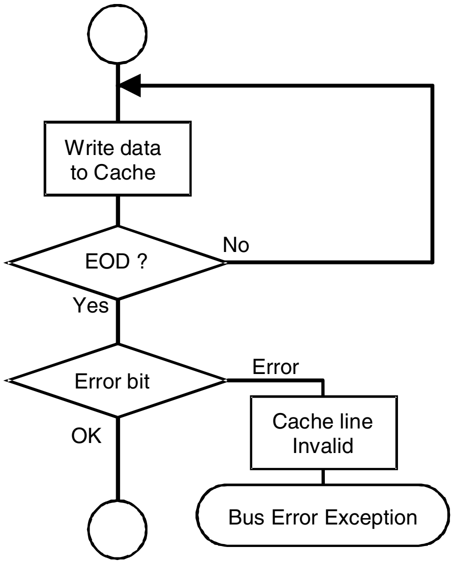
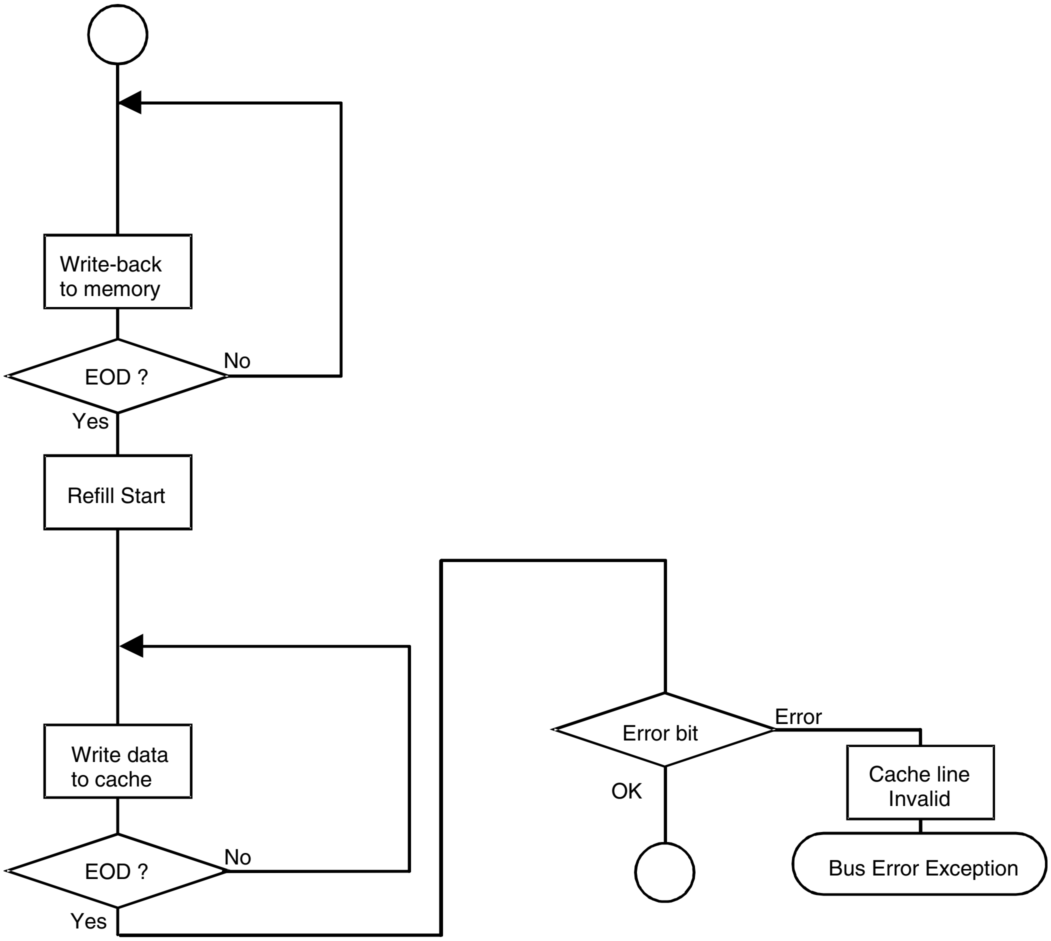


Figure 2-86. Writeback & Refill Flow



Remark Write-back Procedure:

On a store miss write-back, data tag is checked and data is transferred to the write buffer. If an error is detected in the data field, the write back is not terminated; the erroneous data is still written out to main memory. If an error is detected in the tag field, the write-back bus cycle is not issued.

The cache data may not be checked during CACHE operation.

2.7.7 Manipulation of the caches by an external agent

The VR4120A does not provide any mechanisms for an external agent to examine and manipulate the state and contents of the caches.

2.8 CPU Core Interrupts

Four types of interrupt are available on the CPU core. These are:

- ✧ one non-maskable interrupt, NMI
- ✧ five ordinary interrupts
- ✧ two software interrupts
- ✧ one timer interrupt

For the interrupt request input to the CPU core.

2.8.1 Non-maskable interrupt (NMI)

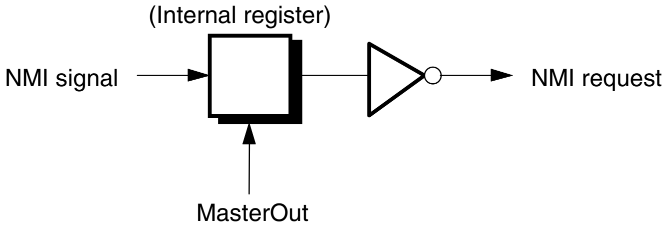
The non-maskable interrupt is acknowledged by asserting the NMI signal (internal), forcing the processor to branch to the Reset Exception vector. This signal is latched into an internal register at the rising edge of MasterOut signal (internal), as shown in Figure 2-87.

NMI only takes effect when the processor pipeline is running.

This interrupt cannot be masked.

Figure 2-87 shows the internal service of the NMI signal. The NMI signal is latched into an internal register by the rising edge of MasterOut. The latched signal is inverted to be transferred to inside the device as an NMI request.

Figure 2-87. Non-maskable Interrupt Signal



2.8.2 Ordinary interrupts

Ordinary interrupts are acknowledged by asserting the Int(4:0) signals (internal). However, Int4 never occurs in the VR4120A.

This interrupt request can be masked with the IM (6:2), IE, and EXL fields of the Status register.

2.8.3 Software interrupts generated in CPU core

Software interrupts generated in the CPU core use bits 1 and 0 of the IP (interrupt pending) field in the Cause register. These may be written by software, but there is no hardware mechanism to set or clear these bits.

After the processing of a software interrupt exception, corresponding bit of the IP field in the Cause register must be cleared before returning to ordinary routine or enabling multiple interrupts until the operation returns to normal routine.

This interrupt request is maskable through the IM (1:0), IE, and EXL fields of the Status register.

2.8.4 Timer interrupt

The timer interrupt uses bit 7 of the IP (interrupt pending) field of the Cause register. This bit is set automatically whenever the value of the Count register equals the value of the Compare register, and an interrupt request is acknowledged.

This interrupt is maskable through IM7 of the IM field of the Status register.

2.8.5 Asserting interrupts

2.8.5.1 Detecting hardware interrupts

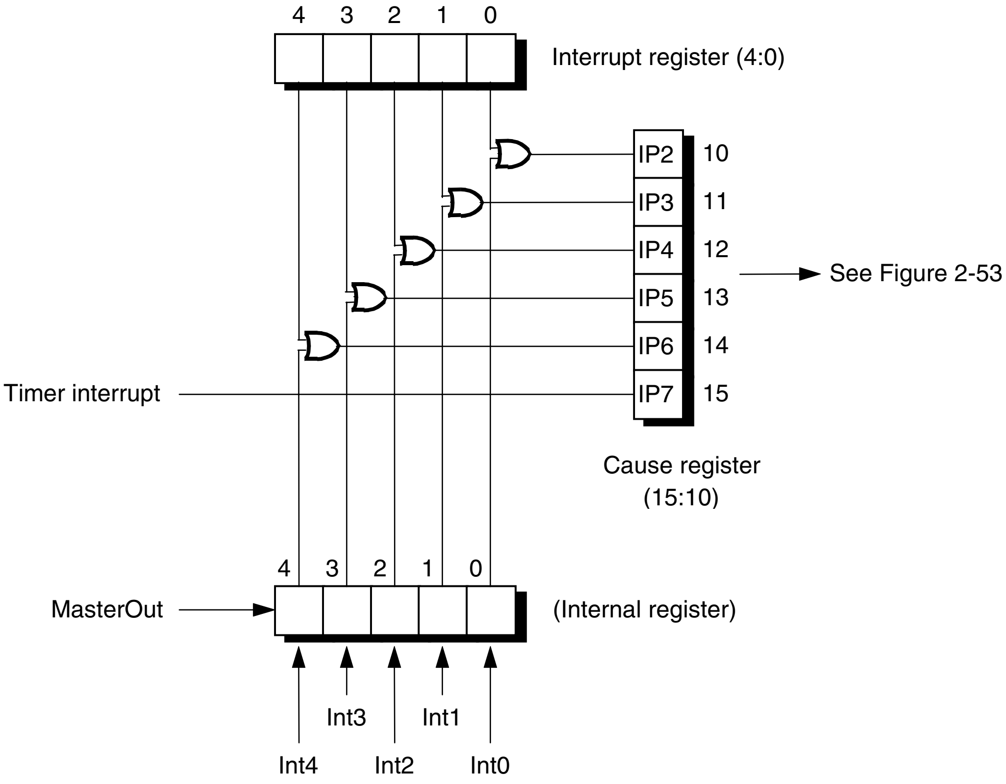
Figure 2-88 shows how the hardware interrupts are readable through the Cause register.

The timer interrupt signal, IP7, is directly readable as bit 15 of the Cause register.

Bits 4 to 0 of the Interrupt register are bit-wise ORed with the current value of the Int4 to 0 signals and the result is directly readable as bits 14 to 10 of the Cause register.

IP1 and IP0 of the Cause register, which are described in **Section 2.5 Exception Processing**, are software interrupts. There is no hardware mechanism for setting or clearing the software interrupts.

Figure 2-88. Hardware Interrupt Signals

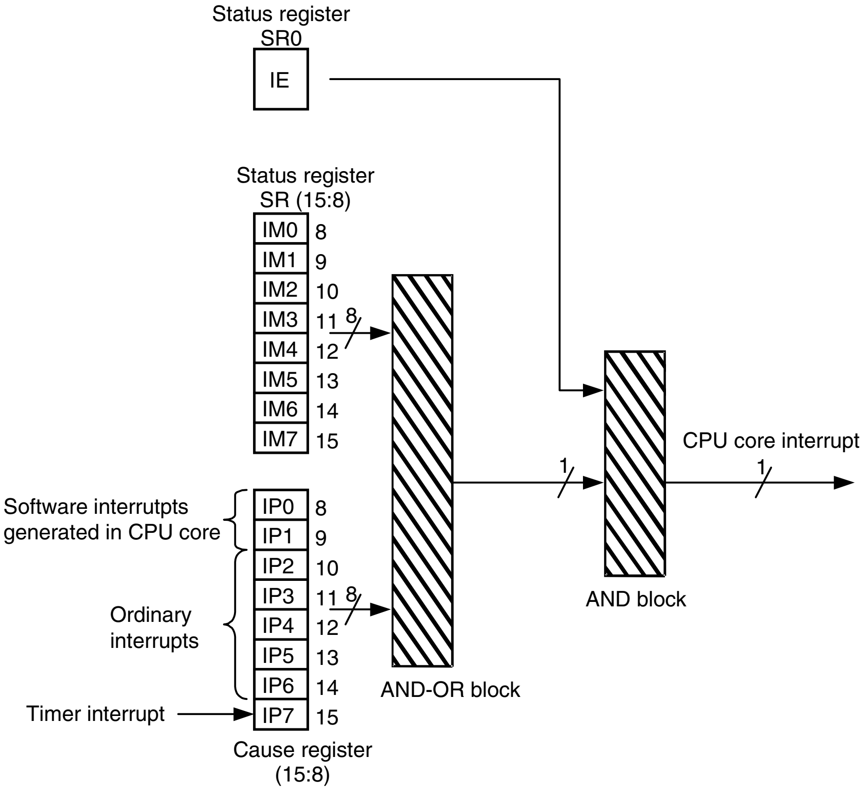


2.8.5.2 Masking interrupt signals

Figure 2-89 shows the masking of the CPU core interrupt signals.

- ✧ Cause register bits 15 to 8 (IP7 to IP0) are AND-ORed with Status register interrupt mask bits 15 to 8 (IM7 to IM0) to mask individual interrupts.
- ✧ Status register bit 0 is a global Interrupt Enable (IE). It is ANDed with the output of the AND-OR logic shown in Figure 2-89 to produce the CPU core interrupt signal. The EXL bit in the Status register also enables these interrupts.

Figure 2-89. Masking of Interrupt Request Signals



Bit	Function	Setting
IE	Whole interrupts enable	1 : Enable 0 : Disable
IM(7:0)	Interrupt mask	Each bit 1 : Enable 0 : Disable
IP(7:0)	Interrupt request	Each bit 1 : Pending 0 : Not pending

CHAPTER 3 SYSTEM CONTROLLER

3.1 Overview

Register map

This block is an internal system controller for the μPD98502. System controller provides bridging function among the CPU system bus "SysAD", NEC original high-speed on-chip bus "IBUS" and memory bus for SDRAM/PROM/flash.

Features of system controller are as follows.

- Provides bus bridging function among SysAD bus, IBUS and memory
- Supports endian converting function on SysAD bus
- Can directly connect to SDRAM and PROM/flash
- Supports Deadman's SW timer and separated 2-ch timers
- Supports NS16550 compatible UART

3.1.1 CPU interface

- Connects directly to the VR4120A CPU bus "SysAD bus"
- Supports all VR4120A bus cycles at 66 MHz or 100 MHz
- Supports only data rate D
- Supports only sequential ordering
- 4-word (16-byte) x 4-entry write command buffer
- Little-endian or big-endian byte order
- Don't support 8-words burst R/W on SysAD bus

3.1.2 Memory interface

- 66-MHz or 100-MHz memory bus
- Up to 32-MB base memory range supports SDRAM
- Up to 8-MB write-protectable boot memory range supports PROM/flash
- On-chip programmable SDRAM refresh controller
- 4-word (16-byte) write data buffer
- 4-word (16-byte) prefetch data buffer (memory-to-CPU)
- PROM/flash data signals multiplexed on SDRAM data signals
- Variable Flash memory data bus (8,16,32 bits)
- Programmable memory bus arbitration priority
- Programmable address ranges for the memory
- Programmable RAS-CAS delay (2, 3, 4 clocks)
- Programmable CAS latency (2, 3 clocks)

3.1.3 IBUS Interface

- Master and target capability
- 64-word (256-byte) IBUS Slave TxFIFO (IBUS reads data from memory)
- 64-word (256-byte) IBUS Slave RxFIFO (IBUS writes data to memory)
- 4-word (16-byte) IBUS Master TxFIFO (VR4120A reads data from IBUS)
- 4-word (16-byte) x 4 entry IBUS Master RxFIFO (VR4120A writes data to IBUS)
- Supports bus timer to detect IBUS stall

- 66-MHz IBUS clock rate
- Supports 266-MB/sec (32 bits @66 MHz) bursts on IBUS.
- Support endian conversion between memory and IBUS slave I/F
- Support endian conversion between SyaAD bus and IBUS master I/F

3.1.4 UART

- Universal Asynchronous Receiver/Transmitter
- Modem control functions
- Even, odd or no parity bit generation
- Fully prioritized interrupt control

3.1.5 EEPROM

- 165/250-kHz clock rate (Depend on CPU clock rate ; 66/100 MHz)
- Support only 3.3-V EEPROM (Recommended National Semiconductor's "NM93C46")
- Support Micro Wire interface for Serial EEPROM
- Support auto-load function for two addresses of MAC at system boot

3.1.6 Timer

- Two 32-bit loadable general-purpose timers generating interrupt to CPU

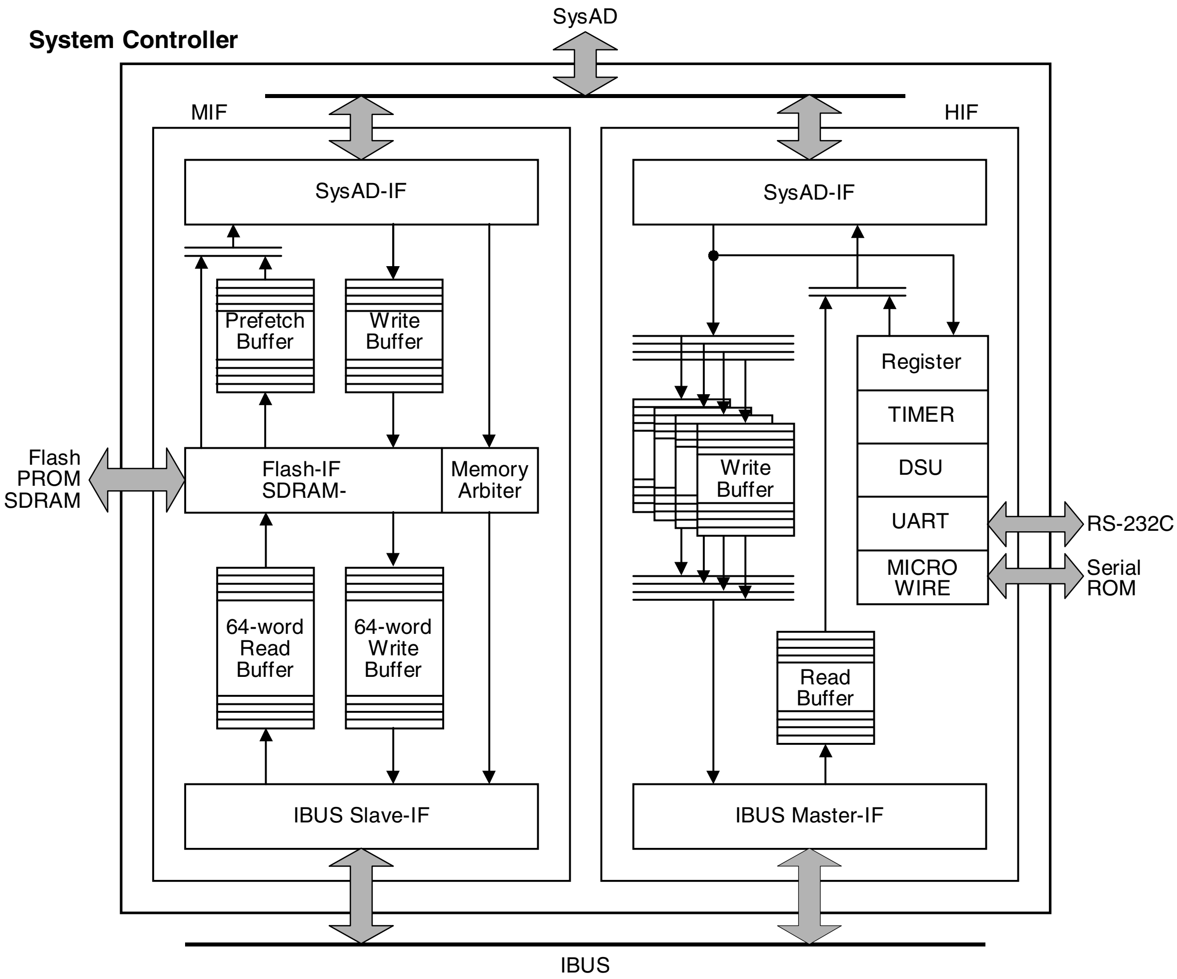
3.1.7 Interrupt controller

- Generates NMI and INT
- All interrupt causing events maskable

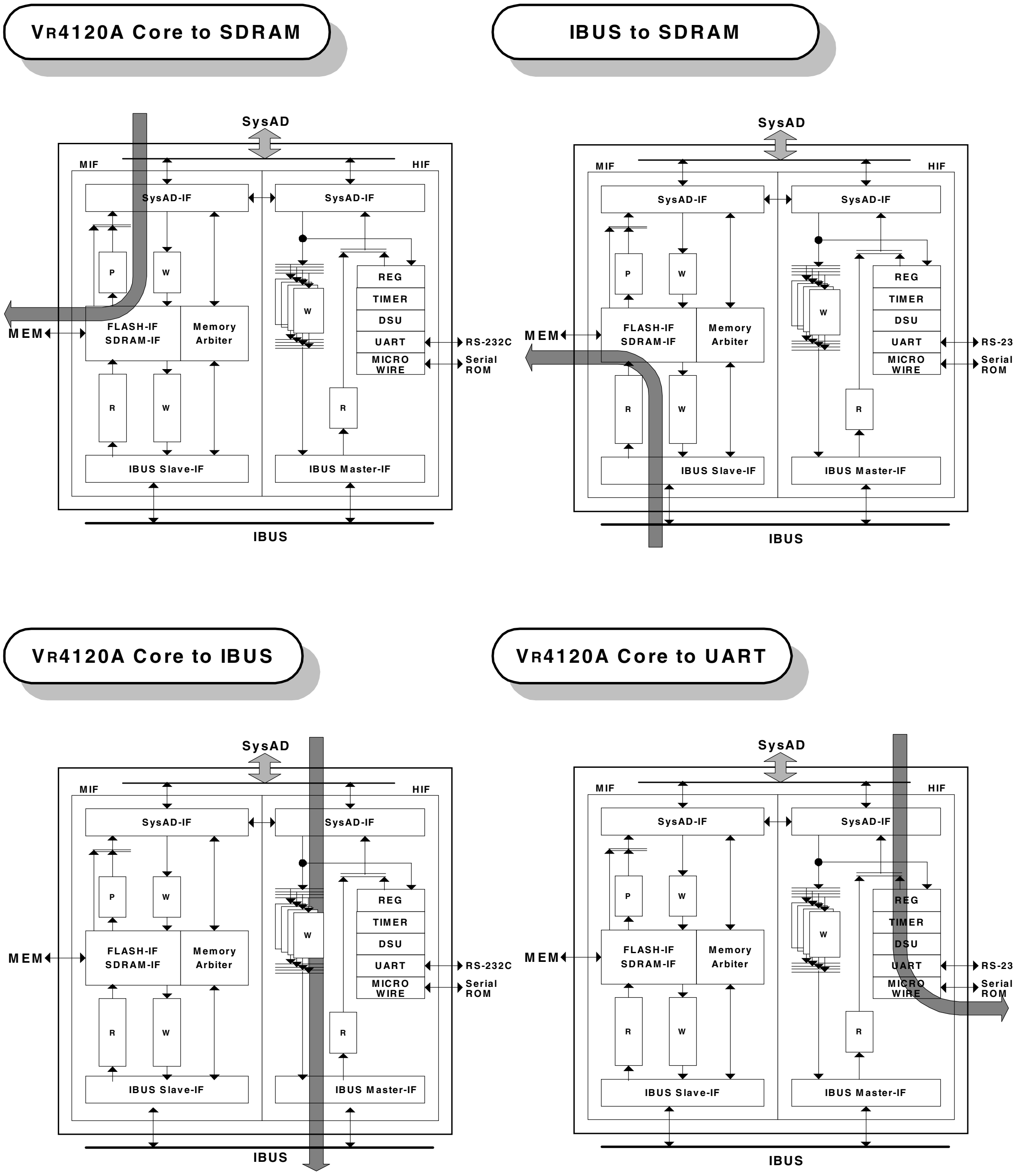
3.1.8 DSU (Deadman's SW Unit)

- Deadman's SW Unit generates cold reset to CPU

3.1.9 System block diagram



3.1.10 Data flow diagram



VR4120A Core to SDRAM

IBUS to SDRAM

VR4120A Core to IBUS

VR4120A Core to UART

3.2 Registers

3.2.1 Register map

Following Table summarizes the controller's register set. The base address for the set is 1000_0000H in the physical address space.

Offset Address	Register Name	R/W	Access	Description
1000_0000H	S_GMR	R/W	W/H/B	General Mode Register
1000_0004H	S_GSR	R	W/H/B	General Status Register
1000_0008H	S_ISR	RC	W/H/B	Interrupt Status Register
1000_000CH	S_IMR	R/W	W/H/B	Interrupt Mask Register
1000_0010H	S_NSR	RC	W/H/B	NMI Status Register
1000_0014H	S_NER	R/W	W/H/B	NMI Enable Register
1000_0018H	S_VER	R	W/H/B	Version Register
1000_001CH	S_IOR	R/W	W/H/B	IO Port Register
1000_0020H: 1000_002CH	N/A	-	-	Reserved for future use
1000_0030H	S_WRCR	W	W/H/B	Warm Reset Control Register
1000_0034H	S_WRSR	R	W/H/B	Warm Reset Status Register
1000_0038H	S_PWCR	R/W	W/H/B	Power Control Register
1000_003CH	S_PWSR	R	W/H/B	Power Control Status Register
1000_0040H: 1000_0048H	N/A	-	-	Reserved for future use
1000_004CH	ITCNTR	R/W	W/H/B	IBUS Timeout Timer Control Register
1000_0050H	ITSETR	R/W	W/H/B	IBUS Timeout Timer Set Register
1000_0054H: 1000_007CH	N/A	-	-	Reserved for future use
1000_0080H	UARTRBR	R	W/H/B	UART, Receiver Buffer Register [DLAB=0,READ]
1000_0080H	UARTTHR	W	W/H/B	UART, Transmitter Holding Register [DLAB=0,WRITE]
1000_0080H	UARTDLL	R/W	W/H/B	UART, Divisor Latch LSB Register [DLAB=1]
1000_0084H	UARTIER	R/W	W/H/B	UART, Interrupt Enable Register [DLAB=0]
1000_0084H	UARTDLM	R/W	W/H/B	UART, Divisor Latch MSB Register [DLAB=1]
1000_0088H	UARTIIR	R	W/H/B	UART, Interrupt ID Register [READ]
1000_0088H	UARTFCR	W	W/H/B	UART, FIFO control Register [WRITE]
1000_008CH	UARTLCR	R/W	W/H/B	UART, Line control Register
1000_0090H	UARTMCR	R/W	W/H/B	UART, Modem Control Register
1000_0094H	UARTLSR	R/W	W/H/B	UART, Line status Register
1000_0098H	UARTMSR	R/W	W/H/B	UART, Modem Status Register
1000_009CH	UARTSCR	R/W	W/H/B	UART, Scratch Register
1000_00A0H	DSUCNTR	R/W	W/H/B	DSU Control Register
1000_00A4H	DSUSETR	R/W	W/H/B	DSU Dead Time Set Register
1000_00A8H	DSUCLRR	W	W/H/B	DSU Clear Register
1000_00ACH	DSUTIMR	R	W/H/B	DSU Elapsed Time Register
1000_00B0H	TMMR	R/W	W/H/B	Timer Mode Register
1000_00B4H	TM0CSR	R/W	W/H/B	Timer CH0 Count Set Register
1000_00B8H	TM1CSR	R/W	W/H/B	Timer CH1 Count Set Register
1000_00BCH	TM0CCR	R	W/H/B	Timer CH0 Current Count Register
1000_00C0H	TM1CCR	R	W/H/B	Timer CH1 Current Count Register
1000_00C4H: 1000_00CCH	N/A	-	-	Reserved for future use
1000_00D0H	ECCR	W	W/H/B	EEPROM Command Control Register
1000_00D4H	ERDR	R	W/H/B	EEPROM Read Data Register

Offset Address	Register Name	R/W	Access	Description
1000_00D8H	MACAR1	R	W/H/B	MAC Address Register 1
1000_00DCH	MACAR2	R	W/H/B	MAC Address Register 2
1000_00E0H	MACAR3	R	W/H/B	MAC Address Register 3
1000_00E4H: 1000_00FCH	N/A	-	-	Reserved for future use
1000_0100H	RMMDR	R/W	W	Boot ROM Mode Register
1000_0104H	RMATR	R/W	W	Boot ROM Access Timing Register
1000_0108H	SDMDR	R/W	W	SDRAM Mode Register
1000_010CH	SDTSR	R/W	W	SDRAM Type Selection Register
1000_0110H	SDPTR	R/W	W	SDRAM Precharge Timing Register
1000_0114H: 1000_0118H	N/A	-	-	Reserved for future use
1000_011CH	SDRMR	R/W	W	SDRAM Refresh Mode Register
1000_0120H	SDRCR	R	W	SDRAM Refresh Timer Count Register
1000_0124H	MBCR	R/W	W	Memory Bus Control Register
1000_0128H: 1000_0FFCH	N/A	-	-	Reserved for future use

Remarks 1. In the “R/W” field,
“W” means “writeable”,
“R” means “readable”,
“RC” means “read-cleared”,
“- “ means “not accessible”.

2. All internal registers are 32-bit word-aligned registers.

3. The burst access to the internal register is prohibited.
If such burst access has been occurred, IRERR bit in NSR is set and NMI will assert to CPU.

4. Read access to the reserved area will set the CBERR bit in the NSR register and the dummy read response data with the data-error bit set on SysCMD [0] is returned.

5. Write access to the reserved area will set the CBERR bit in the NSR register, and the write data is lost.

6. In the “Access” filed,
“W” means that word access is valid,
“H” means that half word access is valid,
“B” means that byte access is valid.

7. Write access to the read-only register cause no error, but the write data is lost.

8. The CPU can access all internal registers, but IBUS master device cannot access them.

3.2.2 S_GMR (General Mode Register)

The general mode register “S_GMR” is a read-write and 32-bit word-aligned register. After initializing, VR4120A sets the IAEN bit to enable the IBUS arbiter. S_GMR is initialized to 0 at reset and contains the following fields:

Bits	Field	R/W	Default	Description
31:10	Reserved	R/W	0	Hardwired to 0.
9	MSWP	R/W	0	MIF block data swap function enable: 0 = disable 1 = enable
8	HSWP	R/W	0	HIF block data swap function enable: 0 = disable 1 = enable
7:4	Reserved	R/W	0	Hardwired to 0.
3	UCSEL	R/W	0	UART source clock selection: 0 = use 1/2 of CPU clock 1 = use external clock (18.432 MHz)
2	MPFD	R/W	0	Memory-to-CPU prefetch FIFO disable: 0 = enable 1 = disable
1	IAEN	R/W	0	IBUS arbiter enable: 0 = disable (IBUS arbiter does not allow the grant except system controller) 1 = enable.
0	CRST	R/W	0	Cold reset: 0 = do nothing 1 = perform cold reset (same as hardware system reset)

3.2.3 S_GSR (General Status Register)

The general status register “S_GSR” is a read-only and 32-bit word-aligned register. S_GSR indicates the status of external pins of the μPD98502. S_GSR contains the following fields:

Bits	Field	R/W	Default	Description
31:3	Reserved	R	0	Hardwired to 0.
2	MIPS16	R	-	Reflects the status of external pin “MIPS16” after reset: This field indicates the same value as M16 bit of CPU configuration register. 0 = connected to GND, that means MIPS16 mode is disabled NOTE. 1 = connected to VCC, that means MIPS16 mode is enabled.
1	CLKSL	R	-	Reflects the status of external pin “CLKSL” after reset: 0 = connected to GND, that means CPU is operated at 100 MHz. 1 = connected to VCC, that means CPU is operated at 66 MHz.
0	ENDCEN	R	-	Reflects the status of external pin “ENDCEN” after reset: 0 = connected to GND, that means Endian Converter is disabled. 1 = connected to VCC, that means Endian Converter is enabled.

NOTE The μPD98502 does not support MIPS16 mode. MIPS16 mode pin (located D11) should be connected to GND.

3.2.4 S_ISR (Interrupt Status Register)

The interrupt status register "S_ISR" is a read-clear and 32-bit word-aligned register. S_ISR indicates the interruption status from SysAD/IBUS interfaces, timer, UART and so on. If corresponding bit in S_IMR (Interrupt Mask Register) is set and the interrupt is not masked, system controller interrupts to VR4120A using interrupt signal. The bit in S_ISR is reset after being read by the VR4120A. When the same type of incident occurs before the bit has been read, the bit will be set again. S_ISR is initialized to 0 at reset and contains the following fields:

Bits	Field	R/W	Default	Description
31:18	Reserved	RC	0	Hardwired to 0.
17	MAC2IS	RC	0	MAC2 interrupt: 0 = no MAC2 interrupt pending. 1 = MAC2 interrupt pending.
16	PCIIS	RC	0	PCI interrupt: 0 = no PCI interrupt pending. 1 = PCI interrupt pending.
15:5	Reserved	RC	0	Hardwired to 0.
4	WUIS	RC	0	Wakeup interrupt: 0 = no wakeup request pending. 1 = some wakeup request pending.
3	EXTIS	RC	0	External interrupt: 0 = no external interrupt pending. 1 = external interrupt pending.
2	UARTIS	RC	0	UART interrupt: 0 = no UART interrupt pending. 1 = UART interrupt pending. UART interruption is one of the following interruptions: 1. UART receive data buffer full interrupt 2. UART transmitter buffer empty interrupt 3. UART line status interrupt 4. UART modem status interrupt
1	TM1IS	RC	0	Timer CH1 interrupt: 0 = no timer CH1 interrupt pending. 1 = timer CH1 interrupt pending.
0	TM0IS	RC	0	Timer CH0 interrupt: 0 = no timer CH0 interrupt pending. 1 = timer CH0 interrupt pending.

Remarks 1. To clear bits 0 to 4 in this register, the VR4120A must read the byte that contains the TM0IS register. In addition, to clear bits 16 to 17, the VR4120A must read the byte that contains the PCIIS register.

2. MAC2 interrupt and PCI interrupt can not be masked by system controller.

3. After clearing MAC2IS/PCIIS bit, MAC/PCI block continues to provide interrupt signal to VR4120A.

3.2.5 S_IMR (Interrupt Mask Register)

The interrupt mask register "S_IMR" is a read-write and 32-bit word-aligned register. S_IMR masks interruption for each corresponding incident. A mask bit, which locates in the same bit location to a corresponding bit in S_ISR, controls interruption triggered by the incident. If a bit of this register is reset to 0, the corresponding bit of the S_ISR is masked. If it is set to 1, the corresponding bit is unmasked. When the unmask bit is set and the bit in S_ISR is set, system controller asserts interrupt signal to VR4120A. S_IMR is initialized to 0 at reset and contains the following fields:

Bits	Field	R/W	Default	Description
31:5	Reserved	R/W	0	Hardwired to 0.
4	WUIM	R/W	0	Wakeup interrupt mask: 1 = unmask. 0 = mask.
3	EXTIM	R/W	0	External interrupt mask: 1 = unmask. 0 = mask.
2	UARTIM	R/W	0	UART interrupt mask: 1 = unmask. 0 = mask.
1	TM1IM	R/W	0	Timer CH1 interrupt mask: 1 = unmask. 0 = mask.
0	TM0IM	R/W	0	Timer CH0 interrupt mask: 1 = unmask. 0 = mask.

Remark MAC2 interrupt and PCI interrupt can not be masked by system controller.

3.2.6 S_NSR (NMI Status Register)

The interrupt status register “S_NSR” is a read-clear and 32-bit word-aligned register. S_NSR indicates the non-maskable interruption “NMI” status from SysAD/IBUS interfaces, external NMI, memory interface and so on. If corresponding bit in S_NER (NMI Enable Register) is set and the NMI is enabled, system controller interrupts to VR4120A using non-maskable interrupt signal. The bit in S_NSR is reset after being read by the VR4120A. When the same type of incident occurs before the bit has been read, the bit will be set again. S_NSR is initialized to 0 at reset and contains the following fields:

Bits	Field	R/W	Default	Description
31:6	Reserved	RC	0	Hardwired to 0.
5	IRERR	RC	0	Illegal internal register access error: 0 = no such access. 1 = illegal internal register access, ex burst access has been performed.
4	EXTNMI	RC	0	External NMI: 0 = external NMI is not asserted. 1 = external NMI is asserted.
3	MAERR	RC	0	Memory address error: Memory address error includes the memory access to the illegal memory space (RFU space and out range of the SDRAM/ROM space) and the illegal memory access (byte or half-word ROM access or burst write access to the ROM). 0 = no such error. 1 = an address range error occurred during the memory access.
2	ITERR	RC	0	IBUS timeout error: IBUS timeout error is occurred when the IBUS is stalled. 0 = no such error. 1 = IBUS timeout error.
1	IBERR	RC	0	IBUS bus error: IBUS bus error is occurred when VR4120A accesses to the RFU area in the IBUS target address space (see **1.9 Memory Map** section). 0 = no such error. 1 = a bus error occurred during the IBUS master access.
0	CBERR	RC	0	CPU (VR4120A) bus error: VR4120A bus error includes the illegal bus command, illegal data align, illegal bust size, and illegal access to the RFU area in register space. 0 = no such error. 1 = VR4120A bus error.

Remark To clear this register, the VR4120A must read the byte contained the CBERR register.

3.2.7 S_NER (NMI Enable Register)

The NMI enable register “S_NER” is a read-write and 32-bit word-aligned register. S_NER enables NMI for each corresponding incident. A enable bit, which locates in the same bit location to a corresponding bit in S_NSR, controls interruption triggered by the incident. If a bit of this register is reset to 0, the corresponding bit of the S_NSR is disabled. If it is set to 1, the corresponding bit is enabled. When the enable bit is set and the bit in S_NSR is set, system controller asserts interrupt signal to VR4120A. S_NER is initialized to 0 at reset and contains the following fields:

Bits	Field	R/W	Default	Description
31:6	Reserved	R/W	0	Hardwired to 0.
5	IRERRE	R/W	0	Illegal internal register access error enable: 1 = enable. 0 = disable.
4	EXTNMIE	R/W	0	External NMI enable: 1 = enable. 0 = disable.
3	MAERRE	R/W	0	Memory address error enable: 1 = enable. 0 = disable.
2	ITERRE	R/W	0	IBUS timeout error enable: 1 = enable. 0 = disable.
1	IBERRE	R/W	0	IBUS bus error enable: 1 = enable. 0 = disable.
0	CBERRE	R/W	0	CPU bus error enable: 1 = enable. 0 = disable.

3.2.8 S_VER (Version Register)

The version register “S_VER” is a read-only and 32-bit word-aligned register. S_VER indicates version number of the μPD98502. S_VER is initialized to 300H at reset and contains the following fields:

Bits	Field	R/W	Default	Description
31:16	Reserved	R	0	Hardwired to 0.
15:8	MAJOR	R	03H	Major revision. Hardwired to 03H.
7:0	MINOR	R	00H	Minor revision. Hardwired to 00H.

3.2.9 S_IOR (IO Port Register)

The IO port register “S_IOR” is a read-write and 32-bit word-aligned register. IO port register is used to indicate the status of software. Each bit of the following POM_OUT fields is connected to the external IO port (POM[7:0]) directly. S_IOR is initialized to 0 at reset and contains the following fields:

Bits	Field	R/W	Default	Description
31:8	Reserved	R/W	0	Hardwired to 0.
7	POM_OUT7	R/W	0	Set output level for POM7 pin: 0 = deassert POM7. 1 = assert POM7.
6	POM_OUT6	R/W	0	Set output level for POM6 pin: 0 = deassert POM6. 1 = assert POM6.
5	POM_OUT5	R/W	0	Set output level for POM5 pin: 0 = deassert POM5. 1 = assert POM5.
4	POM_OUT4	R/W	0	Set output level for POM4 pin: 0 = deassert POM4. 1 = assert POM4.
3	POM_OUT3	R/W	0	Set output level for POM3 pin: 0 = deassert POM3. 1 = assert POM3.
2	POM_OUT2	R/W	0	Set output level for POM2 pin: 0 = deassert POM2. 1 = assert POM2.
1	POM_OUT1	R/W	0	Set output level for POM1 pin: 0 = deassert POM1. 1 = assert POM1.
0	POM_OUT0	R/W	0	Set output level for POM0 pin: 0 = deassert POM0. 1 = assert POM0.

3.2.10 S_WRCR (Warm Reset Control Register)

The warm reset control register “S_WRCR” is a write-only and 32-bit word-aligned register. S_WRCR generates warm-reset request to USB Controller, Ethernet Controller, ATM Cell Processor, UART, and PCI Controller independently. S_WRCR is initialized to 0 at reset and contains the following fields:

Bits	Field	R/W	Default	Description
31:6	Reserved	W	0	Hardwired to 0.
5	PCIWR	W	0	Warm reset request for PCI Controller: 0 = do nothing. 1 = perform warm reset.
4	UARTWR	W	0	Warm reset request for UART: 0 = do nothing. 1 = perform warm reset.
3	MAC2WR	W	0	Warm reset request for Ethernet Controller #2: 0 = do nothing. 1 = perform warm reset.
2	ATMWR	W	0	Warm reset request for ATM Cell Processor: 0 = do nothing. 1 = perform warm reset.
1	MACWR	W	0	Warm reset request for Ethernet Controller #1: 0 = do nothing. 1 = perform warm reset.
0	USBWR	W	0	Warm reset request for USB Controller: 0 = do nothing. 1 = perform warm reset.

Remark All fields in this register will read back as 0.

3.2.11 S_WRSR (Warm Reset Status Register)

The warm reset status register "S_WRSR" is a read-only and 32-bit word-aligned register. S_WRSR indicates the response from USB Controller, Ethernet Controller, ATM Cell Processor, UART, and PCI Controller independently. S_WRSR is initialized to 0 at reset and contains the following fields:

Bits	Field	R/W	Default	Description
31:6	Reserved	R	0	Hardwired to 0.
5	PCIWRST	R	0	Indicates warm reset status from PCI Controller: 0 = PCI Controller is busy to perform the warm reset. 1 = warm reset has been done. PCI Controller is ready.
4	UARTWRST	R	0	Indicates warm reset status from UART: 0 = UART is busy to perform the warm reset. 1 = warm reset has been done. UART is ready.
3	MAC2WRST	R	0	Indicates warm reset status from Ethernet Controller #2: 0 = Ethernet Controller #2 is busy to perform the warm reset. 1 = warm reset has been done. MAC Ethernet Controller #2 is ready.
2	ATMWRST	R	0	Indicates warm reset status from ATM Cell Processor: 0 = ATM Cell Processor is busy to perform the warm reset. 1 = warm reset has been done. ATM Cell Processor is ready.
1	MACWRST	R	0	Indicates warm reset status from Ethernet Controller #1: 0 = Ethernet Controller #1 is busy to perform the warm reset. 1 = warm reset has been done. MAC Ethernet Controller #1 is ready.
0	USBWRST	R	0	Indicates warm reset status from USB Controller: 0 = USB Controller is busy to perform the warm reset. 1 = warm reset has been done. USB Controller is ready.

3.2.12 S_PWCR (Power Control Register)

The power control register “S_PWCR” is a read-write and 32-bit word-aligned register. S_PWCR requests to keep the idle state for USB Controller, Ethernet Controller, ATM Cell Processor, and PCI Controller by setting following IDRQ fields. VR4120A must request these blocks to keep the idle state and check their acknowledgement by reading the power status register “S_PWSR” prior to perform suspend by setting following STOP fields. S_PWCR is initialized to 0 at reset and contains the following fields:

Bits	Field	R/W	Default	Description
31:22	Reserved	R/W	0	Hardwired to 0.
21	PCISTOP	R/W	0	Suspend request for PCI Controller: 0 = enable system clock for PCI Controller. 1 = disable system clock for PCI Controller.
20	Reserved	R/W	0	Hardwired to 0.
19	MAC2STOP	R/W	0	Suspend request for Ethernet Controller #2: 0 = enable system clock for Ethernet Controller #2. 1 = disable system clock for Ethernet Controller #2.
18	ATMSTOP	R/W	0	Suspend request for ATM Cell Processor: 0 = enable system clock for ATM Cell Processor. 1 = disable system clock for ATM Cell Processor.
17	MACSTOP	R/W	0	Suspend request for Ethernet Controller #1: 0 = enable system clock for Ethernet Controller #1. 1 = disable system clock for Ethernet Controller #1.
16	USBSTOP	R/W	0	Suspend request for USB Controller: 0 = enable system clock for USB Controller. 1 = disable system clock for USB Controller.
15:6	Reserved	R/W	0	Hardwired to 0.
5	PCIIDRQ	R/W	0	Idle request for PCI Controller: 0 = do nothing. 1 = request to keep the idle state.
4	Reserved	R/W	0	Hardwired to 0.
3	MAC2IDRQ	R/W	0	Idle request for Ethernet Controller #2: 0 = do nothing. 1 = request to keep the idle state.
2	ATMIDRQ	R/W	0	Idle request for ATM Cell Processor: 0 = do nothing. 1 = request to keep the idle state.
1	MACIDRQ	R/W	0	Idle request for Ethernet Controller #1: 0 = do nothing. 1 = request to keep the idle state.
0	USBIDRQ	R/W	0	Idle request for USB Controller: 0 = do nothing. 1 = request to keep the idle state.

Remark Before accesses to this register, the VR4120A must flush the internal write command buffer by reading the IBUS target.

3.2.13 S_PWSR (Power Status Register)

The power status register “S_PWSR” is a read-only and 32-bit word-aligned register. The IDLE field in S_PWSR indicates the status that it is ready to suspend. The WKUP filed in S_PWSR indicates the wakeup request. When a bit of IDLE fields gets 1, VR4120A can disable the system clock for the corresponding device by setting the STOP field in S_PWCR. When a bit of WKUP fields in S_PWSR gets 1, VR4120A must enable the system clock for the corresponding device by resetting the STOP field in S_PWCR. The S_ PWSR is initialized to 0 at reset and contains the following fields:

Bits	Field	R/W	Default	Description
31:14	Reserved	R	0	Hardwired to 0.
13	PCIWKUP	R	0	Indicates the wakeup request from PCI Controller: 0 = no wakeup request is pending. 1 = wakeup request is pending.
12	Reserved	R	0	Hardwired to 0.
11	MAC2WKUP	R	0	Indicates the wakeup request from Ethernet Controller #2: 0 = no wakeup request is pending. 1 = wakeup request is pending.
10	ATMWKUP	R	0	Indicates the wakeup request from ATM Cell Processor: 0 = no wakeup request is pending. 1 = wakeup request is pending.
9	MACWKUP	R	0	Indicates the wakeup request form Ethernet Controller #1: 0 = no wakeup request is pending. 1 = wakeup request is pending.
8	USBWKUP	R	0	Indicates the wakeup request from USB Controller: 0 = no wakeup request is pending. 1 = wakeup request is pending.
7:6	Reserved	R	0	Hardwired to 0.
5	PCIIDLE	R	0	Indicates the idle status in PCI Controller: 0 = not in idle state. It means PCI Controller is not ready to suspend. 1 = in idle state. It means PCI Controller is ready to suspend.
4	Reserved	R	0	Hardwired to 0.
3	MAC2IDLE	R	0	Indicates the idle status in Ethernet Controller #2: 0 = not in idle state. It means Ethernet Controller #2 is not ready to suspend. 1 = in idle state. It means Ethernet Controller #2 is ready to suspend.
2	ATMIDLE	R	0	Indicates the idle status in ATM Cell Processor: 0 = not in idle state. It means ATM Cell Processor is not ready to suspend. 1 = in idle state. It means ATM Cell Processor is ready to suspend.
1	MACIDLE	R	0	Indicates the idle status in Ethernet Controller #1: 0 = not in idle state. It means Ethernet Controller #1 is not ready to suspend. 1 = in idle state. It means Ethernet Controller #1 is ready to suspend.
0	USBIDLE	R	0	Indicates the idle status in USB Controller: 0 = not in idle state. It means USB Controller is not ready to suspend. 1 = in idle state. It means USB Controller is ready to suspend.

3.3 CPU Interface

The system controller provides the direct interface for the VR4120A using the 32-bit SysAD bus operated at 100 MHz or 66 MHz.

3.3.1 Overview

- Connects to the VR4120A CPU bus “SysAD bus” directly.
- Supports all VR4120A bus cycles at 66 MHz or 100 MHz.
- Supports data rate D only.
- Supports sequential ordering only.
- 4-word (16-byte) x 4-entry write command buffer.
- Little-endian or big-endian byte order.
- Not support 8-word burst R/W on SysAD bus

3.3.2 Data rate control

The VR4120A-to-system controller data rate is programmable by setting the EP field (bits 27:24) of the configuration register in the VR4120A. The controller supports only data rate D. Thus this block does not support AD mode.

3.3.3 Burst size control

This block - to - VR4120A burst data size is determined by the OSysCMD[2:0] signal on SysAD bus. It is programmable in the IB bit (bits 5) of the VR4120A’s Configuration Register (Please see the section **2.4.5.8 Config register (16)**). The μPD98502 support 4-word burst mode only. Please set “0” to IB bit of VR4120’s Configuration Register.

3.3.4 Address decoding

The controller latches the address on the SysAD bus. It then decodes the address and SysCmd signals to determine the transaction type. Ten address ranges can be decoded:

- One range for external boot PROM or flash.
- One range for external SDRAM.
- One range for system controller’s internal configuration registers.

Boot PROM/flash is mapped according to its size. System controller’s internal registers are fixed at base address 1000_0000H, to allow the VR4120A to access them during boot, before they have been configured. All other decode ranges are programmable.

3.3.5 Endian conversion

The BE bit in the configuration register in the VR4120A specifies the byte ordering at reset. BE = 0 configures little-endian order, BE = 1 configures big-endian order. The endian mode is controlled by “BIG” signal. VR4120A interface of the system controller supports both big- and little-endian byte ordering on the SysAd bus by using endian converter. All of the other interfaces in the system controller operate only in little-endian mode.

When the VR4120A is operated in the big-endian mode (external BIG pin is high), the system controller provides the two endian conversion methods controlled by external ENDCEN pin. If ENDCEN pin is low, the system controller performs the data swap on the SysAD bus (see **Table 3-2. Endian Translation Table in Endian Converter** for data swap mode). If ENDCEN pin is high, the system controller performs the address swap on the SysAD bus (the detail is described in the **Table 3-2. Endian Translation Table in Endian Converter** for the address swap mode).

Table 3-1. Endian Configuration Table

BIG pin	ENDCEN pin	Status register RE field in VR4120A	Endian in VR4120A	Endian in system controller	Endian converter operation
0	0	0	LITTLE	LITTLE	Transparent
0	1	0	LITTLE	LITTLE	Transparent
1	0	0	BIG	LITTLE	Data swap mode
1	1	0	BIG	LITTLE	Address swap mode

Remark VR4120A does not support reverse endian mode.

Table 3-2. Endian Translation Table in Endian Converter

CPU access type		BIG	ENDCEN	Before translation SysAD[1:0]	After translation SysAD[1:0]	Notes
Block	2-word 4-word	1	1	00	00	Valid
				01	01	Invalid
				10	10	Invalid
				11	11	Invalid
Single	1-byte	1	1	00	11	Valid
				01	10	Valid
				10	01	Valid
				11	00	Valid
Single	2-byte	1	1	00	10	Valid
				01	11	Invalid
				10	00	Valid
				11	01	Invalid
Single	3-byte	1	1	00	01	Valid
				01	00	Valid
				10	11	Invalid
				11	10	Invalid
Single	4-byte	1	1	00	00	Valid
				01	01	Invalid
				10	10	Invalid
				11	11	Invalid

CPU access type	BIG	ENDCEN	Before translation SysAD[1:0]	After translation SysAD[1:0]	Note
Any	1	0	[31:24][23:16][15:8][7:0]	[7:0][15:8][23:16][31:24]	Byte swap

3.3.6 I/O performance

The following table indicates the I/O performance accessing from the VR4120A through the system controller.

W/R	Target area	Burst length	Access latency [VR4120A clocks]
R	IBUS target	1	24
R	IBUS target	2	24-1
R	IBUS target	4	27-1-1-1
R	Internal register (except UART)	1	7
R	Internal register (except UART)	2	Invalid
R	Internal register (except UART)	4	Invalid
R	Internal UART register	1	14 (depend UART source clock)
R	Internal UART register	2	Invalid
R	Internal UART register	4	Invalid
W	IBUS target	1	23
W	IBUS target	2	23
W	IBUS target	4	23-2-1-2
W	Internal write command FIFO	1	6-1(wait)
W	Internal write command FIFO	2	6-1
W	Internal write command FIFO	4	6-1-1-1
W	Internal register (except UART)	1	7
W	Internal register (except UART)	2	Invalid
W	Internal register (except UART)	4	Invalid
W	Internal UART register	1	15 (depend UART source clock)
W	Internal UART register	2	Invalid
W	Internal UART register	4	Invalid

Remarks 1. BUS frequency: SysAD = 100 MHz , IBUS = 66 MHz

2. The latency value accessing to the IBUS target does not include IBUS bus arbitration cycle (about 6 CPU clocks).

3.4 Memory Interface

The VR4120A accesses memory attached to the controller in the normal way, by addressing the memory space.

3.4.1 Overview

- 66 MHz or 100 MHz memory bus
- Supports up to 32 MB base memory range for SDRAM
- Supports up to 8 MB write-protectable boot memory range for PROM/flash
- On-chip programmable SDRAM refresh controller
- 4 words (16 bytes) prefetch data buffer (memory-to-VR4120A)
- PROM/flash data signals multiplexed on SDRAM data signals
- Programmable memory bus arbitration priority
- Programmable address ranges for the memory
- Programmable RAS-CAS delay (2, 3, 4 clocks)
- Programmable CAS latency (2, 3 clocks)
- Endian converter on IBUS slave I/F
- Don’t supports 8-word burst R/W from SysAD bus

3.4.2 Memory regions

The controller connects to memory directly and manages the addresses, data and control signals for the following address ranges:

- One boot PROM/flash range (programmable)
- One system memory range (programmable)

The following types of memory modules as an example but not limited to, can be used:

- Flash with variable data size (8, 16, 32 bits) can be used in the boot ROM.
- PROM with variable data size (8, 16, 32 bits) can be used in the boot ROM.
- SDRAM can be used in the system memory.

Boot ROM can be populated with PROM or 85-ns flash chips. Prior to accessing PROM/flash, software must configure this address range. The system memory can be populated with SDRAM chips. The system memory is used for the RTOS, M/W and F/W. Prior to accessing SDRAM, software must configure this address range.

3.4.3 Memory signal connections

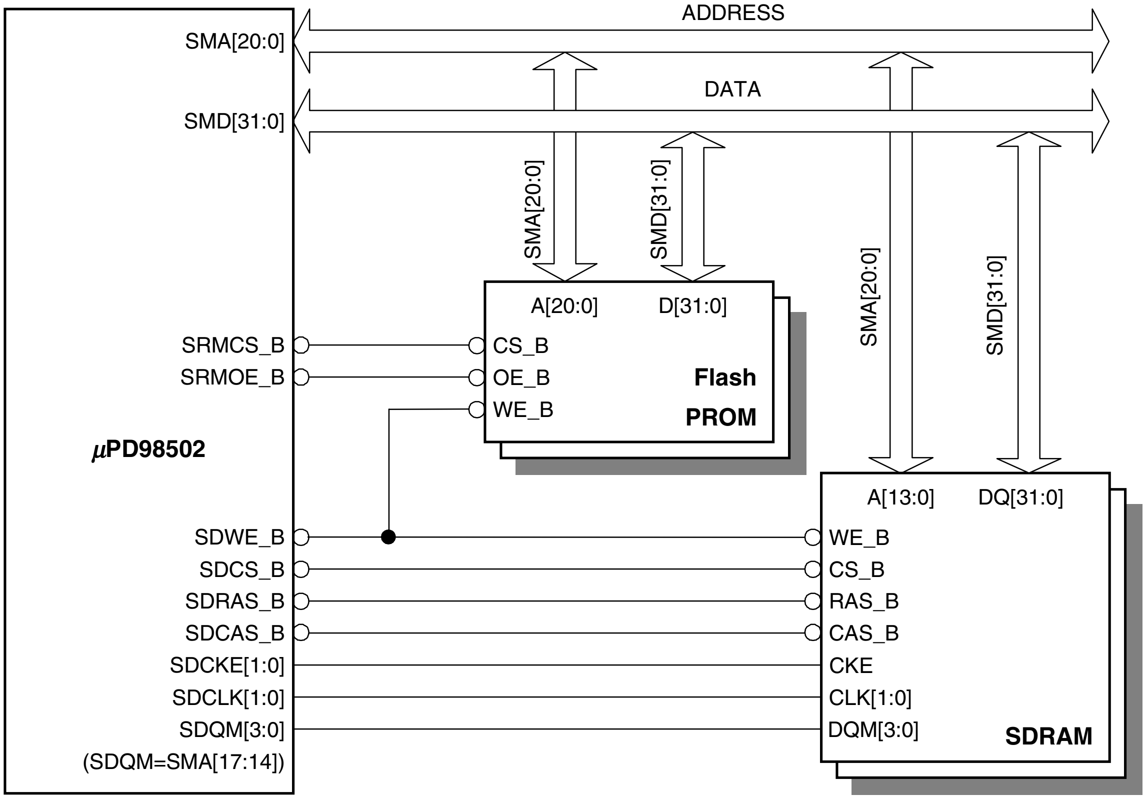


Table 3-3. External Pin Mapping

External Pin		Access to ROM	Access to SDRAM
Name	Bits		
SMA	[13:0]	A[13:0]	A[13:0]
	[17:14]	A[17:14]	SDQM[3:0]
	[20:18]	A[20:18]	
SMD	[31:0]	D[31:0]	DQ[31:0]
SDCS_B		---	SDCS_B
SDRAS_B		---	SDRAS_B
SDCAS_B		---	SDCAS_B
SDWE_B		SDWE_B	SDWE_B
SDCKE	[1:0]	---	SDCKE[1:0]
SDCLK	[1:0]	---	SDCLK[1:0]
SRMCS_B		SRMCS_B	---
SRMOE_B		SRMOE_B	---
RMSL	[1:0]	---	---

Remark RMSL signal determines boot memory data bus size.

3.4.4 Memory performance

The latency of memory accesses is determined by memory type, speed and prefetch scheme. Following lists some examples of access latencies. 66-MHz or 100-MHz memory-bus clock is required for each transfer of a 4-word (16-byte) CPU instruction-cache line fill. The first number in the “SysAD clocks” column is for the first word; the remaining numbers are for the subsequent words. The most common combinations are shown.

Table 3-4. Examples of Memory Performance (4-word-burst access from CPU)

Memory type	Bank-interleaved	Page hit	R/W	Prefetch hit	Access latency view from CPU [SysAD clocks]
SDRAM,10 ns	No	Yes	R	Yes	6-1-1-1
SDRAM,10 ns	No	Yes	R	No	14-1-1-1
SDRAM,10 ns	No	Yes	W	N/A	9-1-1-1
Flash, 85 ns (32-bit bus)	No	No	R	N/A	19-12-1-12
Flash, 85 ns (16-bit bus)	No	No	R	N/A	31-24-24-24
Flash, 85 ns (8-bit bus)	No	No	R	N/A	55-48-48-48
Flash, 85 ns	No	No	W	N/A	18 (single access only)
PROM	No	No	R	N/A	19-12-12-12

Remarks 1. SDRAM configuration: RCD = 3, CL = 2, SDCLK = 100 MHz, FAT = 10

2. BUS frequency: SysAD = 100 MHz, IBUS = 66 MHz

3. Read performance is calculated by counting the rising edge for CPU clock where the read command is issued by the CPU. Because the CPU issues write data with no wait-states once the write command is issued, the numbers in the table represent the rate at which data is written to memory. The sum of the numbers represents the number of cycles between when the write operation was issued and when the next CPU memory operation can begin.

4. The burst-write access to the flash/ROM is invalid. The CPU can access to the flash/ROM using single access only

Table 3-5. Examples of Memory Performance (4-word-burst access from IBUS Master)

Memory type	Bank-interleaved	Page hit	R/W	Prefetch hit	Access latency view from IBUS [IBUS clocks]
SDRAM,10 ns	No	Yes	R	N/A	18-1-1-1
SDRAM,10 ns	No	Yes	W	N/A	12-1-1-1
Flash, 85 ns (32-bit bus)	No	No	R	N/A	45-1-1-1
Flash, 85 ns (16-bit bus)	No	No	R	N/A	77-1-1-1
Flash, 85 ns (8-bit bus)	No	No	R	N/A	141-1-1-1
Flash, 85 ns	No	No	W	N/A	Invalid
PROM	No	No	R	N/A	45-1-1-1

Remarks 1. SDRAM configuration: RCD = 3, CL = 2, SDCLK = 100 MHz, FAT = 10

2. BUS frequency: SysAD = 100 MHz, IBUS = 66 MHz

3. Above access latency doses not include the IBUS arbitration cycle (4 IBUS clocks).

4. Any write access to the flash/ROM is prohibited. If the IBUS master perform the write access to the flash/ROM, The IBUS bus error will be occurred.

3.4.5 RMMDR (ROM Mode Register)

The ROM mode register "RMMDR" is a read-write and 32-bit word-aligned register. RMMDR is used to setup the PROM/flash memory interface. RMMDR is initialized to 0 at reset and contains the following fields:

Bits	Field	R/W	Default	Description
31:9	Reserved	R/W	0	Hardwired to 0.
8	WM	R/W	0	Write mask: 0 = masked. Flash data is protected from unintentional write. 1 = not masked. Flash data is not protected.
7:2	Reserved	R/W	0	Hardwired to 0.
1:0	FSM	R/W	00	Flash/PROM size model: 00 = mode1 (4-MByte mode) 01 = mode2 (8-MByte mode) 10 = mode3 (1-MByte mode) 11 = mode4 (2-MByte mode)

Remark Don't change the value on the FSM field after setting a value into the FSM field.

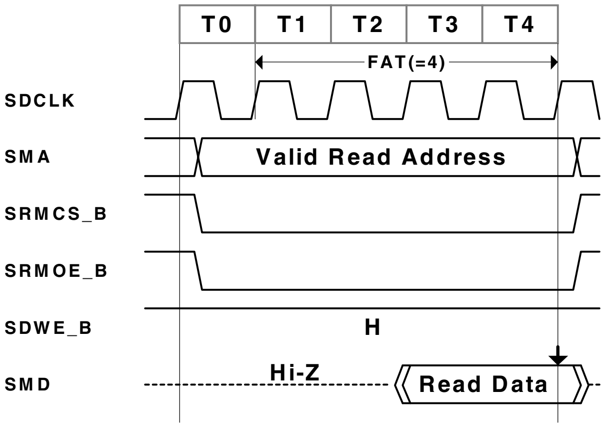
3.4.6 RMATR (ROM Access Timing Register)

The ROM access timing register "RMATR" is a read-write and 32-bit word-aligned register. RMATR is used to set the access time in the PROM/flash interface. RMATR is initialized to 0 at reset and contains the following fields:

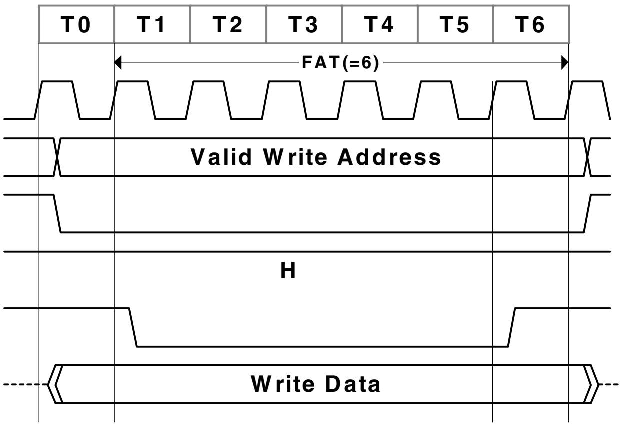
Bits	Field	R/W	Default	Description		
31:3	Reserved	R/W	0	Hardwired to 0.		
2:0	FAT	R/W	000	Flash/PROM access timing for normal ROM:		
				000 = 18 clocks	66 MHz: 272.4 ns	100 MHz: 180 ns
				001 = 4 clocks	66 MHz: 60.6 ns	100 MHz: 40 ns
				010 = 6 clocks	66 MHz: 90.9 ns	100 MHz: 60 ns
				011 = 8 clocks	66 MHz: 121.2 ns	100 MHz: 80 ns
				100 = 10 clocks	66 MHz: 151.5 ns	100 MHz: 100 ns
				101 = 12 clocks	66 MHz: 181.8 ns	100 MHz: 120 ns
				110 = 14 clocks	66 MHz: 212.1 ns	100 MHz: 140 ns
				111 = 16 clocks	66 MHz: 242.4 ns	100 MHz: 160 ns

Remark ROM access timing is depended on the system clock frequency.

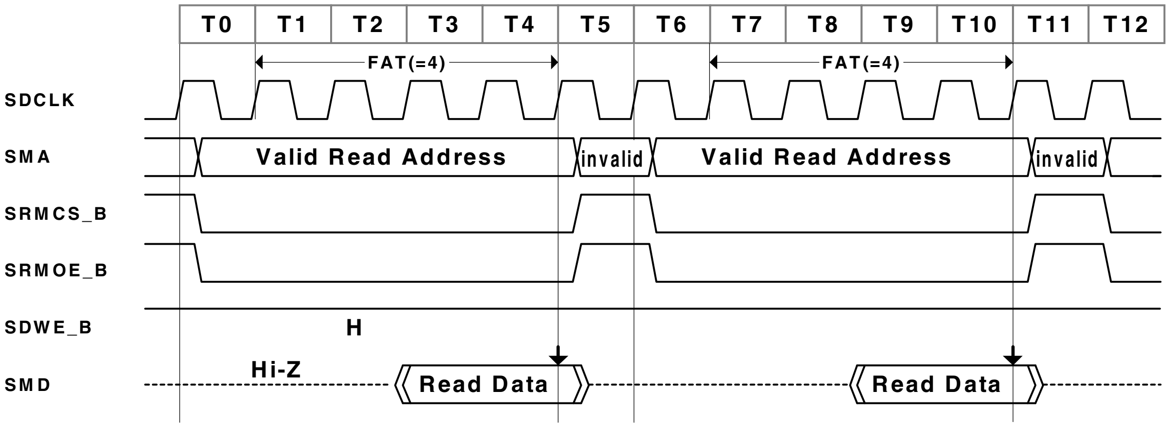
Normal ROM Read Cycle



FLASH Memory Write Cycle



ROM Burst Read Cycle



3.4.7 SDMDR (SDRAM Mode Register)

The SDRAM mode register “SDMDR” is a read-write and 32-bit word-aligned register. SDMDR is used to setup the SDRAM interface. SDMDR is initialized to 330H at reset and contains the following fields:

<table>
<tr><th>Bits</th><th>Field</th><th>R/W</th><th>Default</th><th colspan="3">Description</th></tr>
<tr><td>31:10</td><td>Reserved</td><td>R/W</td><td>0</td><td colspan="3">Hardwired to 0.</td></tr>
<tr><td rowspan="2">9:8</td><td rowspan="2">RCD</td><td rowspan="2">R/W</td><td rowspan="2">11</td><td colspan="3">SDRAM RAS-CAS delay:</td></tr>
<tr><td>00 = reserved
01 = 2 clocks
10 = 3 clocks
11 = 4 clocks (default)</td><td>
66 MHz: 30.3 ns
66 MHz: 45.5 ns
66 MHz: 60.6 ns</td><td>
100 MHz: 20 ns
100 MHz: 30 ns
100 MHz: 40 ns</td></tr>
<tr><td>7</td><td>Reserved</td><td>R/W</td><td>0</td><td colspan="3">Hardwired to 0.</td></tr>
<tr><td>6:4</td><td>LTMD</td><td>R/W</td><td>011</td><td colspan="3">SDRAM CAS latency:
000 = reserved
001 = reserved
010 = 2
011 = 3 (default)
1xx = reserved</td></tr>
<tr><td>3</td><td>Reserved</td><td>R/W</td><td>0</td><td colspan="3">Hardwired to 0.</td></tr>
<tr><td>2:0</td><td>BL</td><td>R/W</td><td>000</td><td colspan="3">SDRAM burst length:
000 = 1 (default)
001 = reserved
010 = reserved
011 = reserved
1xx = reserved</td></tr>
</table>

Remarks 1. RAS-CAS delay time is depended on the system clock frequency.

2. Don’t change the value on this register after using the SDRAM.

3. The initialization by setting this register must be done before using the SDRAM.

3.4.8 SDTSR (SDRAM Type Selection Register)

The SDRAM type selection register “SDTSR” is a read-write and 32-bit word-aligned register. SDTSR is used to setup the type of SDRAM. SDTSR is initialized to 0 at reset and contains the following fields:

Bits	Field	R/W	Default	Description	
31:10	Reserved	R/W	0	Hardwired to 0.	
9:8	SDS	R/W	00	Total SDRAM size:	
				00 = 4 MBytes (default) 01 = 8 MBytes 10 = 16 MBytes 11 = 32 MBytes	003F_FFFFH – 0000_0000H 007F_FFFFH – 0000_0000H 00FF_FFFFH – 0000_0000H 01FF_FFFFH – 0000_0000H
7	BTM	R/W	0	Number of bank: 0 = 1 or 2 banks (default) 1 = 3 or 4 banks	
6:4	RAB	R/W	000	Total number of SDRAM address bits (RAS + CAS) (except bank select pins): 000 = 17 bits (default) 001 = 18 bits 010 = 19 bits 011 = 20 bits 100 = 21 bits 101 = 22 bits 110 = reserved 111 = reserved	
3:2	Reserved	R/W	0	Hardwired to 0.	
1:0	CAB	R/W	00	Number of column address bits (except bank select pins): 00 = 7 bits (default) 01 = 8 bits 10 = 9 bits 11 = 10 bits	

Remark Don’t set the reserved value to each field in this register.

3.4.9 SDPTR (SDRAM Precharge Timing Register)

The SDRAM precharge timing register “SDPTR” is a read-write and 32-bit word-aligned register. SDPTR is used to set the precharge timing for the SDRAM controller. SDPTR is initialized to 142H at reset and contains the following fields:

Bits	Field	R/W	Default	Description		
31:9	Reserved	R/W	0	Hardwired to 0.		
8	DPL	R/W	1	Input data -> precharge command timing (t_{DPL}):		
				0 = 1 clock	66 MHz: 15.2 ns	100 MHz: 10 ns
				1 = 2 clocks (default)	66 MHz: 30.3 ns	100 MHz: 20 ns
7	Reserved	R/W	0	Hardwired to 0.		
6:4	APT	R/W	100	Active command -> precharge command timing (t_{RAS}):		
				000 = 4 clocks	66 MHz: 60.6 ns	100 MHz: 40 ns
				001 = 5 clocks	66 MHz: 75.7 ns	100 MHz: 50 ns
				010 = 6 clocks	66 MHz: 90.9 ns	100 MHz: 60 ns
				011 = 7 clocks	66 MHz: 106.0 ns	100 MHz: 70 ns
				100 = 8 clocks (default)	66 MHz: 121.2 ns	100 MHz: 80 ns
				101 = reserved		
				110 = reserved		
				111 = reserved		
3:2	Reserved	R/W	0	Hardwired to 0.		
1:0	PAT	R/W	10	Precharge command -> active command timing (t_{RP}):		
				00 = 2 clocks	66 MHz: 30.3 ns	100 MHz: 20 ns
				01 = 3 clocks	66 MHz: 45.5 ns	100 MHz: 30 ns
				10 = 4 clocks (default)	66 MHz: 60.6 ns	100 MHz: 40 ns
				11 = reserved		

Remark Don’t set the reserved value to each field in this register.

3.4.10 SDRMR (SDRAM Refresh Mode Register)

The SDRAM refresh mode register “SDRMR” is a read-write and 32-bit word-aligned register. SDRMR is used to initialize the SDRAM refresh controller. SDRMR is initialized to 200H at reset and contains the following fields:

Bits	Field	R/W	Default	Description
31:16	Reserved	R/W	0	Hardwired to 0.
15:0	RCSET	R/W	0200H	Reload value for SDRAM refresh timer counter. This value, in system clock ticks, is automatically reloaded into the refresh timer counter after the counter reached zero. The refresh timer counter counts down from this value. Thus, time of the count cycle corresponds to 1 plus this filed value. The default value (200H = 512) is the refresh rate for an SDRAM chip that requires 4096 refresh cycles every 32 ms (ex. one refresh every 7.8125 µs) for system clock running at 66 MHz. This is very conservative but it allows for successful boot, after which the reload value can be increased. RCSET [7:0] is hardwired to 0, thus the timer value less than 100H cannot be set this field. If such value is set into this field, the default value “200H” is automatically loaded.

3.4.11 SDRCR (SDRAM Refresh Timer Count Register)

The SDRAM refresh timer count register “SDRCR” is a read-only and 32-bit word-aligned register. SDRCR is a 16-bit timer that causes an SDRAM refresh when it expires. The SDRAM refresh controller automatically reloads this free-running timer. SDRCR is initialized to 200H at reset and contains the following fields:

Bits	Field	R/W	Default	Description
31:16	Reserved	R	0	Hardwired to 0.
15:0	RCC	R	0200H	This field indicates the current value of SDRAM refresh timer.

3.4.12 MBCR (Memory Bus Control Register)

The memory bus control register “MBCR” is a read-write and 32-bit word-aligned register. MBCR is used to select priority for either VR4120A or IBUS to access memory. The VR4120A can assign higher priority to CPU request for memory than IBUS request or assign equal priority to VR4120A and IBUS request for memory. MBCR is initialized to 0 at reset and contains the following fields:

Bits	Field	R/W	Default	Description
31:1	Reserved	R/W	0	Hardwired to 0.
0	BPR	R/W	0	Priority for memory access: 0: SysAD (CPU) > Refresh > IBUS (default) The memory arbiter allows one CPU memory transaction or refresh operation even through the current memory access from IBUS is not yet completed. CPU will be allowed to perform its memory access when the current word count of burst access from IBUS reaches each 16 words or 32 words. Refresh operation is also allowed to interrupt IBUS access. 1: SysAD (CPU) = Refresh = IBUS Three operations are served in order of occurrence. Yet CPU will be granted to access memory when three requests occur simultaneously.

3.4.13 Boot ROM

The system controller supports up to 8 MB of boot memory. This memory must be populated with either of the following two types of memory devices: PROM/flash memory.

3.4.13.1 Boot ROM configuration and address ranges

Boot ROM can be populated with PROM or 85-ns flash chips, and it must have an access time of 200 ns or less. The system controller supports 8, 16 and 32-bit boot ROM at locations 1F80_0000H through 1FFF_FFFFH in the physical memory space on VR4120A. The boot ROM does not support VR4120A cache operations.

Table 3-6. Boot-ROM Size Configuration at Reset

RMMDR.FSM	Boot ROM Size	Address Range
00	4 MB	1FC0_0000H through 1FFF_FFFH
01	8 MB	1F80_0000H through 1FFF_FFFH
10	1 MB	1FC0_0000H through 1FCF_FFFH
11	2 MB	1FC0_0000H through 1FDF_FFFH

The controller asserts the flash/ROM chip select (SRMCS_B) in the address range 1F80_0000H through 1FFF_FFFFH. When writes are performed to the ROM/flash memory space, the controller asserts SDWE_B in conjunction with SRMCS_B. When reads are performed, the controller asserts SRMOE_B in conjunction with SRMCS_B. If the VR4120A attempts to access boot ROM addresses outside the defined size of the flash/ROM, the controller returns 0 with the data error bit set on SysCMD [0]. In addition, the NMI status register “S_NSR” is updated and NMI is asserted to VR4120A, if the interrupt is enabled in the NMI enable register “S_NER”.

3.4.13.2 Flash memory write-protection

The flash memory can be protected in software. Software protection is implemented by programming the WM field in ROM mode register “RMMDR”.

3.4.13.3 Flash memory operations

Flash memory I/F has 3 modes for each Flash data BUS size, that is 8,16 and 32 bits. And on the case of each Bus size, the way of causing write cycle will be changed. The flash memory can be programmed using following write cycle sequence by VR4120A. The following commands are example of operations for the AMD AM29LV800BT flash memory (using Byte Mode) at 32-bit Flash data Bus mode in System Controller.

Table 3-7. Command Sequence

(a) Program Command Sequence (4 Write Cycles)

1st Write		2nd Write		3rd Write		4th Write		5th Write		6th Write	
A	1FC0_2AA8H	A	1FC0_1554H	A	1FC0_2AA8H	A	PA*	A		A	
D	AAAA_AAAAH	D	5555_5555H	D	A0A0_A0A0H	D	PD*	D		D	

(b) Chip Erase Command Sequence (6 Write Cycles)

1st Write		2nd Write		3rd Write		4th Write		5th Write		6th Write	
A	1FC0_2AA8H	A	1FC0_1554H	A	1FC0_2AA8H	A	1FC0_2AA8H	A	1FC0_1554H	A	1FC0_2AA8H
D	AAAA_AAAAH	D	5555_5555H	D	8080_8080H	D	AAAA_AAAAH	D	5555_5555H	D	1010_1010H

(c) Sector Erase Command Sequence (6 Write Cycles)

1st Write		2nd Write		3rd Write		4th Write		5th Write		6th Write	
A	1FC0_2AA8H	A	1FC0_1554H	A	1FC0_2AA8H	A	1FC0_2AA8H	A	1FC0_1554H	A	EA*
D	AAAA_AAAAH	D	5555_5555H	D	8080_8080H	D	AAAA_AAAAH	D	5555_5555H	D	3030_3030H

Remark A = memory write address
D = memory write data
PA = address of flash location to be programmed.
PD = data to be programmed at location PA.
EA = block address of flash location to be erased.

In case of Flash memory programming, please consider following system factors:

(1) Read cycle can't interrupt these write commands. Therefore, it is impossible for the μPD98502 to program Flash memory with fetching from Flash memory.

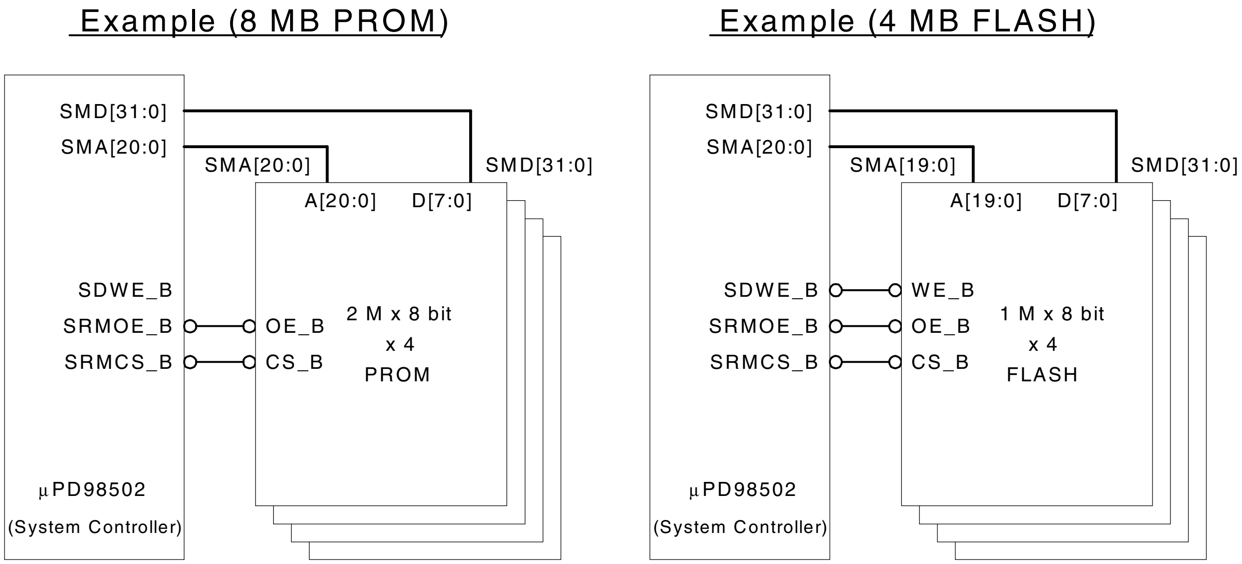
(2) These write commands for Flash memory will be change on following system factors.

- Flash manufacturer company: Write sequences are differing among each company
- Endian mode (there are 3 system endian modes; Little endian, Big endian with data swap mode, and Big endian with address swap mode)
- Flash data BUS size of Flash memory (ordinary Flash memory has both 8- and 16-bit BUS modes)
- Flash data BUS size of system controller (8, 16, 32 bits)

(3) Please make SMD and SMA signal outputs the same for write sequences.

3.4.1.4 Boot ROM signal connections

FLASH/ROM Configuration



Example (8 MB PROM)

Example (4 MB FLASH)

3.4.14 SDRAM

3.4.14.1 SDRAM address range

System memory can be populated with SDRAM chips, and it must have an access time of 10 ns or less. The system controller supports 16-Mbit or 64-Mbit and 128-Mbit SDRAM at locations 0000_0000H through 01FF_FFFFH in the physical memory space on VR4120A. The SDRAM supports VR4120A cache operations.

Table 3-8. SDRAM Size Configuration at Reset

SDMDR.SDS	SDRAM Size	Address Range
00	4 MB	0000_0000H through 003F_FFFH
01	8 MB	0000_0000H through 007F_FFFH
10	16 MB	0000_0000H through 00FF_FFFH
11	32 MB	0000_0000H through 01FF_FFFH

3.4.14.2 SDRAM device configurations

The controller supports the following 16-Mbit, 64-Mbit and 128-Mbit SDRAM organization. Following table indicates some of the SDRAM organizations supported for system memory.

Table 3-9. SDRAM Configurations Supported

Memory Size	SMA Address Bits Required	Organization (bank x word x bit)	Quantity
4 MB	12:0	2 x 0.5 M x 16	2
8 MB	12:0	4 x 0.5 M x 32	1
16 MB	13:0	4 x 1.0 M x 16	2
32 MB	13:0	4 x 2.0 M x 16	2

3.4.14.3 SDRAM burst-type and banks

The terms interleaved and bank have multiple meanings in the context of memory design using SDRAM chips. The meanings are:

- Banks (applied to memory modules and SDRAM chips in different ways): The banks referenced with respect to memory modules differ from the banks inside an SDRAM chip. For module, this controller does not support what identifies their bank.

- Burst Type (applied to SDRAM chips): The burst type of a single SDRAM chip is programmed in the chip's mode register to be either interleaved or sequential. This concept relates only to the word order in which data is read into and written out of the SDRAM chip. The concept does not relate to the number of words transferred in a given clock cycle. The burst type for all SDRAM chips attached to the μPD98502 is configured during the memory initialization procedure. The memory controller in the system controller does not support the interleaved burst mode and support only sequential burst mode.

3.4.1.4 SDRAM word ordering

Following table indicates the word-address order for a 4-word instruction-cache line fill from SDRAM. This order is determined by the SDRAM chips' burst type, which is programmed during the memory initialization procedure. The memory controller programs the burst type and word order the same for all SDRAM chips connected to it (in the system memory ranges). The term "sequential" in this table refers to the SDRAM burst type. Burst length depends only on the access type performed by the CPU.

Table 3-10. SDRAM Word Order for Instruction-Cache Line-Fill

Start Column Address A1.A0	SDRAM-Chip Burst Type	
	Sequential	Interleaved
00	0-1-2-3	Not supported
01	1-2-3-0	Not supported
10	2-3-0-1	Not supported
11	3-0-1-2	Not supported

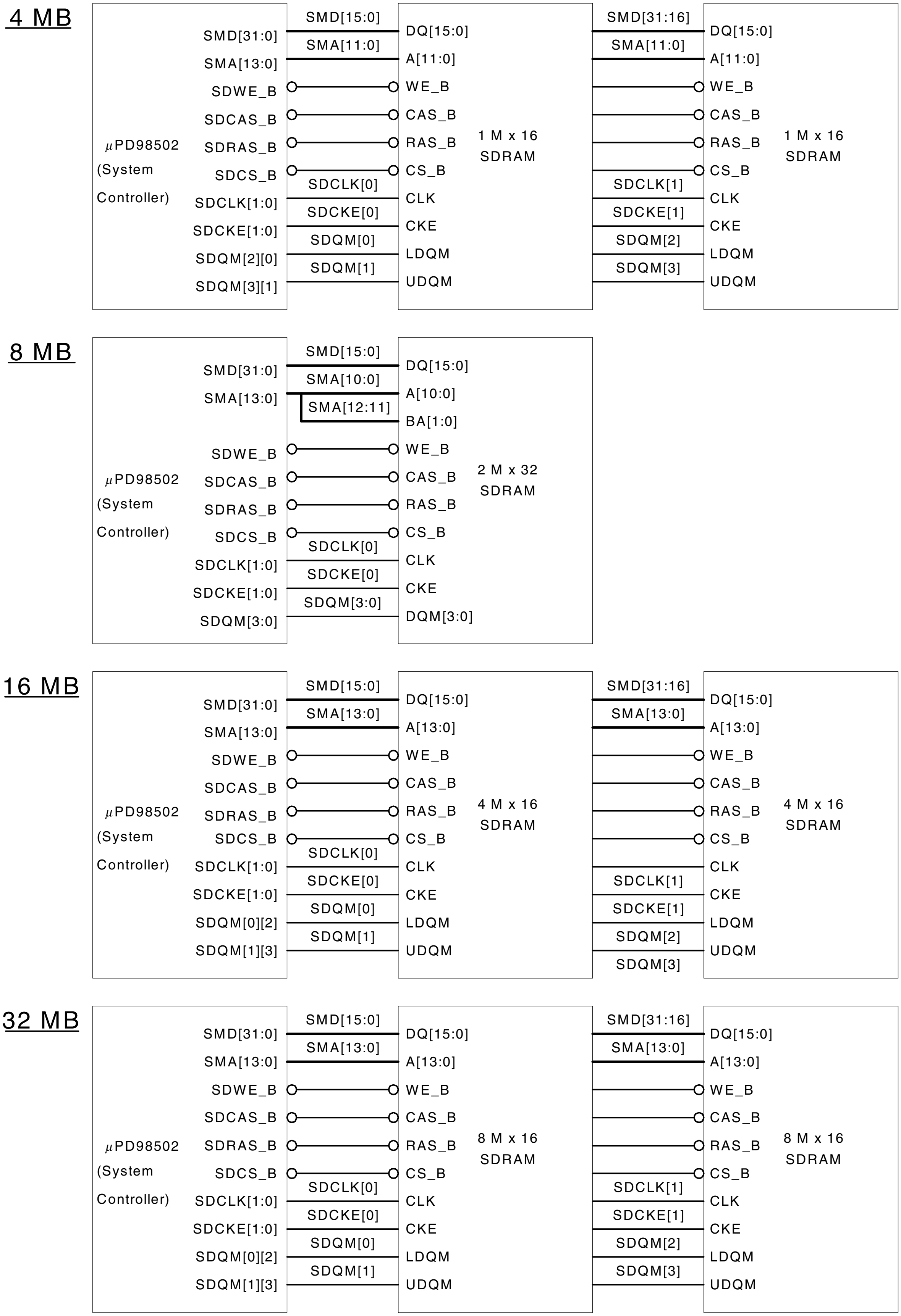
Remark The memory controller does not support the interleaved burst type for SDRAMs. It assumes that all SDRAMs are initialized to the sequential burst type, using a burst length of 4 words.

3.4.1.5 SDRAM signal connections

Following figure indicates an example of SDRAM signal connections. SMA [11] is the bank select signal. In command cycle, SMA [11] low selects bank A and SMA [11] high selects bank B. Both banks share the same SDCSB, SDRASB, SDCASB, and SDWEB signals.

The two banks of system memory behave as two halves of the address range, with the highest unmasked address bit controlling bank selection.

SDRAM Configuration



3.4.15 SDRAM refresh

The system controller supports CAS-Before-RAS (CBR) DRAM refresh to all SDRAM address ranges. The refresh clock is derived from the system clock; its rate is determined by programming the RCR filed in the SDRAM Refresh Mode Register “SDRMR”.

The refresh logic requests access to SDRAM each time the counter reaches 0. The refresh logic can accumulate up to a maximum of 15 refresh requests while it is waiting for the bus. Once the refresh logic owns the bus, all accumulated refreshes are performed to system memory, and no other accesses (CPU or IBUS) are allowed. Refreshes are staggered by one clock; that is, there will be at least one bus clock between transitions on any pair of SDRASB signals. Refresh clears the system-memory prefetch FIFO automatically.

3.4.16 Memory-to-CPU prefetch FIFO

After each burst 4-words read, the memory controller prefetches 4 additional words into its internal prefetch FIFO. If the processor subsequently attempts a read from an address immediately following (sequential to) the address of the last read cycle, the first 4 words will supplied from the prefetch FIFO.

The memory controller compares the current SysAD address with the previous address to determine the sequential nature of the access. Prefetched words are retained in the prefetch FIFO if accesses to resources other than system memory are performed between system memory accesses.

3.4.17 CPU-to-memory write FIFO

The memory controller has a 4-word CPU-to-memory write FIFO. This FIFO accepts writes at the maximum CPU speed. A single address is held for the buffered write, allowing the buffering of a single write transaction. That transaction may be a word, double word, 4-word data-cache write-back. When a word is placed in the FIFO by the CPU, the memory controller attempts to write the FIFO’s contents to memory as quickly as possible. If the next CPU read or write is addressed to memory, the controller negates ready signal, thus causing the next CPU transaction (read or write) to stall until the controller empties its FIFO. If the next CPU transaction (read or write) is addressed to a IBUS target, the memory controller asserts ready signal, thus the CPU transaction to complete.

3.4.18 SDRAM memory initialization

The following sections describe the configuration sequence used in this initialization.

3.4.1.1 Power-on initialization sequence by memory controller

The following sequence to configure memory is done automatically after reset:

1. Waits for 100 μs after power-on.
2. Performs all bank precharges.
3. Performs eight sequential auto refreshes (CBR).

3.4.1.2 Memory initialization sequence using software

The SDRAM must be initialized by the software using following sequence after power-on initialization.

1. Program the SDRAM type selection register “SDTSR”
2. Program the SDRAM mode register “SDMDR”.
3. Wait for 20 μs.
4. Program the DRAM refresh counter register.

At this point, memory is ready to use. All other configuration registers in the controller should then be programmed before commencing normal operation.

Remark The software should not change the SDTSR and SDMDR after SDRAM initialization sequence

3.5 IBUS Interface

3.5.1 Overview

- IBUS Master and target capability
- 64-word (256-byte) IBUS Slave TxFIFO (IBUS read data from IBUS)
- 64-word (256-byte) IBUS Slave RxFIFO (IBUS write data to IBUS)
- 4-word (16-byte) IBUS Master TxFIFO (VR4120A read data from IBUS)
- 4-word (16-byte) IBUS Master RxFIFO (VR4120A write data to IBUS)
- Supports bus timer to detect IBUS stall
- 66-MHz IBUS clock rate
- Support 266 MB/sec (32 bits @66 MHz) burst on IBUS
- Support endian conversion between SysAD BUS and IBUS master I/F
- Support endian conversion between memory BUS and IBUS slave I/F

3.5.2 Endian Conversion on IBUS master

“HSWP” bit on S_GMR is enabler for endian converter that is located on space between SysAD interface and IBUS master interface, so this works only IBUS target area. This converter is effective at the case of address swap mode only. This converter performs following data operations.

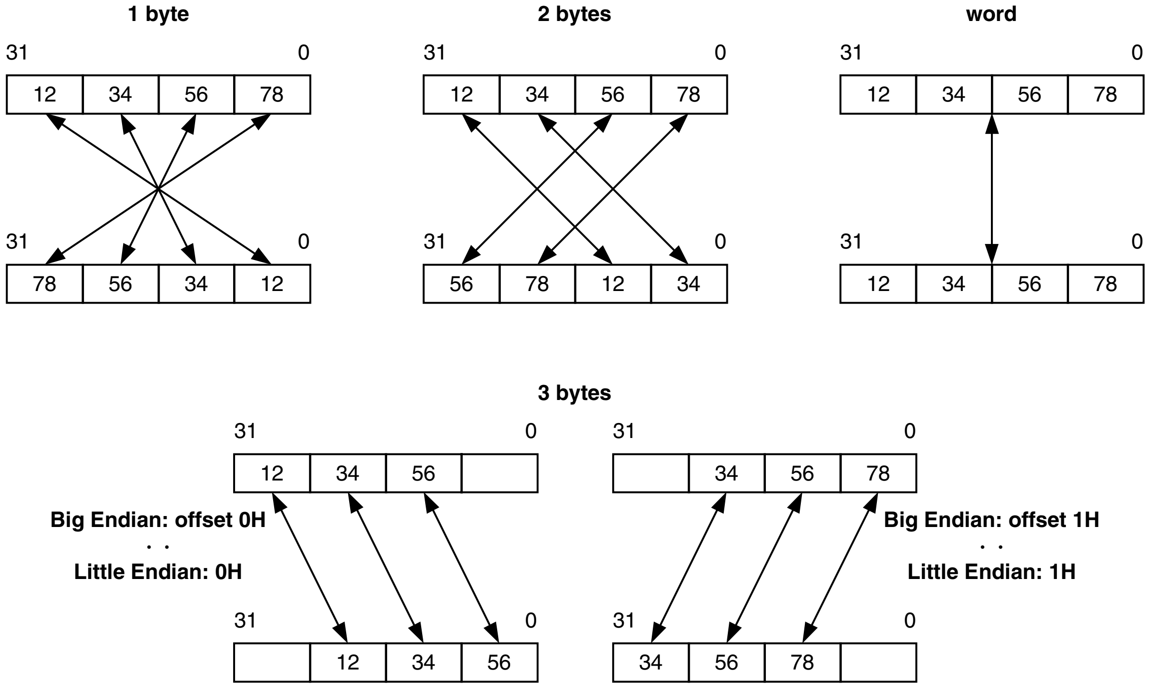
Table 3-11. Endian Translation Table for the data swap mode (IBUS master)

HSWP on GMR	Data Size	offset address [1:0] Note	Before Translation input data[31:0] in Data Phase	After Translation output data[31:0] in Data Phase	Remark
0	any	any	[31:0]	[31:0]	i.e. now
1	Over 1 word	0			
	1 byte	0, 1, 2, 3	[31:24] [23:16] [15:8] [7:0]	[7:0] [15:8] [23:16] [31:24]	-
	2 bytes	0, 2	[31:16] [15:0]	[15:0] [31:16]	-
	3 bytes	0	[31:8] [7:0]	[31:24] [23:0]	-
		1	[31:24] [23:0]	[31:8] [7:0]	-

Note This offset address[1:0] is expression on big endian mode.

In the following Figure, Upper side is 4 octet data of SysAD BUS. And Under side is 4 octet data of IBUS master I/F.

Outline figure of Endian converter



3.5.3 Endian Conversion on IBUS slave

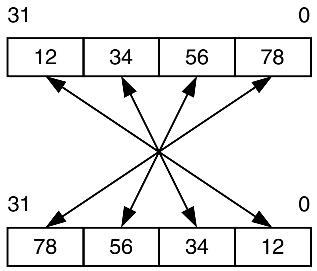
"MSWP" bit on S_GMR register is enabler for endian converter that is located on space between memory interface and IBUS slave interface, so this works only memory access via IBUS slave I/F. This converter is effective at the case of address swap mode only. This converter performs following data operations.

Table 3-12. Endian Translation Table for the data swap mode (IBUS slave)

MSWP on GMR	Before Translation input data[31:0] in Data Phase	After Translation output data[31:0] in Data Phase	Remark
0	[31:0]	[31:0]	i.e. now
1	[31:24] [23:16] [15:8] [7:0]	[7:0] [15:8] [23:16] [31:24]	-

In the following Figure, Upper side is 4 octet data of memory I/F. And Under side is 4 octet data of IBUS slave I/F.

Outline figure of Endian converter



3.5.4 ITCNTR (IBUS Timeout Timer Control Register)

The IBUS Timeout Timer control register “ITCNTR” is a read-write and word-aligned 32-bit register. ITCNTR is used to enable use of the IBUS Timeout Timer. ITCNTR is initialized to 0H at reset and contains the following field:

Bits	Field	R/W	Default	Description
31:1	Reserved	R/W	0	Hardwired to 0.
0	ITWEN	R/W	0	IBUS Timeout Timer enable 1 = Enable 0 = Disable

3.5.5 ITSETR (IBUS Timeout Timer Set Register)

This register sets the cycle for Deadman’s Switch functions. The Deadman’s Switch cycle can be set in 1-clock increments in a range from 1 to 232–1 clock. The DSUCLRR’s DSWCLR bit must be set by means of software within the specified cycle time. ITSETR is a 32-bit word-aligned register. Default is 8000_0000H.

Bits	Field	R/W	Default	Description
31:0	ITTIME	R/W	8000_ 0000H	IBUS Timeout value setting Timeout = ITTIME value system clock period (100 MHz:10 ns, 66 MHz:15 ns) example: ITTIME = 05F5_E100H (100 MHz) or 03F9_40AAH (66 MHz) -> Timeout = 1 sec ITTIME = 0BEB_C200H (100 MHz) or 07F2_8154H (66 MHz) -> Timeout = 2 sec ITTIME = 11E1_A300H (100 MHz) or 0BEB_C200H (66 MHz) -> Timeout = 3 sec : : :

3.6 DSU (Deadman's SW Unit)

3.6.1 Overview

The DSU detects when the VR4120A is in runaway (endless loop) state and resets the VR4120A. The use of the DSU to minimize runaway time effectively minimizes data loss that can occur due to software-related runaway states.

3.6.2 DSUCNTR (DSU Control Register)

This register is used to enable use of the Deadman's Switch functions.

DSUCNTR is a 32-bit word-aligned register. Default is 0H.

Bits	Field	R/W	Default	Description
31:1	Reserved	R/W	0	Hardwired to 0.
0	DSWEN	R/W	0	Deadman's Switch function enable 1 = enable 0 = disable

3.6.3 DSUSETR (DSU Time Set Register)

This register sets the Deadman's Switch cycle. The Deadman's Switch cycle can be set in 1-clock increments in a range from 1 to 2^{32} – 1 clock. The DSUCLRR's DSWCLR bit must be set by means of software within the specified cycle time. DSUSETR is a 32-bit word-aligned register. Default is 8000_0000H.

Bits	Field	R/W	Default	Description
31:0	DEDTIM	R/W	8000_ 0000H	Deadman's Switch cycle setting DSU cycle = DEDTIME value system clock period (100 MHz:10 ns, 66 MHz:15 ns) example: DEDTIM = 05F5_E100H (100 MHz) or 03F9_40AAH (66 MHz) -> DSU cycle = 1 sec DEDTIM = 0BEB_C200H (100 MHz) or 07F2_8154H (66 MHz) -> DSU cycle = 2 sec DEDTIM = 11E1_A300H (100 MHz) or 0BEB_C200H (66 MHz) -> DUS cycle = 3 sec : : :

3.6.4 DSUCLRR (DSU Clear Register)

Setting the DSWCLR bit in this register to '1' clears the Deadman's Switch counter. The VR4120A is reset automatically if a '1' is not written to the bit within the period specified in DSUSETR. DSUCLR is a 32-bit word-aligned register. Default is 0H.

Bits	Field	R/W	Default	Description
31:1	Reserved	W	0	Hardwired to 0.
0	DSWCLR	W	0	Deadman's Switch counter clear. Cleared to 0 when 1 is written. 1 = Clear 0 = Don't clear

3.6.5 DSUTIMR (DSU Elapsed Time Register)

This register indicates the elapsed time for the current Deadman's Switch timer.

DSUTIMR is a read-only and 32-bit word-aligned register. Default is 0H.

Bits	Field	R/W	Default	Description

31:0	CRTTIM	R	0	Current Deadman's Switch timer value (Elapsed time)
				
				example:
				CRTTIM = 05F5_E100H (100 MHz) or 03F9_40AAH (66 MHz) ->1 sec
				CRTTIM = 0BEB_C200H (100 MHz) or 07F2_8154H (66 MHz) -> 2 sec
				CRTTIM = 11E1_A300H (100 MHz) or 0BEB_C200H (66 MHz) -> 3 sec
				: : :

3.6.6 DSU register setting flow

The DSU register setting flow is described below.

1. Set the DSU's count-up value (from 1 to 231 – 1).
   The CPU will be reset if it does not clear (1 is not written to DSUCLRR) the timer within this period.
2. Enable the DSU
3. Clear the timer within the period specified in step 1 above.
   For normal use, repeat step 3. To obtain the current elapsed time, read DSUTIMR.
4. Disable the DSU for shutdown.

3.7 Endian Mode Software Issues

3.7.1 Overview

The native endian mode for MIPS processors, like Motorola and IBM 370 processors, is big endian. However, the native mode for Intel (which developed the PCI standard) and VAX processors is little endian. For PCI-compatibility reasons, most PCI peripheral chips operate natively in little-endian mode. While the μPD98502 is natively little-endian, it supports either big- or little-endian mode on the SysAD bus. The state of the ENDIAN signal at reset determines this endian mode. However, there are important considerations when using the controller in a mixed-endian design. The most important aspect of the endian issue is which byte lanes of the SysAD bus are activated for a particular address. If the big-endian mode is implemented for the CPU interface, the controller swaps bytes within words and half-words that are coming in and going out on the SysAD bus. All of the other interfaces of system controller operate in little-endian mode.

The sections below view the endian issue from a programmer's perspective. They describe how to implement mixed-endian designs and how to make code endian-independent.

Data in memory is always ordered in little-endian mode, even with a big-endian CPU.

Data in all internal registers and FIFOs is considered little endian regardless of CPU's endian mode.

Data addresses or Data byte order in the word are not swapped inside the device for accesses from a little-endian VR4120A to all local registers and memory when "BIG" signal is Low.

Data addresses are swapped inside the device for accesses from a big-endian VR4120A to all local registers and memory when "BIG" signal and "ENDCEN" signal are High.

Data byte order in the word are swapped inside the device for accesses from a big-endian CPU to all local registers and memory when "BIG" signal is High and "ENDCEN" signal is Low.

3.7.2 Endian modes

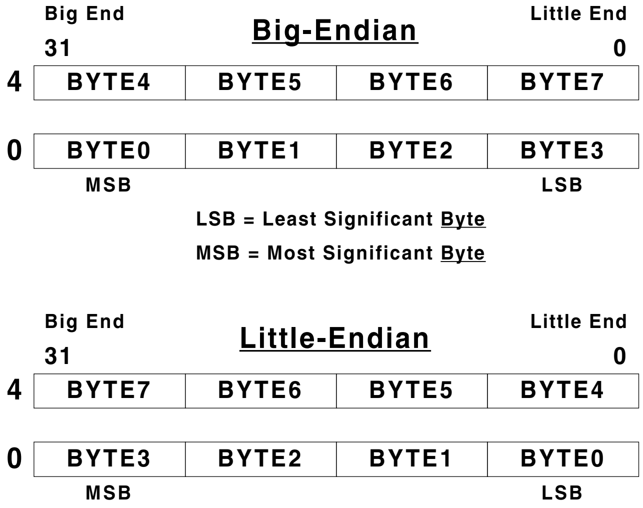
The endian mode of a device refers to its word-addressing method and byte order:

Big-endian devices address data items at the big end (most significant bit number). The most-significant byte (MSB) in an addressed data item is at the lowest address.

Little-endian devices address data items at the little end (least significant bit number). The most significant byte (MSB) in an addressed data item is at the highest address.

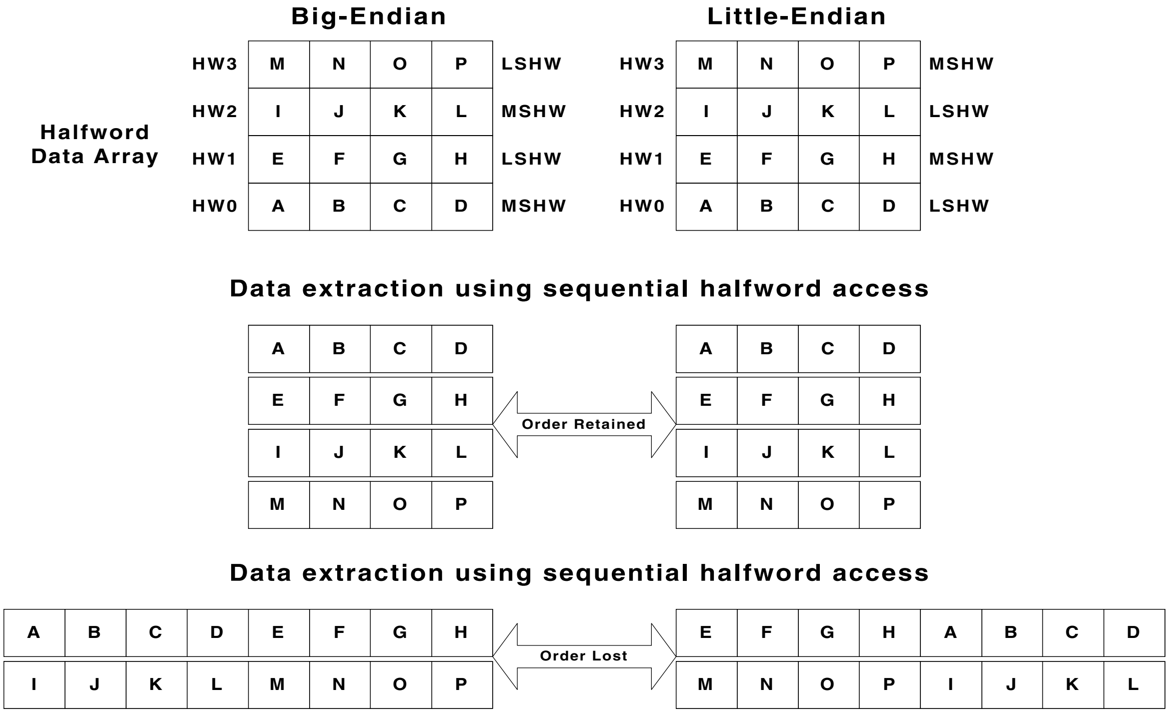
The following figures indicate the bit and byte order of the two endian modes, as it applies to bytes within word-sized data items. The bit order within bytes is the same for both modes. The big (most-significant) bit is on the left side, and the little (least significant) bit is on the right side. Only the bit order of sub-items is reversed within a larger addressable data item (half word, word, and double word) when crossing between the two endian modes. The sub-items' order of significance within the larger data item remains the same. For example, the least significant half word (LSHW) in a word is always to the right and the most significant half word (MSHW) is to the left.

Figure 3-1. Bit and Byte Order of Endian Modes


Big End
Big-Endian
Little End
31
0
4 BYTE4 BYTE5 BYTE6 BYTE7
0 BYTE0 BYTE1 BYTE2 BYTE3
MSB
LSB
LSB = Least Significant Byte
MSB = Most Significant Byte
Big End
Little-Endian
Little End
31
0
4 BYTE7 BYTE6 BYTE5 BYTE4
0 BYTE3 BYTE2 BYTE1 BYTE0
MSB
LSB

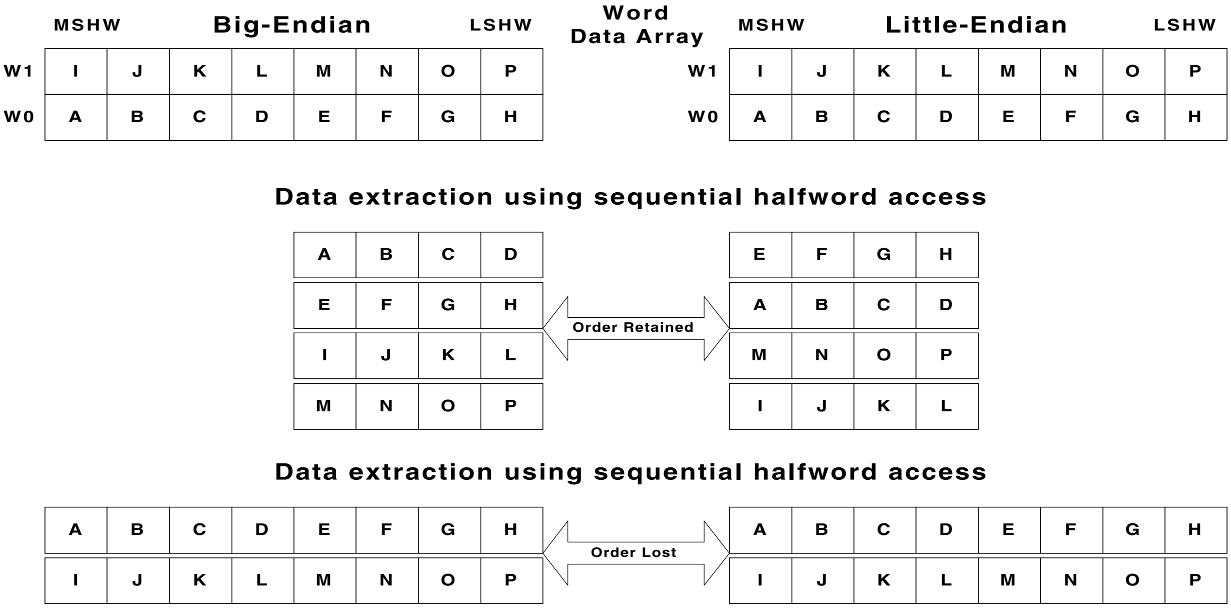
If the access type matches the data item type, no swapping of data sub-items is necessary. Thus, when making half-word accesses into a data array consisting of half-word data, no byte swapping takes place. In this case, data item bit order is retained between the two endian modes. The code that sequentially accesses the half-word data array would be identical, regardless of the endian mode of its VR4120A. The code would be endian-independent.

Figure 3-2. Half-word Data Array Example



However, when making half-word accesses into a data array consisting of word data, access to the more-significant half word requires the address corresponding to the less significant half word (and vice versa). Such code is not endian-independent. A super-group access (for example, accessing two half words simultaneously as a word from a half-word data array) causes the same problem. Such problems also arise when a half-word access is made into a 32-bit register, whereas a word access into a 32-bit register creates no problem.

Figure 3-3. Word Data Array Example



CHAPTER 4 ATM CELL PROCESSOR

4.1 Overview

This section describes functional specifications of ATM cell processor unit.

4.1.1 Function features

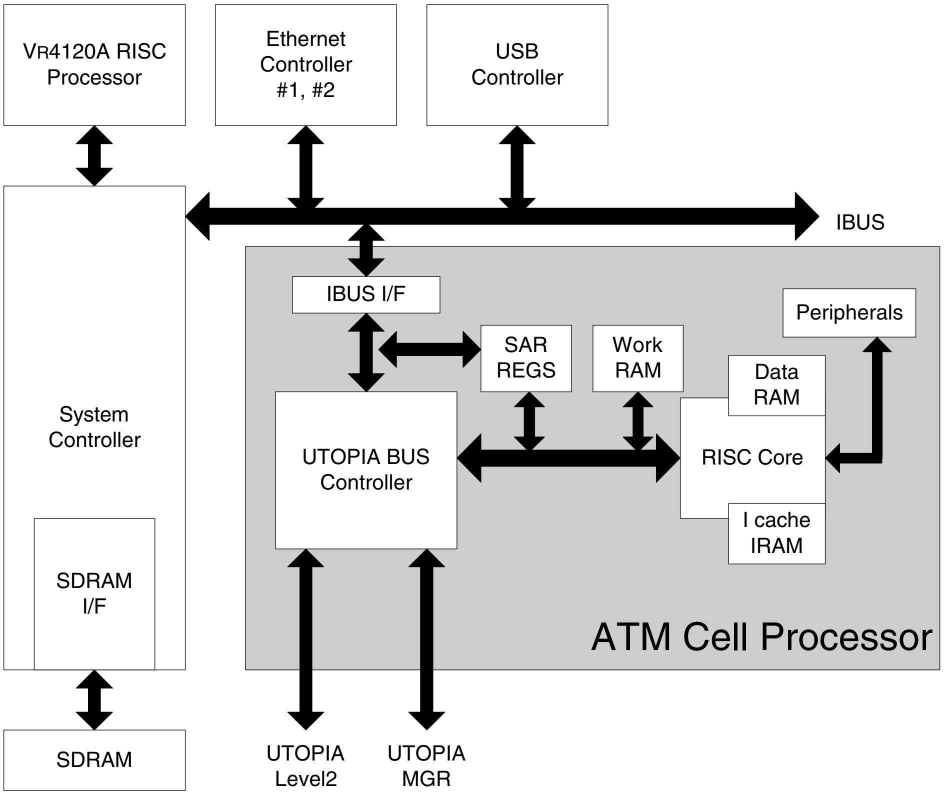
Features of ATM Cell Processor with out Firmware (F/W) is as follows:

- Data Transmission Capacity
  Aggregated transmission capacity is 50 Mbps, 25 Mbps for downstream and 25 Mbps for upstream.
- Supports ATM Adaptation Layers (AAL)
  AAL-0 (raw cells), AAL-2 and AAL-5 are supported.
- Supports Service Classes
  CBR, VBR and UBR.
- Number of VC's
  Maximum of 64 VC's will be supported.
- Scheduling
  Cell rate shaping will be performed in one-cell-time granularity on per VC basis
- Supports OAM function
- Supports switching function

ATM Cell Processor has a 32-bit micro-controller. All the above functions are realized by the Firmware assisted by H/W circuits.

4.1.2 Block diagram of ATM cell processor

Figure 4-1. Block Diagram of ATM Cell Processor



This block is an ATM cell processor. It consists of a 32-bit MCU, Peripherals (Interrupt Controller, Cell Timer, Scheduling Table and Rx Lookup Table), DMA controllers, a Work-RAM, and SAR-Registers.

4.1.2.1 RISC core

This block is RISC micro-controller. Its features are as follows:

- High performance 32-bit RISC micro-controller, 76 MIPS @ 66 MHz
- 32 x 32-bit General Purpose Registers
- 32-bit ALU, 32-bit Shifter, 16 x 16 Multiply-Adder
- 1-KB Data RAM, 8-KB Instruction RAM, 8-KB Instruction Cache

4.1.2.2 Peripherals

- Interrupt Controller (INTC) and Interrupt Edge Detector (INTEDGE)
- Peripherals for ATM functions – Scheduling Table, Rx Lookup Table, and Cell Timer

4.1.2.3 UTOPIA bus controller

This block has some H/W resources – DMA controller, FIFOs, CRC calculators/checkers. Its features are as follows:

- Scatter/Gather-DMA controller that can operate the distributed data according to descriptor tables, without F/W help. The DMA controller is used for each transmission and reception.
  Normal DMA mode is also supported.
  Furthermore, this DMA controller updates the information related to the DMA operations in the VC table in Work RAM.

- Internal BUS interface(IBUS)
- ATM Zero-padding

  Zero-padding is required in AAL-5 function. This block has the circuits for the padding. If the source address and the number of bytes to be padded are given, this block inserts zero padding as indicated.

- Transmission and reception SAR FIFO

  UTOPIA I/F Control block has a four-cell-depth FIFO for each transmission and reception. The last cell in Tx FIFO and the first cell in Rx FIFO are mapped to VR4120A RISC Processor/RISC Core memory space.
  UTOPIA 2 I/F is an 8-bit bus I/F to PHY devices, which is defined in a ATM Forum document, “ATM-PHY-0039”.
  UTOPIA MGR I/F will be supported as well.
  Its features are as follows:
  - It supports up to 15 PHY devices at a time. PHY addresses either from 0 through 14 or from 16 through 30 can be selected by setting command register.
  - The first word of cell header and the last two words of payload can be read in Big-Endian byte in order to insert/extract some special bit-fields.
  - To avoid Head-of-line Blocking, later cells can pass earlier cells if their destination PHY devices are not ready.
  - In Rx side, it filters out idle cells and unassigned cells when it detects their pre-determined header patterns.
  - It provides 3 different frequency clocks as Tx and Rx clocks. The frequency is defined by CLKUSL [1:0] signals. In the case that 33 MHz clock is used SCLK, the frequency is determined as follows.

33 MHz:	CLKUSL [1:0] = 00
16.5 MHz:	CLKUSL [1:0] = 01
25 MHz:	CLKUSL [1:0] = 10
No output:	CLKUSL [1:0] = 11 (Do not set)

- CRC-32/CRC-10 calculator & checker

  UTOPIA Bus Controller block has the CRC-32 calculator for each transmission and reception. CRC-32 value is calculated for every packet. For transmission, CRC-32 value is inserted into the CRC-32 field in trailer. For reception, CRC-32 value is compared with the value in the CRC-32 field in trailer, in order to check whether any errors occurred or not.
  UTOPIA Bus Controller block also has CRC-10 calculator for each transmission and reception. The user can select whether CRC-10 is included in the payload or not. CRC-10 value is calculated for every cell. For transmission, CRC-10 value is inserted into the last 10-bit area of the payload if selected. For reception, CRC-10 value is compared with the value in the last 10-bit of the payload if selected.

4.1.2.4 Other blocks

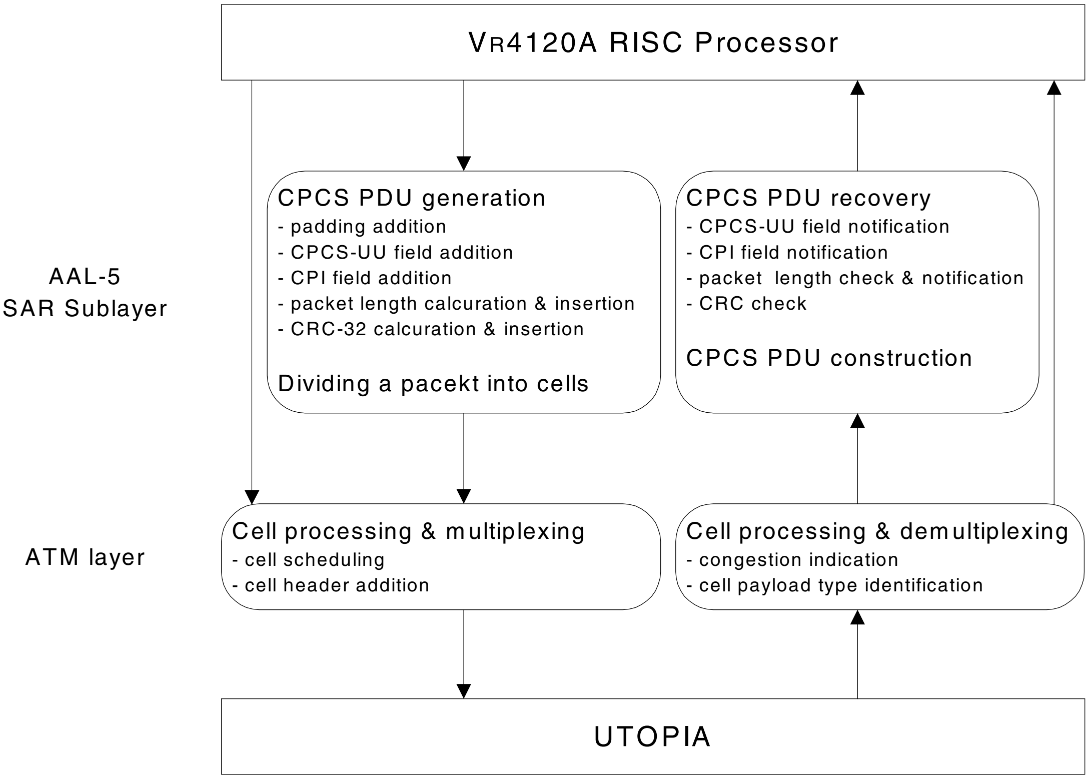
Work-RAM is 12 K-byte memory. Tables and Pool Descriptors are located in this RAM. It is shared between MCU and UTOPIA Bus Controller block. It also can be accessed by VR4120A RISC Processor, using Indirect-Access.

4.1.3 ATM cell processing operation overview

In this section, only overview is described. Please refer to section **4.7** for more detailed information.

ATM Cell Processor supports AAL-5 SAR sublayer and ATM layer functions. This block provides LLC encapsulation.

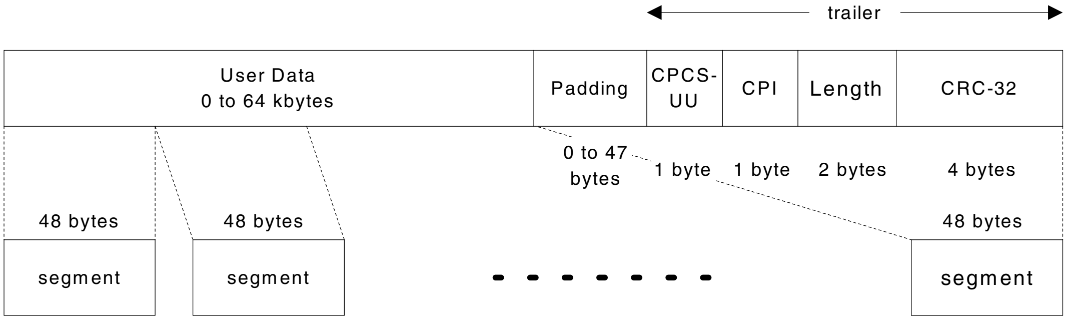
Figure 4-2. AAL-5 Sublayer and ATM Layer



4.1.3.1 AAL-5 SAR sublayer function

When ATM Cell Processor transmits a cell in AAL-5 mode, it adds a trailer to the variable-length data, as well as padding, so that its overall length becomes a multiple of 48 bytes, thereby generating an AAL-5 PDU. When ATM Cell Processor receives cells, it stores them in the SDRAM in order to assemble a CPCS PDU. ATM Cell Processor verifies the trailer of the assembled CPCS PDU. If the errors have occurred, the ATM Cell Processor informs the result to VR4120A RISC Processor with Rx indication.

Figure 4-3. AAL-5 Sublayer and ATM Layer



(a) Padding field: Field of 0 to 47 bytes that is inserted between the user data and the trailer to adjust the length of the resulting packet to a multiple of 48 bytes. ATM Cell Processor writes zeros to all its bits.

(b) CPCS-UU field: Used to transfer user information. The value set in the packet descriptor by the host is written in this field.

(c) CPI field: The use of this field has yet to be finalized. According to the current specifications, all its bits must be set to zeros. ATM Cell Processor, however, writes the value set in the packet descriptor into this field, as it does to the CPCS-UU field.

(d) Packet length (Length) field: Indicates the user data length in bytes, in binary notation.

(e) CRC-32 field: The CRC-32 value calculated for the range from the user data to the end of the Length field is set in this field. Generation polynomial is following

$$G(x) = 1 + x + x^2 + x^4 + x^5 + x^7 + x^8 + x^{10} + x^{11} + x^{12} + x^{16} + x^{22} + x^{23} + x^{26} + x^{32}$$

4.1.3.2 ATM layer function

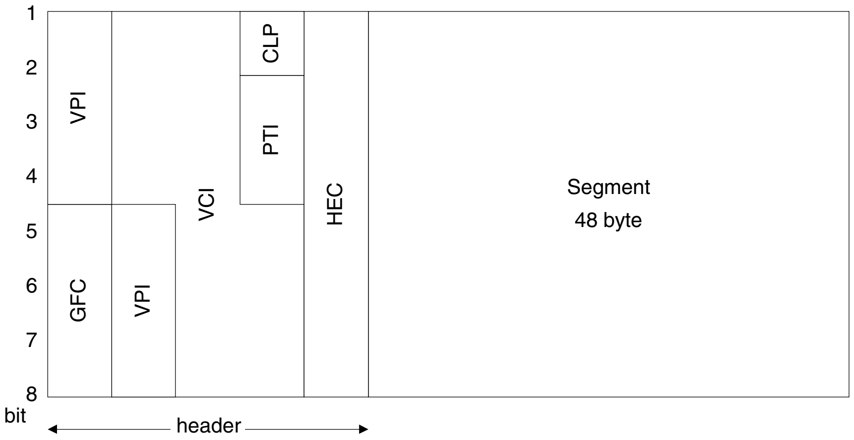
(1) Traffic classes

ATM Cell Processor supports 3 traffic classes; CBR (Constant Bit Rate), VBR (Variable Bit Rate) and UBR (Undefined Bit Rate).

(2) Generation a cell

ATM Cell Processor generates a cell by adding 5 bytes header to the segment as the figure shown below.

Figure 4-4. ATM Cell



The function of each field in the header is as follows:

(a) GFC (General Flow Control) field: Used for flow control. At transmission, the value set in the packet descriptor is written into this field. At reception, this field is ignored.

(b) VPI/VCI fields: VPI (Virtual Path Identifier), VCI (Virtual Channel Identifier) are routing fields which indicate the routing path. ATM Cell Processor supports a total of 64 VPI/VCI combinations for transmission and reception. For transmission, the 24-bit value set in the Tx VC table is written into these fields. For reception, only the VPIs/VCIs registered in the reception lookup table are allowed for reception.

(c) PTI (Payload Type Indication) field: 3-bit field indicating whether the cell payload is user data or management data. It also contains congestion information.

PTI	Usage
000	user data cell, no congestion, SDU type = 0
001	user data cell, no congestion, SDU type = 1
010	user data cell, congestion occurred, SDU type = 0
011	user data cell, congestion occurred, SDU type = 1
100	OAM F5 flow cell
101	OAM F5 flow cell
110	reserved for future use
111	reserved for future use

Here,

SDU type = 0: all segment except last cell of AAL-PDU

SDU type = 1: last cell of AAL-PDU. In this segment, trailer is included.
However, setting SDU type = 1 by user is prohibited.

OAM F5 flow cell: The cells for Operation, Administration and Maintenance.

(d) CLP (Cell Loss Priority) field: Indicates whether this cell is to be discarded preferentially if the network is congested. A CLP value of 1 indicates that the cell is to be discarded preferentially. ATM Cell Processor sets the appropriate value in this field according to the CLPM field of the packet descriptor.

(e) HEC (Header Error Control) field: Used for cell delineation, header error detection and correction. This field is processed in TC sublayer.

(3) Cell scheduling

ATM Cell Processor uses Scheduling Table, Cell Timer and Tx VC table for the cell scheduling. Before the VR4120A starts transmitting a packet, it sets the rate information in Tx VC table. ATM Cell Processor calculates cell transmission interval from the rate information, and put the next transmission time in Scheduling Table. When the Cell Timer and the next transmission time of certain VC becomes equal, a cell belongs to the VC is transmitted.

If the VC is CBR or UBR, only PCR (Peak Cell Rate) is used for scheduling, or if VC is VBR, PCR, SCR (Sustained Cell Rate) and MBS (Maximum Burst Size) are used.

(4) AAL-2 support/OAM support

ATM Cell Processor also supports AAL-2 and OAM F5 function. For the further information, please refer to the Application notes.

(5) Raw cell support

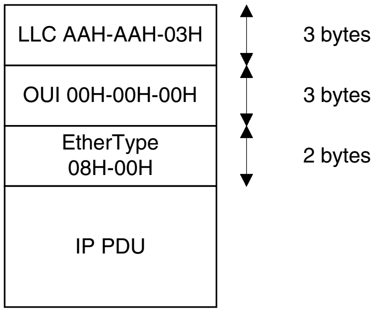
ATM Cell Processor also handles a cell as a raw cell, in order to support non AAL-5 traffic. For the VC which is set as the raw cell mode, ATM Cell Processor doesn't execute any AAL-5 dependent operation, such as calculating CRC-32 and adding trailers. In receiving mode, the ATM Cell Processor stores received cells with header and 11 bytes indication in SDRAM.

ATM Cell Processor has a CRC-10 insertion and verification function for non AAL-5 traffic. In the case that CRC-10 insertion is enabled, ATM Cell Processor calculates CRC-10 for each cell and inserts the result in the end of its payload at transmission side. ATM Cell Processor always verifies CRC-10 in receiving cells. If ATM Cell Processor detects an error, it sets error flag in indication and informs to the VR4120A.

4.1.3.3 LLC encapsulation

When LLC encapsulation mode is set in the VC table, ATM Cell Processor adds the LLC header to the top of the IP packet. In this case, ATM Cell Processor always encapsulates CPCS-PDU as IP PDU. However, if ATM mode Tx_Ready command is used, ATM Cell Processor does not execute encapsulation.

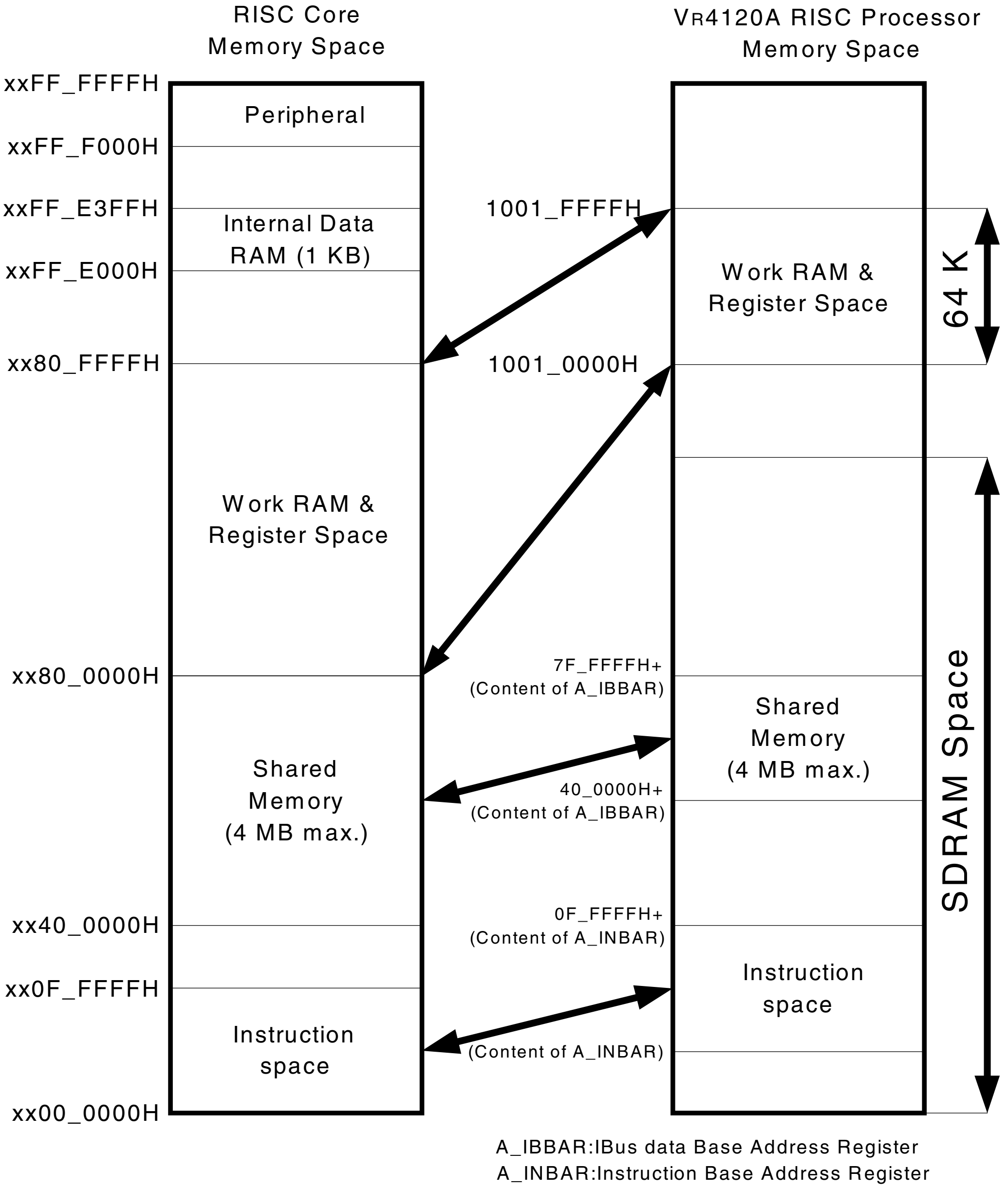
Figure 4-5. LLC Encapsulation



4.2 Memory Space

Although the RISC Core in the ATM Cell Processor is a 32-bit MPU, its physical memory space is 24-bit width.

Figure 4-6. Memory Space from VR4120A and RISC Core



A_IBBAR:IBus data Base Address Register
A_INBAR:Instruction Base Address Register

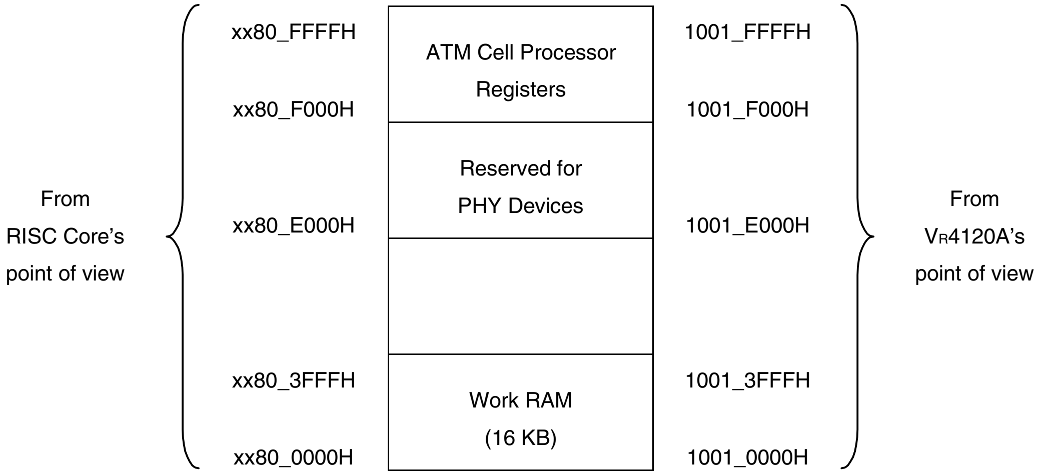
The configuration is shown as Figure 4-6. It contains instruction space, shared memory space, work RAM, internal memory space, and peripheral space.

VR4120A and RISC Core in the ATM Cell Processor share an external memory space. Shared memory will be implemented by using SDRAM devices. The address in VR4120A memory space will be determined by S/W and notified to RISC Core by setting A_IBBAR (IBUS data Base Address Register). Its capacity depends on the total capacity of physical memory, but not exceeds 4 MB.

4.2.1 Work RAM and register space

Work RAM and Register Space are shown in Figure 4-7. The capacity of Work RAM is 16 KB max. In order to access Work RAM, the user has to use “Indirect Access Command”. In register space, A_GMR (general mode register), A_GSR (general status register), A_CMR (command register), A_CER (command extension register) and other registers will be mapped. In PHY space, PHY devices can be accessed through UTOPIA management I/F.

Figure 4-7. Work RAM and Register Space



Internal memory space and peripheral space are exclusively used by RISC Core and cannot be seen by other blocks. Internal memory space will be used as stack and global variable space. In peripheral space, an interrupt controller and some other special blocks will be mapped. Scheduling table, VC lookup table and Cell Timer will be mapped in peripheral space as well.

4.2.2 Shared memory

ATM Cell Processor can access 4 MB or less of the memory space that is used as cell buffer and packet buffer. It is also used for instruction memory. This memory will be implemented off the chip. RISC Core in the ATM Cell Processor can access this memory through System Controller. From the RISC Core’s point of view the base address to access the memory should be written to A_IBBAR (IBUS data Base Address Register). A_IBBAR will be set during initializing period.

4.3 Interruption

When any bit in A_GSR (General Status Register) is NOT set to a ‘1’, that is, an interruption will be issued to VR4120A. The status of interruption is obtained by reading in A_GSR. When VR4120A reads A_GSR, the bits which are set and are NOT masked using A_IMR will be reset. The interruption can be masked by resetting bits of corresponding incidents in A_IMR (Interrupt Mask Register).

Interruptions from PHY devices are forwarded to VR4120A by PI bit of A_GSR automatically.

4.4 Registers for ATM Cell Processing

Registers in ATM Cell Processor block can be classified into 3 groups: SAR registers, DMA registers and FIFO Control registers. These registers can be accessed both VR4120A and RISC Core in ATM Cell Processor.

4.4.1 Register map

Registers are used for SAR functions. VR4120A writes to these registers to control SAR functions and reads from these registers to know the status. F/W on RISC Core reads these registers to know indication from VR4120A and writes to these registers to indicate the status of ATM Cell Processor.

4.4.1.1 Direct addressing register

From the VR4120A's point of view, 1001_0000H is the Base Address to access the registers in ATM Cell Processor.

Offset Address	Register Name	R/W	Access	Description
1001_F000H	A_GMR	R/W	W	General Mode Register
1001_F004H	A_GSR	RC	W	General Status Register
1001_F008H	A_IMR	R/W	W	Interrupt Mask Register
1001_F00CH	A_RQU	R	W	Receive Queue Underrun Register
1001_F010H	A_RQA	R	W	Receive Queue Alert Register
1001_F014H	N/A	-	-	Reserved for future use
1001_F018H	A_VER	R	W	Version Register
1001_F01CH	N/A	-	-	Reserved for future use
1001_F020H	A_CMR	R/W	W	Command Register
1001_F024H	N/A	-	-	Reserved for future use
1001_F028H	A_CER	R/W	W	Command Extension Register
1001_F02CH: 1001_F04CH	N/A	-	-	Reserved for future use
1001_F050H	A_MSA0	R/W	W	Mailbox0 Start Address Register
1001_F054H	A_MSA1	R/W	W	Mailbox1 Start Address Register
1001_F058H	A_MSA2	R/W	W	Mailbox2 Start Address Register
1001_F05CH	A_MSA3	R/W	W	Mailbox3 Start Address Register
1001_F060H	A_MBA0	R/W	W	Mailbox0 Bottom Address Register
1001_F064H	A_MBA1	R/W	W	Mailbox1 Bottom Address Register
1001_F068H	A_MBA2	R/W	W	Mailbox2 Bottom Address Register
1001_F06CH	A_MBA3	R/W	W	Mailbox3 Bottom Address Register
1001_F070H	A_MTA0	R/W	W	Mailbox0 Tail Address Register
1001_F074H	A_MTA1	R/W	W	Mailbox1 Tail Address Register
1001_F078H	A_MTA2	R/W	W	Mailbox2 Tail Address Register
1001_F07CH	A_MTA3	R/W	W	Mailbox3 Tail Address Register
1001_F080H	A_MWA0	R/W	W	Mailbox0 Write Address Register
1001_F084H	A_MWA1	R/W	W	Mailbox1 Write Address Register
1001_F088H	A_MWA2	R/W	W	Mailbox2 Write Address Register
1001_F08CH	A_MWA3	R/W	W	Mailbox3 Write Address Register
1001_F090H	A_RCC	R	W	Valid Received Cell Counter
1001_F094H	A_TCC	R	W	Valid Transmitted Cell Counter
1001_F098H	A_RUEC	R	W	Receive Unprovisioned VPI/VCI Error Cell Counter
1001_F09CH	A_RIDC	R	W	Receive Internal Dropped Cell Counter
1001_F0A0H: 1001_F0BCH	N/A	-	-	Reserved for future use
1001_F0C0H	A_T1R	R/W	W	T1 Time Register
1001_F0C4H	N/A	-	-	Reserved for future use

Offset Address	Register Name	R/W	Access	Description
1001_F0C8H	A_TSR	R/W	W	Time Stamp Register
1001_F0CCH: 1001_F1FCH	N/A	-	-	Reserved for future use
1001_F200H: 1001_F2FCH	N/A	-	-	Can not access from VR4120A RISC Core. This area is used for an internal function.
1001_F300H	A_IBBAR	R/W	W	IBUS Base Address Register
1001_F304H	A_INBAR	R/W	W	Instruction Base Address Register
1001_F308H: 1001_F31CH	N/A	-	-	Reserved for future use
1001_F320H	A_UMCMD	R/W	W	UTOPIA Management Interface Command Register
1001_F324H: 1001_F3FCH	N/A	-	-	Reserved for future use
1001_F400H: 1001_F4FCH	N/A	-	-	Can not access from VR4120A RISC Core. This area is used for an internal function.
1001_F500H: 1001_FFFCH	N/A	-	-	Reserved for future use

Remarks 1. In the "R/W" field,
"W" means "writeable",
"R" means "readable",
"RC" means "read-cleared",
"- " means "not accessible".

2. All internal registers are 32-bit word-aligned registers.

3. The burst access to the internal register is prohibited.
If such burst access has been occurred, IRERR bit in NSR is set and NMI will assert to CPU.

4. Read access to the reserved area will set the CBERR bit in the NSR register and the dummy read response data with the data-error bit set on SysCMD [0] is returned.

5. Write access to the reserved area will set the CBERR bit in the NSR register, and the write data is lost.

6. In the "Access" filed,
"W" means that word access is valid,
"H" means that half word access is valid,
"B" means that byte access is valid.

7. Write access to the read-only register cause no error, but the write data is lost.

8. The CPU can access all internal registers, but IBUS master device cannot access them.

4.4.1.2 Indirect addressing register

Address Note	Register Name	R/W	Access	Description
FFF410H	RXLCTR	R/W	W	Rx Lookup Table control Register
FFF600H	RxTBL000	W	H	Rx Lookup Table Entry00 Halfword0
FFF602H	RxTBL001	W	H	Rx Lookup Table Entry00 Halfword1
FFF6FCH	RxTBL3F0	W	H	Rx Lookup Table Entry3F Halfword0
FFF6FEH	RxTBL3F1	W	H	Rx Lookup Table Entry3F Halfword1
FFF700H	RxTBC00	R/W	W	Rx Lookup Table Control Entry00
FFF77EH	RXTBC3F	R/W	W	Rx Lookup Table Control Entry3F

Note These addresses are used in Indirect Address Command.

4.4.2 A_GMR (General Mode Register)

A_GMR is used to select operation mode of this block, enables/disables ATM SAR operations. After reset, VR4120A must write this register for initialization. Modification of A_GMR after starting Tx/Rx operations is prohibited. All bits of this register are writeable, but the bits 31-15, 13-2 are reserved for future use. Initial value is all zero.

Bits	Field	R/W	Default	Description
31:15	Reserved	R/W	0	Reserved for future use. Write '0's.
14	LP	R/W	0	0 = Loopback is not performed at the UTOPIA interface 1 = Loopback is performed at the UTOPIA interface
13:2	Reserved	R/W	0	Reserved for future use. Write '0's.
1	TE	R/W	0	0 = Transmit disable 1 = Transmit enable
0	RE	R/W	0	0 = Receive disable 1 = Receive enable

4.4.3 A_GSR (General Status Register)

A_GSR shows interruption status. When an event that triggers interruption occurs, F/W on RISC Core set a bit in A_GSR corresponds to the type of event. If the corresponding bit in A_IMR (Interrupt Mask Register) is set to a '0' and the interruption is not masked, an interruption is issued to VR4120A. The bit in A_GSR is able to be read cleared. When the same type of events occurs before the bit has been read, the bit will be set again.

Initial value is all zero.

Bits	Field	R/W	Default	Description
31	PI	RC	0	0 = PHY layer device interruption has not occurred 1 = PHY layer device interruption has occurred
30	RQA	RC	0	0 = receive Queue alert has not occurred 1 = receive Queue alert has occurred
29	RQU	RC	0	0 = receive Queue underflow has not occurred 1 = receive Queue underflow has occurred
28:24	Reserved	R	0	Reserved for future use
23	SQO	RC	0	0 = scheduling Queue overflow has not occurred 1 = scheduling Queue overflow has occurred
22	Reserved	R	0	Reserved for future use
21	FER	RC	0	0 = Fatal Error has not occurred 1 = Fatal Error has occurred
20:17	Reserved	R	0	Reserved for future use
16	BER	RC	0	0 = Internal Bus Error has not occurred 1 = Internal Bus Error has occurred
15:8	RCR[7:0]	RC	0	0 = raw cell is not in Pool No. [7:0] 1 = raw cell is in Pool No. [7:0]
7:4	MF[3:0]	RC	0	0 = Mailbox No. [3:0] is not full 1 = Mailbox No. [3:0] is full
3:0	MM[3:0]	RC	0	0 = mailbox No. [3:0] is not marked 1 = Mailbox No. [3:0] is marked

4.4.4 A_IMR (Interrupt Mask Register)

A_IMR masks interruption for each corresponding event. A Mask bit, which locates in the same bit location to a corresponding bit in A_GSR, masks interruption. If a bit of this register is reset to a ‘0’, the corresponding bit of the A_GSR is masked. If it is set to a ‘1’, the corresponding bit is unmasked. When the mask bit is reset and the bit in A_GSR is set, an interruption is issued to VR4120A.

All bits of this register is writeable, but the bits 28-24, 22, 20-16 are reserved for future use.

Initial value is all zero.

Bits	Field	R/W	Default	Description
31	PI	R/W	0	Mask bit for PHY layer device interruption 0 = mask 1 = unmask
30	RQA	R/W	0	Mask bit for Receive Queue Alert 0 = mask 1 = unmask
29	RQU	R/W	0	Mask bit for Receive Queue underflow 0 = mask 1 = unmask
28:24	Reserved	R/W	0	Reserved for future use. Write ‘0’s.
23	SQO	R/W	0	Mask bit for Scheduling Queue overflow 0 = mask 1 = unmask
22	Reserved	R/W	0	Reserved for future use. Write ‘0’s.
21	FER	R/W	0	Mask bit for Fatal Error 0 = mask 1 = unmask
20:16	Reserved	R/W	0	Reserved for future use. Write ‘0’s.
15:8	RCR[7:0]	R/W	0	Mask bit for Raw cell reception 1 = Raw cell is in Pool No. [7:0] 0 = Raw cell is not in Pool No. [7:0]
7:4	MF[3:0]	R/W	0	Mask bit for Mailbox full 1 = Mailbox No. [3:0] is full 0 = Mailbox No. [3:0] is not full
3:0	MM[3:0]	R/W	0	Mask bit for Mailbox mark 1 = Mailbox No. [3:0] is marked 0 = Mailbox No. [3:0] is not marked

4.4.5 A_RQU (Receiving Queue Underrun Register)

A_RQU shows the status of each pool. When a pool has no free buffers, the corresponding bit is set. ATM Cell Processor detects a pool empty when it receives a cell and try to send the cell to buffer. Whenever one of A_RQU bits is set, A_RQU bit in A_GSR will be set. In this block, only pool7 to pool0 will be used. If a bit is set to '1', corresponding pool has no free buffers. Initial value is all zero.

Bits	Field	R/W	Default	Description
31:8	Reserved	R	0	Reserved for future use
7:0	A_RQU[7:0]	R	0	0 = pool [7:0] has free buffers 1 = pool [7:0] has no free buffers

4.4.6 A_RQA (Receiving Queue Alert Register)

A_RQA shows pools with less remaining batches than "ALERT LEVEL", which is set by VR4120A. Whenever one of A_RQA bits is set, A_RQA bit in A_GSR will be set. In this block, only pool7 to pool0 will be used. If a bit is set to '1', the number of remaining batches is less than "ALERT LEVEL". Initial value is all zero.

Bits	Field	R/W	Default	Description
31:8	Reserved	R	0	Reserved for future use
7:0	A_RQA[7:0]	R	0	0 = pool [7:0] has more or equal remaining batches than "ALERT LEVEL" 1 = pool [7:0] has less remaining batches than "ALERT LEVEL"

4.4.7 A_VER (Version Register)

A_VER shows version number of ATM Cell Processor block. Initial value is 0000_0200H.

Bits	Field	R/W	Default	Description
31:16	Reserved	R	0	Reserved for future use
15:8	MAJOR	R	02H	Major revision
7:0	MINOR	R	00H	Minor revision

4.4.8 A_CMR (Command Register)

ATM Cell Processor receives command and parameter when VR4120A writes them in A_CMR and A_CER. ATM Cell Processor can handle only one command at a time. When ATM Cell Processor receives a command from VR4120A, it sets Busy Flag in the register automatically to indicate it is busy. While Busy Flag is set, ATM Cell Processor can not receive a new command. If VR4120A writes a new command when Busy Flag is set, the new command will be ignored. Initial value is zero. Detail of this register is described in Section **4.7 Commands**.

Bits	Field	R/W	Default	Description
31	BSY	R/W	0	Busy Flag
30:0	A_CMR	R/W	0	Command and parameter

4.4.9 A_CER (Command Extension Register)

Command Extension Register. Initial value is zero. Detail of this register is described in Section **4.7 Commands**.

Bits	Field	R/W	Default	Description
31:0	A_CER	R/W	0	Parameter of command

4.4.10 A_MSA0 to A_MSA3 (Mailbox Start Address Register)

A_MSA0 to A_MSA3 shows start address of Receive Mailbox (Mailbox0 and Mailbox1) and Transmit Mailbox (Mailbox2 and Mailbox3) respectively. Initial value is all zero.

Bits	Field	R/W	Default	Description
31:0	A_MSA0	R/W	0	Start address of Mailbox0

Bits	Field	R/W	Default	Description
31:0	A_MSA1	R/W	0	Start address of Mailbox1

Bits	Field	R/W	Default	Description
31:0	A_MSA2	R/W	0	Start address of Mailbox2

Bits	Field	R/W	Default	Description
31:0	A_MSA3	R/W	0	Start address of Mailbox3

4.4.11 A_MBA0 to A_MBA3 (Mailbox Bottom Address Register)

A_MBA0 to A_MBA3 shows bottom address of Receive Mailbox (Mailbox0 and Mailbox1) and Transmit mailbox (Mailbox2 and Mailbox3) respectively. Initial value is all zero.

Bits	Field	R/W	Default	Description
31:0	A_MBA0	R/W	0	Bottom address of Mailbox0

Bits	Field	R/W	Default	Description
31:0	A_MBA1	R/W	0	Bottom address of Mailbox1

Bits	Field	R/W	Default	Description
31:0	A_MBA2	R/W	0	Bottom address of Mailbox2

Bits	Field	R/W	Default	Description
31:0	A_MBA3	R/W	0	Bottom address of Mailbox3

4.4.12 A_MTA0 to A_MTA3 (Mailbox Tail Address Register)

A_MTA0 to A_MTA3 shows tail address of Receive Mailbox (Mailbox0 and Mailbox1) and Transmit Mailbox (Mailbox2 and Mailbox3) respectively. Initial value is zero.

Bits	Field	R/W	Default	Description
31:0	A_MTA0	R/W	0	Tail address of Mailbox0

Bits	Field	R/W	Default	Description
31:0	A_MTA1	R/W	0	Tail address of Mailbox1

Bits	Field	R/W	Default	Description
31:0	A_MTA2	R/W	0	Tail address of Mailbox2

Bits	Field	R/W	Default	Description
31:0	A_MTA3	R/W	0	Tail address of Mailbox3

4.4.13 A_MWA0 to A_MWA3 (Mailbox Write Address Register)

A_MWA0 to A_MWA3 shows write address of Receive Mailbox (Mailbox0 and Mailbox1) and Transmit Mailbox (Mailbox2 and Mailbox3) respectively. Initial value is zero.

Bits	Field	R/W	Default	Description
31:0	A_MWA0	R/W	0	Write address of Mailbox0

Bits	Field	R/W	Default	Description
31:0	A_MWA1	R/W	0	Write address of Mailbox1

Bits	Field	R/W	Default	Description
31:0	A_MWA2	R/W	0	Write address of Mailbox2

Bits	Field	R/W	Default	Description
31:0	A_MWA3	R/W	0	Write address of Mailbox3

4.4.14 A_RCC (Valid Received Cell Counter)

A_RCC counts the number of valid received cells. It is a 32-bit counter. Overflow of this counter does NOT cause any interruption. Initial value is zero.

Bits	Field	R/W	Default	Description
31:0	A_RCC	R	0	Number of valid received cells

4.4.15 A_TCC (Valid Transmitted Cell Counter)

A_TCC counts the number of valid transmitted cells. It is a 32-bit counter. Overflow of this counter does NOT cause any interruption. Initial value is zero.

Bits	Field	R/W	Default	Description
31:0	A_TCC	R	0	Number of valid transmitted cells

4.4.16 A_RUEC (Receive Unprovisioned VPI/VCI Error Cell Counter)

A_RUEC counts the number of received cells with VPI/VCI Error. It is a 32-bit counter. Overflow of this counter does NOT cause any interruption. Initial value is zero.

Bits	Field	R/W	Default	Description
31:0	A_RUEC	R	0	Number of received cells with VPI/VCI Error

4.4.17 A_RIDC (Receive Internal Dropped Cell Counter)

A_RIDC counts the number of received cells which are dropped inside ATM Cell Processor. It is a 32-bit counter. Overflow of this counter does NOT cause any interruption. Initial value is zero.

Bits	Field	R/W	Default	Description
31:0	A_RIDC	R	0	Number of dropped cells

4.4.18 A_T1R (T1 Time Register)

A_T1R shows time which user allows ATM Cell Processor to spend to receive a whole of one packet. Initial value is “0000_FFFFH”.

Bits	Field	R/W	Default	Description
31	Reserved	R/W	0	Reserved for future use. Write ‘0’s.
30:0	A_T1R	R/W	FFFFH	Allowable time to receive a whole of one packet

4.4.19 A_TSR (Time Stamp Register)

A_TSR shows a value of the 32-bit counter that ATM Cell Processor counts its system clock. It is used as time stamps of receive start time of T1 timer function. Initial value is “0000_0000H”, and count up starts right after reset.

Bits	Field	R/W	Default	Description
31:0	A_TSR	R/W	0	System clock count of ATM Cell Processor

4.4.20 A_IBBAR (IBUS Base Address Register)

A_IBBAR contains the base address for the access thorough IBUS to outside. RISC Core-space is addressed using 24-bit address, while VR4120A RISC Processor space is addressed using 32-bit address. Therefore, the extension of address is necessary when the access from the inside of this block to the outside is requested. Initial value is zero.

Bits	Field	R/W	Default	Description
31:0	A_IBBAR	R/W	0	Base address for the access thorough IBUS to outside

4.4.21 A_INBAR (Instruction Base Address Register)

A_INBAR contains the base address to fetch instructions. RISC Core-space is addressed using 24-bit address, while VR4120A RISC Processor space is addressed using 32-bit address. Therefore, the extension of address is necessary when the access from the inside to the outside is requested. Initial value is zero.

Bits	Field	R/W	Default	Description
31:0	A_INBAR	R/W	0	Base address to fetch instructions

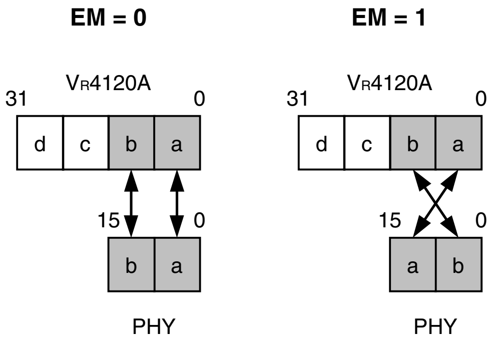
4.4.22 A_UMCMD (UTOPIA Management Interface Command Register)

A_UMCMD selects operation mode of UTOPIA Management Interface. After reset, RISC Core must write this register to configure UTOPIA Management Interface.

When BM bit is set to ‘0’, it means 8-bit mode and UMD [7:0] pins are valid.

When BM bit is set to ‘1’, it means 16-bit mode. In this case, only half-word-aligned access is accepted.

EM bit is set to ‘1’ only in 16-bit transfer mode. EM bit can change the data alignment as shown below.



Initial value of A_UMCMD is zero.

Bits	Field	R/W	Default	Description
31	Reserved	R/W	0	Reserved for future use. Write ‘0’s.
30	BM	R/W	0	0 = 8-bit transfer mode 1 = 16-bit transfer mode
29	EM	R/W	0	0 = data let straight 1 = data let cross
28:3	Reserved	R/W	0	Reserved for future use. Write ‘0’s.
2	PR	R/W	0	0 = to deassert UMRSTB 1 = to assert UMRSTB to reset PHY device
1:0	MSL	R/W	00	00 = UTOPIA Management acts in the Motorola-compatible mode (DS, R/W, DTACK style) 01 = UTOPIA Management acts in the Intel-compatible mode (RD, WR, RDY style) 1x = reserved

4.5 Data Structure

ATM Cell Processor has Tx/Rx buffer structure similar to that of Ethernet Controller and USB Controller.

4.5.1 Tx buffer structure

The following figure shows Tx buffer structure used by ATM Cell Processor. It consists of a packet descriptor, some buffer directories, and data buffers. A Rx buffer structure and a Tx buffer structure are similar, so that reconstructing buffer structure is not needed when sending out a received packet.

Figure 4-8. Tx Packet

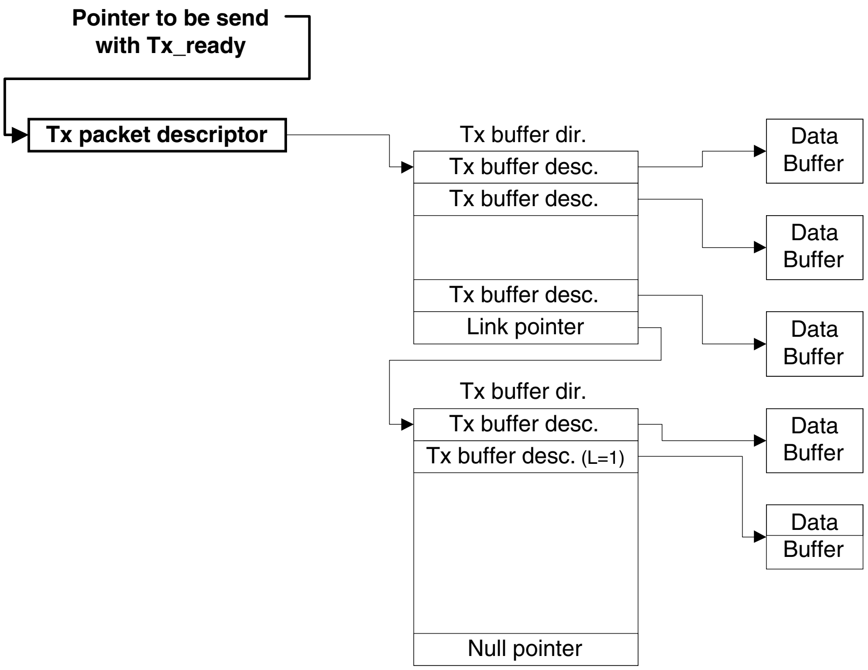


Figure 4-9. Tx Buffer Elements

- Tx packet descriptor

<table>
<tr><td>31</td><td>16</td><td>15</td><td>0</td></tr>
<tr><td colspan="2">Attribute</td><td>CPCS-UU</td><td>CPI</td></tr>
<tr><td colspan="4">Tx buffer directory Address</td></tr>
</table>

- Tx buffer directory

<table>
<tr><td>31</td><td>0</td></tr>
<tr><td colspan="2">Tx buffer desciptor 0</td></tr>
<tr><td colspan="2">Tx buffer desciptor 1</td></tr>
<tr><td colspan="2">Tx buffer desciptor 2</td></tr>
<tr><td colspan="2">Tx buffer desciptor 3</td></tr>
<tr><td colspan="2">Tx buffer desciptor 4</td></tr>
<tr><td colspan="2">|</td></tr>
<tr><td colspan="2">Tx buffer desciptor N</td></tr>
<tr><td colspan="2">Tx link pointer</td></tr>
</table>

- Tx buffer descriptor

<table>
<tr><td>31</td><td></td><td>16</td><td>15</td><td>0</td></tr>
<tr><td>L</td><td>DL</td><td>0</td><td colspan="2">Size</td></tr>
<tr><td colspan="5">Buffer Address</td></tr>
</table>

- Tx link pointer

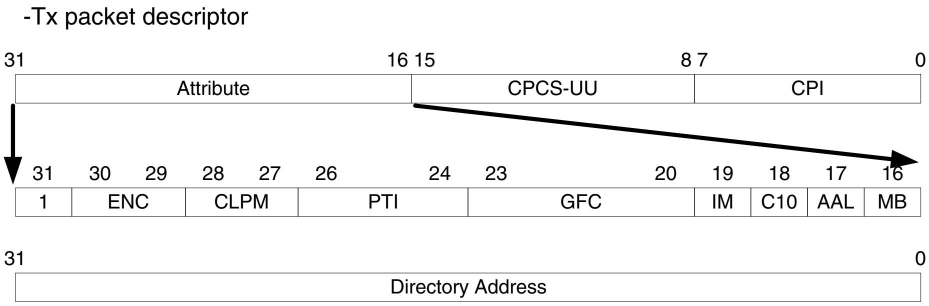
<table>
<tr><td>31</td><td></td><td>0</td></tr>
<tr><td>0</td><td>DL</td><td>0</td></tr>
<tr><td colspan="3">Tx buffer directory Address</td></tr>
</table>

Figure 4-9 shows Tx buffer elements. Each element consists of a couple of 32-bit words in sequential address. Detail is given in following sections.

4.5.1.1 Packet descriptor

A packet descriptor contains two words shown as Figure 4-10. Its address is word aligned.

Figure 4-10. Tx Packet Descriptor



-Tx packet descriptor

31 16	15 8	7 0
Attribute	CPCS-UU	CPI

31	30 29	28 27	26 24	23 20	19	18	17	16
1	ENC	CLPM	PTI	GFC	IM	C10	AAL	MB

31 0
Directory Address

Table 4-1 is a list of Tx packet attributes. Detail will be given in Operation chapter.

Table 4-1. List of Tx Packet Attribute

Field	Description
ENC	It contains Encapsulation bits to specify Encapsulation modes.
CLPM	It contains CLP bit to be set in Tx cell header.
PTI	It contains PTI bit to be set in Tx cell header.
GFC	It contains GFC bit to be set in Tx cell header.
IM	It disables interruption and indication when Tx is completed.
C10	It enables generation and insertion of CRC10 code.
AAL	It specifies type of AAL.
MB	It indicates the number of mailbox.
CPCS-UU	It contains CPCS-UU bits to be set in Tx cell trailer.
CPI	It contains CPI bits to be set in Tx cell trailer.

4.5.1.2 Tx buffer directory

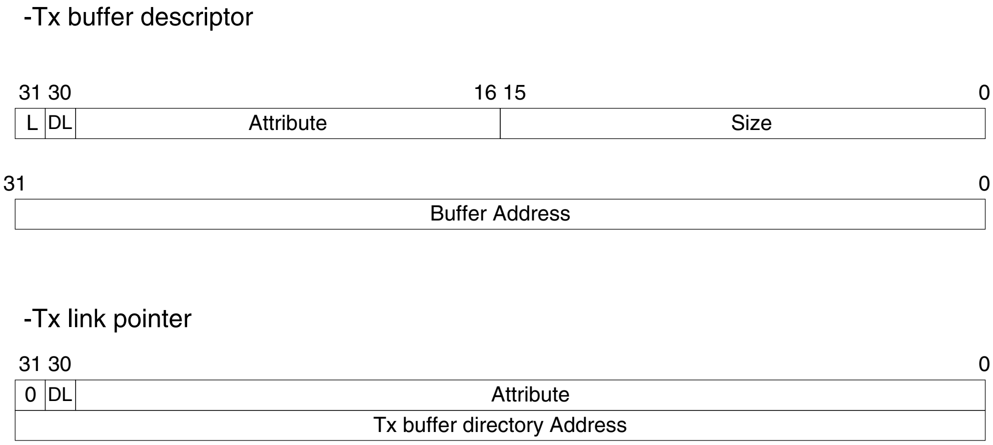
Tx buffer directory contains some buffer descriptors, up to 255, and a link pointer. Its address is word aligned. The end of buffer directory must be a link pointer. Buffer descriptors must be read and served from the top in a sequential manner.

4.5.1.3 Tx buffer descriptor

Both a Tx buffer descriptor and a Tx link pointer consist of 2 words. DL bit, bit 30 of the first word, indicates that these two words are a buffer descriptor (DL = 1) or a link pointer (DL = 0). In the Tx buffer descriptor, L bit, bit 31, indicates that the buffer pointed by this descriptor contains the last portion of a packet.

A Tx link pointer is shown as Figure 4-11. L bit, bit 31, is fixed to zero. If there is no buffer directory to be linked, directory address of link pointer must be zero, as a null pointer.

Figure 4-11. Tx Buffer Descriptor/Link Pointer



4.5.1.4 Data buffer

Data buffer contains actual packet data to be sent. Size of a buffer can vary from 1 byte to 64 Kbytes. Its address is byte aligned.

4.5.2 Rx pool structure

Rx buffer structure is defined as a pool. Eight pools are supported. A pool is composed of chain of Rx buffer directories. Each Rx buffer directory has some buffer descriptors and a link pointer. Each buffer descriptor points a buffer string of received cell data. A link pointer has an address to a next Rx buffer directory.

VR4120A will create up to 8 pools and give them to ATM Cell Processor.

Figure 4-12. Rx Pool Structure

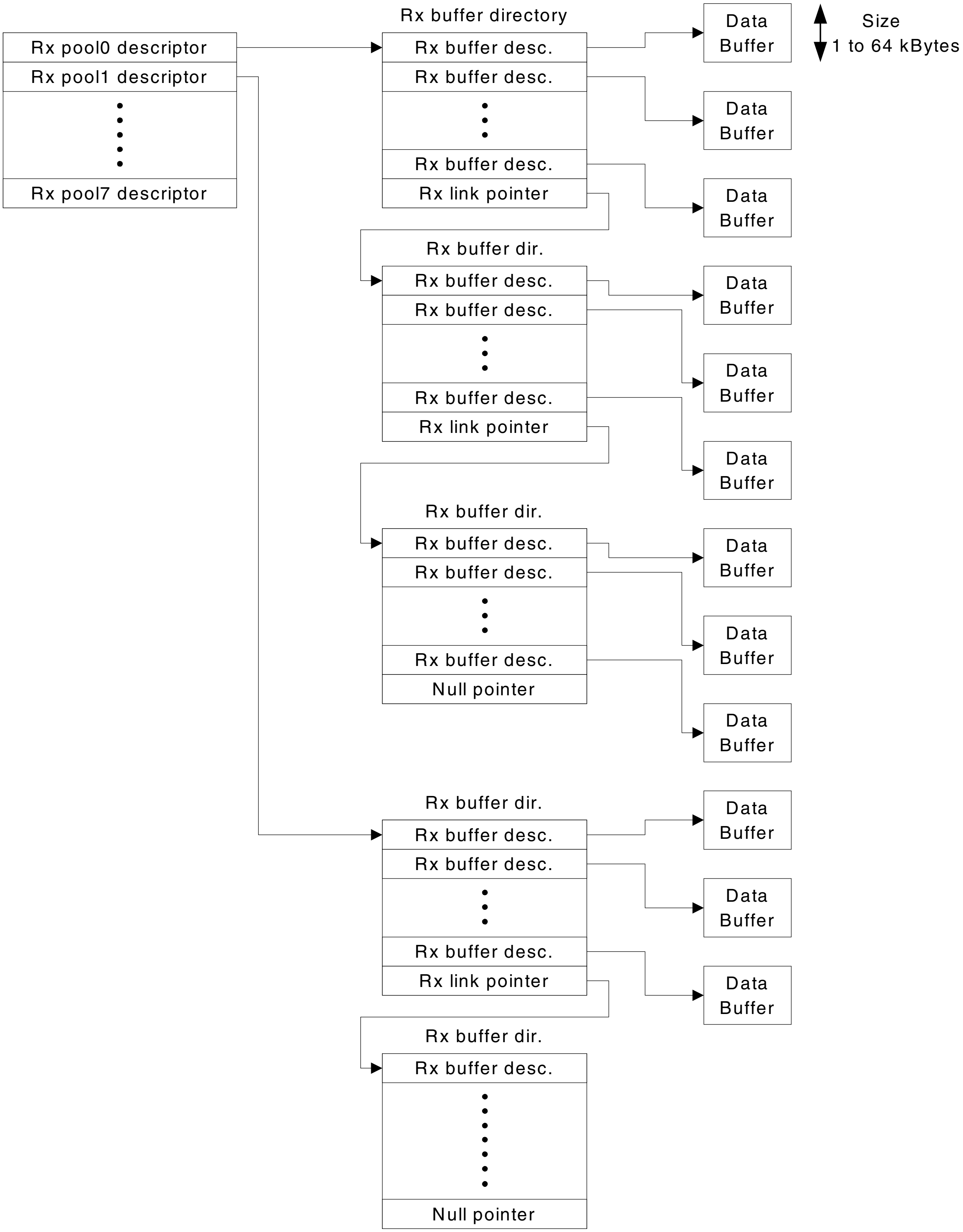


Figure 4-13. Rx Pool Descriptor/Rx Buffer Directory/Rx Buffer Descriptor/Rx Link Pointer

-Rx pool descriptor

31 16	15 0
Attribute	
Rx buffer directory Address	

-Rx buffer directory

31 0
Rx buffer desciptor 0
Rx buffer desciptor 1
Rx buffer desciptor 2
Rx buffer desciptor 3
Rx buffer desciptor 4
Rx buffer desciptor 5
|
Rx link pointer

-Rx buffer descriptor

31 16	15 0
Attribute	Size
Buffer Address	

-Rx link pointer

31 0
Attribute
Rx buffer directory Address

Figure 4-13 shows Rx buffer elements. Each element consists of a couple of 32-bit words of sequential address. Detail is given in following sections.

4.5.2.1 Rx pool descriptor

A pool descriptor contains two words shown as Figure 4-14. Its address is word aligned.

Figure 4-14. Rx Pool Descriptor

-Rx pool descriptor

31	30 28	27 24	23 16	15 0
0	Alert level	all 0	dir. size in a pool	Remaining # of dir. in the pool

31 0
Rx buffer directory Address

Table 4-2. List of Rx Pool Attributes

Field	Note
Alert level	When the remaining number of buffer directories is lower than this number times 4, an alert interrupt will be issued.
Dir. size in a pool	It contains the number of Rx buffer descriptors in a buffer directories in the pool.
Remaining number of dir. in the pool	It contains the current remaining number of buffer directories.
Directory Address	It contains address of the first buffer directory in the pool.

4.5.2.2 Rx buffer directory

Rx buffer directory contains some buffer descriptors, up to 255, and a link pointer. Number of buffer descriptors in each directory in one pool is identical. Number can vary in different pools.

Address of buffer directory is word aligned. The end of buffer directory must be a link pointer. Buffer descriptor must be read and used from the top in a sequential manner.

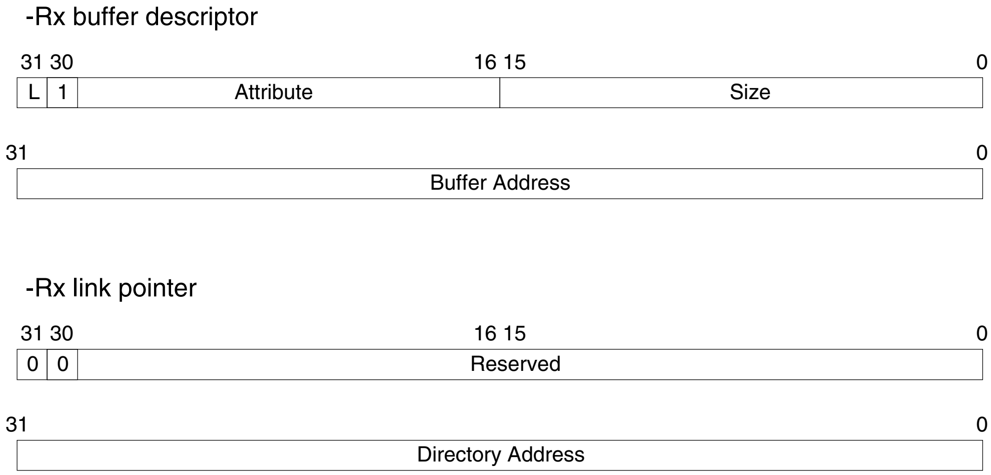
4.5.2.3 Rx buffer descriptor

Both an Rx buffer descriptor and an Rx link pointer consist of 2 words. DL bit, bit 30 of the first word, indicates that these two words are a buffer descriptor (DL = 1) or a link pointer (DL = 0).

An Rx buffer descriptor and an Rx link pointer are shown as Figure 4-16. Its address is word aligned. L bit indicates that the buffer pointed by this descriptor contains the last portion of a packet.

If there is no buffer directory to be linked, directory address of link pointer must be zero, as a null pointer.

Figure 4-15. Rx Buffer Descriptor/ Link Pointer



4.5.2.4 Rx data buffer

Rx Data buffer contains actual received cell data. Size of a buffer can vary from 1 byte to 64 kbytes. Its address is byte aligned.

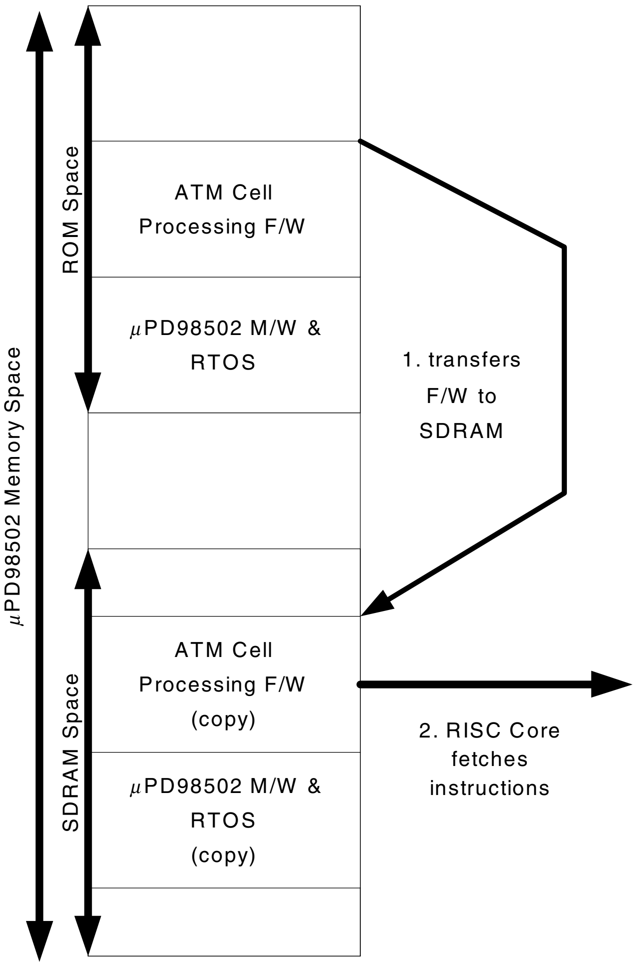
4.6 Initialization

This ATM Cell Processor is initialized by firmware that is based RISC instruction.

4.6.1 Before starting RISC core

RISC Core has 1 MB of Instruction space and 8 KB of physical Instruction RAM and 8 KB of instruction cache. The Instruction space will be mapped to the external system memory space. Address of instruction space will be translated by adding content of A_INBAR. VR4120A will set A_INBAR during initialization. Instruction memory space will be placed in the system memory space to achieve faster instruction fetch. VR4120A has to transfer RISC Core F/W to the assigned SDRAM area as well as its own S/W in its initialization, as shown in Figure 4-16. After transferring F/W, it sets base address of RISC Core F/W in A_INBAR. At the same time, base address of shared memory space has to be set in A_IBBAR. Then VR4120A let ATM Cell Processor to start operation.

Figure 4-16. Transfer of F/W

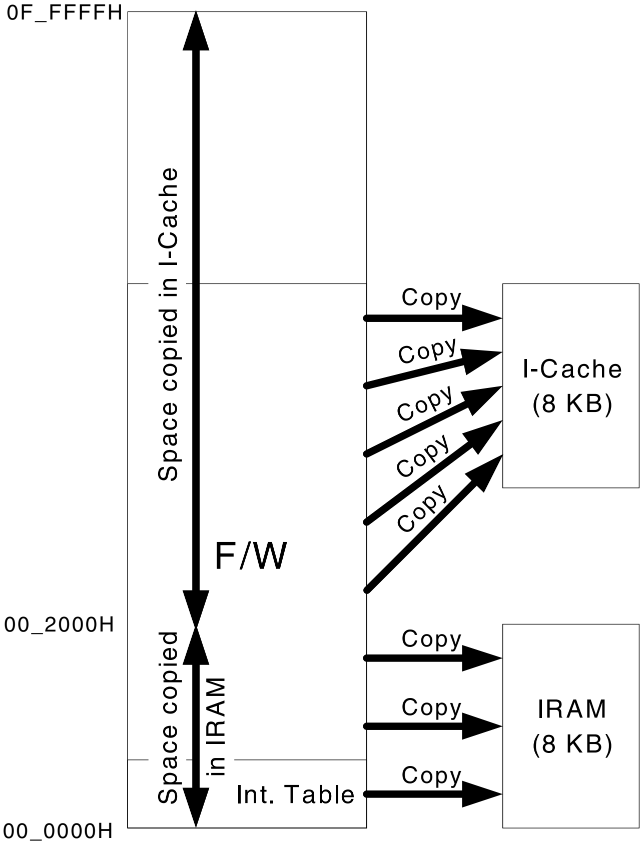

μPD98502 Memory Space
ROM Space
ATM Cell Processing F/W
μPD98502 M/W & RTOS
1. transfers F/W to SDRAM
SDRAM Space
ATM Cell Processing F/W (copy)
μPD98502 M/W & RTOS (copy)
2. RISC Core fetches instructions

4.6.2 After RISC core's F/W is starting

RISC Core starts its operation from address xx00_0000H. When it starts fetching an instruction located in address xx00_0000H, a dedicated H/W will stop RISC Core and will copy a block of instructions. This copy operation will be handled in the same manner as I-cache replacement.

Lower 8 KB of Instruction space in RISC Core will be copied on Instruction RAM because it will contain interruption vector table. The other part of the space is accessed through 8 KB of Instruction cache. In case that the total size of F/W is smaller than 16 KB, RISC Core can run fastest because once all necessary instruction code is copied in, no cache miss will occur.

Figure 4-17. Instruction RAM and Instruction Cache



4.7 Commands

Here, basic commands used in AAL-5 operation are described. Other commands used in AAL-2, OAM and cell switching functions are described in **μPD98502 Application Note (to be planned)**.

ATM Cell Processor provides VR4120A with the following basic commands.

Table 4-3. Commands

Command	What this block does
Set_Link_Rate	Set PHY Link Rate
Open_Channel	Reserves VC table area in Work RAM.
Close_Channel	Releases VC table area.
Tx_Ready	Starts transmission process.
Add_Buffers	Add Buffer Directories to the indicated pool
Indirect_Access	Enables VR4120A RISC Processor to access Work RAM

All commands are written in command register (A_CMR) and command extension register (A_CER) by VR4120A. Command register has busy flag. Since ATM Cell Processor only proceeds one command at a time, it sets the busy flag when it accepts the command. VR4120A cannot issue another command while this busy flag is 1. When finish the command operation, ATM Cell Processor sets the busy flag to 0. VR4120A has to read the busy flag of the command register and checks if busy bit is 0, before issues new command.

(1) Commands which ATM Cell Processor returns command indication

When ATM Cell Processor receives Open_Channel, Close_Channel, Open_IP_Channel, Close_IP_Channel, Tx_Ready and Add_Buffers command, it writes command indication in command register. VR4120A RISC Processor has to read command indication, after issuing these commands. However, while busy flag is 1, ATM Cell Processor has not finished command processing yet, so that, VR4120A RISC Processor has to wait until busy flag in command register becomes 0 in order to read the indication.

(2) Commands which command extension register is used

Indirect_Access command and Add_Buffers command use command extended register.

When VR4120A writes these commands, VR4120A has to write to command extension register first, and then command register. ATM Cell Processor starts command operation after command register is written. So that, unless command extension register is written first, the information in command extension register is ignored by ATM Cell Processor.

When command extension register is used for getting information from ATM Cell Processor, VR4120A writes to command register and wait until busy flag in command register becomes a ‘0’, and reads command extended register.

4.7.1 Set_Link_Rate command

This command is used to set the link rate of ATM PHY interface. After initializing ATM Cell Processor, this command has to be issued once, before any packet is transmitted.

Figure 4-18. Set_Link_Rate Command

[Set_Link_Rate command]

CMR

0	1	1	1	0	PHY No.	Link Rate
31 30	2 9	28	27	26 20	19 16	15 0

PHY No. PHY Number.

Link Rate Actual Link Rate is inserted in this field. The format of the rate is following.
$2^{e}(1 + m/512)$*nz cells/sec

Re- served	nz	e	m
15	14	13 9	8 0

4.7.2 Open_Channel command

This command is used to open a new channel to be used for a send or receive operation. When the channel is opened, ATM Cell Processor reserves the area for the VC table in Work RAM and returns the VC Number as an indication.

The indication that ATM Cell Processor returns for this command is of the following format:

Figure 4-19. Open_Channel Command and Indication

[Open_Channel command]

CMR

0	1	0	0	0	0
31 30	29	28	27	26	25 0

[Open_Channel command Indication]

CMR

Undefined	VC Number	0
31 18	17 6	5 0

VC Number VC Number of the opened channel. If no more new VCs can be opened, VC Number in the indication is set to 0.

4.7.3 Close_Channel command

The Close_Channel command is used to close a send or receive channel. Upon accepting this command, ATM Cell Processor returns the VC table to VC Table pool.

The indication that ATM Cell Processor returns for this command has the following format:

Figure 4-20. Close_Channel Command and Indication

[Close_Channel command]

CMR

0	1	0	0	1	R/T	0	VC Number	0
31 30	29	28	27	26	25	24 18	17 6	5 0

[Close_Channel command indication]

CMR

Undefined	E	Undefined	VC Number	0
31 25	24	23 18	17 6	5 0

Close_Channel command

R/T Indicates whether the channel to be closed is a send or a receive channel.

1: Receive channel

0: Transmit channel

VC Number VC Number of the channel that VR4120A intends to close.

Close_Channel command indication

E Error bit. If detects an error (ex. Invalid VC number), sets this bit to a ‘1’. When VR4120A issues this command, this bit has to be set to a ‘0’.

VC Number Closed VC Number of the channel. If the channel cannot be closed, 0 is set in this area.

4.7.4 Tx_Ready command

The Tx_Ready command is used by the VR4120A to notify ATM Cell Processor that a transmit packet has been added for a specified channel (a new packet descriptor has been set in system memory queue). Upon receiving this command, ATM Cell Processor makes the scheduling table active to perform scheduling. In this command, when it detects some errors, writes E bit in A_CMR.

This command has the following format:

Figure 4-21. Tx_Ready Command and Indication

[Tx_Ready command]

CMR

0	1	1	0	0	1	0	VC Number	0
31 30	29	28	27	26	25	24 18	17 6	5 0

CER

Packet Descriptor Address
31 0

[Tx_Ready command indication]

CMR

0	1	1	0	0	1	E	0	VC Number	0
31 30	29	28	27	26	25	24	23 18	17 6	5 0

Tx_Ready command

VC Number The VC number of the channel which VR4120A intends to start transmission.

Packet descriptor address Address of the packet descriptor.

Tx_Ready command indication

E Error bit. If detects an error (ex. Invalid VC number), sets this bit to a ‘1’. When VR4120A issues this command, this bit has to be a ‘0’.

VC Number The VC number of the channel which VR4120A intends to start transmission.

4.7.5 Add_Buffers command

The Add_Buffers command is used to add unused buffer directories to a single receive free buffer pool.

In this command, when ATM Cell Processor detects some errors, it writes E bit in A_CMR. This command has the following format:

Figure 4-22. Add_Buffers Command

[Add_Buffers command]

CMR

0	1	1	0	1	0	Pool No.	Number of Buffer Directories
31 30	29	28	27	26	25 21	20 16	15 0

CER

Buffer Directory Pointer
31 0

[Add_Buffers command indication]

CMR

0	1	1	0	1	0	E	0	Pool No.	Number of Buffer Directories
31 30	29	28	27	26	25	24	23 21	20 16	15 0

Add_Buffers command

Pool No. Number of the pool to which buffer directories are to be added. Since ATM Cell Processor supports only 8 pools (0 to 7), Bit 19 and 20 should be set to a ‘0’.

Number of Buffer Directories Number of new buffer directories to be added

Buffer Directory Pointer Start address of the first buffer directory in the list of new buffer directories to be added

Add_Buffers command indication

E Error bit. If detects an error, sets this bit to a ‘1’. When VR4120A issues this command, this bit has to be a ‘0’. The possible error is the number of buffer directories after this command is executed exceeds 65535.

Pool No. Number of the pool to which new buffer directories are just added

Number of Buffer Directories Number of new buffer directories just added

4.7.6 Indirect_Access command

The Indirect_Access command is used to perform read/write access to Work RAM.

Figure 4-23. Indirect_Access Command

[Indirect_Access command]

CMR

0	R/W	B3	B2	B1	B0	Address	0
31 29	28	27	26	25	24	23 2	1 0

CER

Data
31 0

Indirect_Access command

R/W Specifies whether access to the target is a read or a write access.

1: Read

0: Write

B0, B1, B2, B3 For write access, used to select bytes.

Address If the address specified in this area is within control memory, the virtual address is converted to ATM Cell Processor address. Otherwise, the address is used as is. The low-order two bits of the address must be 0, that is, the address must be a word address.

4.8 Operations

In this section, functional specifications mainly SAR function is described.

4.8.1 Work RAM usage

The size of the Work RAM is 16 Kbytes. This memory is used for following five purposes.

(1) Temporary data

The data which are exchanged with SDRAM using DMA. The data stored in this area are transmitting and receiving indications and first cell of IP packet.

(2) Flow table pool

The area in which Flow tables is stored.

(3) Packet Info structure

The area in which Packet Info structures is stored.

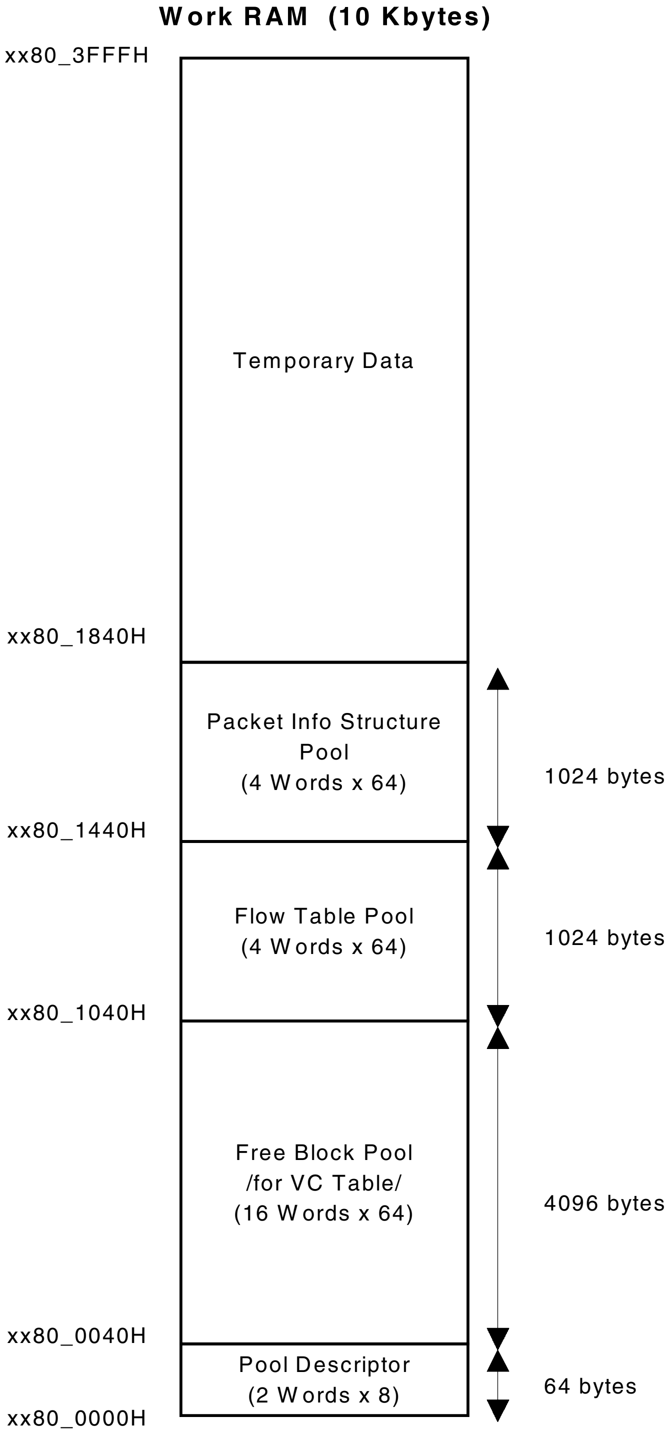
(4) Receive free buffer pool

The are in which “pool descriptors” is stored. Each pool descriptors uses 2 words.

(5) VC table pool

The area in which transmit and receive VC Tables is stored. Each VC table uses 1 block (16 words).

Figure 4-24. Work RAM Usage


Work RAM (10 Kbytes)
xx80_3FFFH
Temporary Data
xx80_1840H
Packet Info Structure Pool (4 Words x 64)
1024 bytes
xx80_1440H
Flow Table Pool (4 Words x 64)
1024 bytes
xx80_1040H
Free Block Pool /for VC Table/ (16 Words x 64)
4096 bytes
xx80_0040H
Pool Descriptor (2 Words x 8)
64 bytes
xx80_0000H

4.8.2 Transmission function

VR4120A sets the VC information in VC table prior to the every packet transmission, and issue Tx_Ready command with VC number in order to transmit packet belongs to the VC.

The transmitting data structure is described in Section **4.5.1 Tx buffer structure**.

4.8.2.1 Transmission procedure

(a) Setting transmitting data
Before transmitting a packet, VR4120A places a packet data to be sent in system memory and sets the packet descriptor.

(b) Opening the send channel
If VR4120A needs a new channel for transmitting of the packet data, VR4120A issues Open_Channel command. When VR4120A issues the command, ATM Cell Processor assigns a new VC Table pool block in Work RAM and reports its start address to VR4120A using a command indication.

(c) Setting the send VC table
The assigned 16-word VC Table pool block in Work RAM is set as the send VC table for each VC.

(d) Issuing the Tx_Ready command -> making preparations for transmission of the first cell
When VR4120A issues Tx_Ready command, ATM Cell Processor sets a Packet Info Structure in Work RAM, fetch the packet descriptor and store it in the area. ATM Cell Processor checks Transmit Queue if any packet is waiting for transmission. If Transmit Queue is not empty, ATM Cell Processor adds the Packet Info structure at the end of the queue. If no packet is waiting in the queue, ATM Cell Processor also schedules next transmission time with “current time plus 1”.

(e) Sending a cell

<1> Generating a header
A header is generated from Word1 in the VC table and written into SAR FIFO. "00H" is inserted into the GFC field of the header.

<2> Sending a segment data from system memory to SAR_FIFO
ATM Cell Processor reads a transmitting segment (48-byte payload data of the cell) from system memory and sets it in SAR_FIFO using Scatter/Gather DMA. The starting address of the segment is indicated by the "Buffer Read Address" field in the VC table. When the 53rd byte of the segment is written, SAR FIFO is updated.

<3> Calculating the CRC-32 value and the length
Each time a segment is read from SDRAM, the CRC-32 value is calculated for that segment and transmitted bytes are also counted. ATM Cell Processor writes those results in VC table.

<4> Updating the VC table
Updates "Buffer Read Address" and "Remaining Bytes in Current Buffer" fields.

(f) Sending the last cell
When the L flag of the current transmit buffer indicates that the buffer is the last one and the value in the field indicating the number of bytes remaining in the VC table is less than 40 bytes, the cell is the last cell of the packet.

<1> When the current cell is the last cell of the packet, and the remaining payload data is less than 40 bytes, zero padding and the 8 byte trailer are added. When the remaining payload data is more than 40 bytes and there are not enough space to add an 8-byte AAL-5 trailer, ATM Cell Processor just adds zero padding to make a 48-byte payload and ATM Cell Processor sends a cell containing only a trailer and padding next.

<2> When the last segment of the AAL-5 PDU is read, the final CRC-32 value and the packet length are inserted into the trailer of the AAL-5 PDU, and the contents of the first word in the VC table are inserted into the CPCS-UU and CPI fields, thereby completing the AAL-5 trailer.

(g) ATM Cell Processor checks whether there is a subsequent packet (checks Last Packet Info address and First Packet Info address in Tx VC table). When a subsequent packet exists, (e) and (f) are repeated.

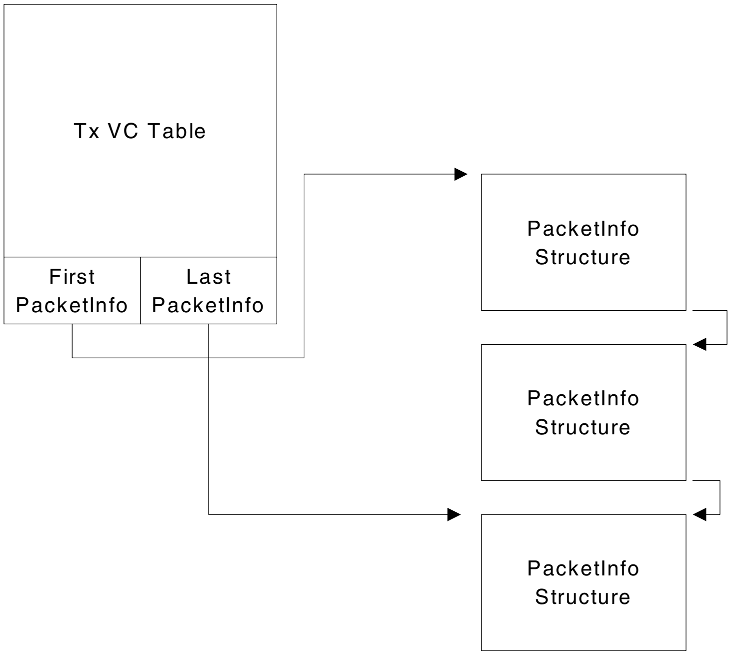
(h) For each packet, ATM Cell Processor stores transmitting indication as a status information in the mailbox and generates an interrupt.

(i) VR4120A reads the mailbox and updates the read pointer of the mailbox.

4.8.2.2 Transmit queue

Tx_Ready command has to be issued in order to transmit a packet. However, VR4120A doesn't have to wait Tx indication before issuing next Tx_Ready command for the same VC. When VR4120A issues Tx_Ready command before completing transmission process for the previous packet, ATM Cell Processor builds Tx Queue for that VC.

Figure 4-25. Structure of the Transmit Queue



(1) Packet info structure

The Maximum number of Packet Info Structure for each VC is 16. However, since total number of Packet Info Structure is 128, some VC may not obtain 16 Packet Info Structure depend on the queue length of other VCs. When ATM Cell Processor can not obtain any Packet Info Structure, ATM Cell Processor returns an error indication for Tx_Ready command.

Figure 4-26. Packet Info Structure

WORD0	CELL HEADER
WORD1 WORD2	PACKET DESCRIPTOR STORAGE
WORD3	NEXT POINTER

CELL HEADER	Cell header
PACKET DESCRIPTOR STORAGE	Area for temporarily storing the packet descriptor of a packet
NEXT POINTER	Address of the next Packet Info structure

(2) Packet descriptor

Figure 4-27. Transmit Queue Packet Descriptor

1	0	ENC	CLPM	PTI	GFC	IM	C10	AAL	MB	CPCS-UU	CPI
31	30	29	28 27	26 24	23 20	19	18	17	16	15 8	7 0

Buffer Directory Address
31 0

ENC Encapsulation mode is indicated.
1 LLC encapsulation
0 No encapsulation

CLPM CLP bit in the packet header is indicated.
00 Sets CLP bit in all cells as 0.
11 Sets CLP bit in all cells as 1.
01 Sets CLP bit in all cells except last cell as 1, and only CLP bit in last cell as 0.
10 Not used.

PTI PTI field in the packet header. Since this field indicates cell payload type, takes different transmission procedure according to this field .
000 User data cell without congestion.
001 User data cell without congestion. Last cell in the packet.
Prohibited to be set by user. If this value is set, treats as 000.
010 User data cell with congestion.
011 User data cell with congestion. Last cell in the packet.
Prohibited to be set by user. If this value is set, treats as 010.
100 OAM F5 cell used to carry information between segments
101 OAM F5 cell used to carry information between end to end.
110 Prohibited to be set by user.
111 Prohibited to be set by user.

GFC GFC field in the packet header is indicated.

IM Indicates if ATM Cell Processor issues the transmitting indication to VR4120A.
1: Doesn’t send any transmitting indication to VR4120A.
0: Sends transmitting indications to VR4120A.

C10 Indicates if CRC-10 should be inserted or not.
1 Insert CRC-10 in every cell.
0 Not insert CRC-10.

AAL Indicates if the transmitting packet is AAL-5 packet or not.
1 AAL-5 mode.
0 Raw cell mode. ATM Cell Processor treats cells in this packet as Raw cell.

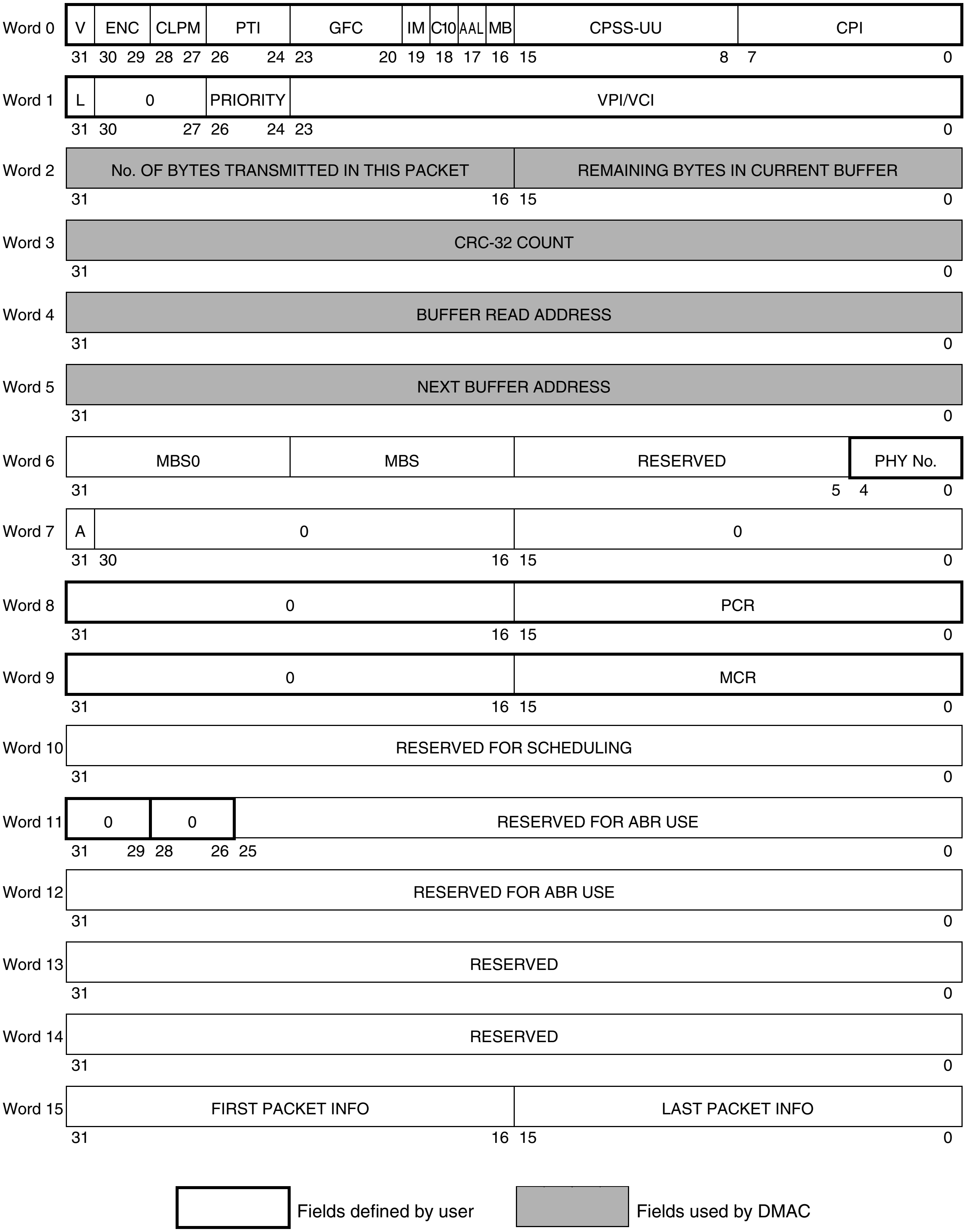
MB Mailbox number for storing transmitting indication for this packet is indicated.
1 Mailbox number 3.
0 Mailbox number 2.

CPCS-UU User information. In AAL-5 mode, ATM Cell Processor inserts the pattern in this field to the CPI field in the trailer. In Raw cell mode, this field is ignored.

CPI User information. In AAL-5 mode, ATM Cell Processor inserts the pattern in this field to the CPI field in the trailer. In Raw cell mode, this field is ignored.

(3) Tx VC table

Figure 4-28. Tx VC Table



Word0
Identical to the contents of Word0 in the packet descriptor in system memory. The initial value must be all zeros. ATM Cell copies the Word0 in the packet descriptor into this field.

L
This bit is used internally for SAR processing. The initial value must always be a 1.

PRIORITY
Specifies send priority.
111: CBR
110: VBR
010: UBR

VPI/VCI
VPI/VCI values to be contained in the cell header

No. OF BYTES TRANSMITTED IN THIS PACKET
Number of bytes in the packet that have been sent so far. The initial value is 0.

REMAINING BYTES IN CURRENT BUFFER
Number of bytes in the current buffer that have not been sent. The initial value is 0.

CRC-32 COUNT
CRC-32 value calculated for the transmitted cells of this packet. The final CRC-32 value is set in the CRC-32 field in the trailer.

BUFFER READ ADDRESS
Pointer to the byte in the current buffer that is to be transferred next Tx time.

NEXT BUFFER ADDRESS
Pointer to the next buffer in the current packet directory

MBS0
Maximum Burst Size. The number of cells continuously transmitted with PCR. Only used in VBR mode.

MBS
Internally used for counting MBS0. Initially, the same value as MBS0 has to be set. Only used in VBR mode.

PHY No.
The number of the PHY which cells that belongs to this VC transmits to.

A
"Active" bit indicating whether the VC table is in the active or idle state.
1: Active state
0: Idle state

MCR
Minimum Cell Rate. In VBR mode, SCR has to be set in this field.

PCR
Peak Cell Rate.

Word 10
These fields are used internally for scheduling use.

Word 11 & Word 12
These fields are used internally.

FIRST PACKET INFO
The start address of the first Packet Info structure in transmit queue for this VC. The initial value has to be 0.

LAST PACKET INFO
The start address of the last Packet Info structure in transmit queue for this VC. The initial value has to be 0.

4.8.2.3 Non AAL-5 traffic support

(1) OAM F5 cell transmission

When host sets OAM F5 cell pattern (100 and 101) in the PTI field in packet descriptor, ATM Cell Processor doesn’t add AAL-5 trailer. In this case, even though host sets more than 48 bytes in “SIZE” field in the packet descriptor, ATM Cell Processor only reads 48 bytes from the top of the data buffer and ignores the data after that. If host sets the bytes less than 48 byte in “SIZE” field, ATM Cell Processor adds padding. For OAM F5 cell transmission, host has to set different packet descriptor for each OAM F5 cell.

For OAM F5 cell transmission, ATM Cell Processor inserts CRC-10, if host sets “C10 bit” to 1 in packet descriptor. ATM Cell Processor calculates CRC-10 for 46 bytes and 6 bits and overwrites the result to the last 10 bits in the payload.

In addition, F/W for ATM Cell Processor supports Forward Monitoring and Backward Reporting functions. The detail of the functions will be announced.

(2) Raw cell transmission

When host sends the non AAL-5 traffic packet which is not OAM F5 cell, host sets “AAL” bit in the packet descriptor to a 0 and “PTI” field “0xx” which indicates user data. In this case, ATM Cell Processor doesn’t calculate or add AAL-5 trailer.

If host sets “C10” bit in packet descriptor to a 1, ATM Cell Processor calculates and adds CRC-10 to each cell to be transmitted.

Figure 4-29. Raw Cell with CRC-10

Header 5 bytes	Payload 46 bytes and 6bits	CRC-10 10 bits

Generation polynomial of CRC-10 is following:

$$G(x) = 1 + x + x^4 + x^5 + x^9 + x^{10}$$

4.8.2.4 Transmission indication

For each transmitted packet, ATM Cell Processor writes a send indication as a transmission completion status in the mailbox. The mailbox used for transmission is mailbox 2 and 3. More specifically, ATM Cell Processor writes a send indication once all the data in the packet has been read. The issuing of a send indication, therefore, does not indicate that the sending of the packet to PMD has been completed.

Upon storing a send indication into the mailbox, ATM Cell Processor sets the corresponding MM bit of the A_GSR register to a 1, and issues an interrupt if it is not masked.

The indication that ATM Cell Processor sends to the host during transmission is of the following format:

Figure 4-30. Send Indication Format

E	VC Number	A	PACKET QUEUE POINTER
31	30 16	15	14 0

E (Error ID) Error ID indicates error condition.
0: Error
1: No error

VC Number VC Number used for this VC

A (for Active) If 0, indicates that the VC enters the idle state because the packet descriptor is invalid.
If 1, indicates that the VC is kept active because the next packet descriptor is valid.

Packet Queue Pointer Low-order 15 bits of the start address of the packet descriptor which is just transmitted.

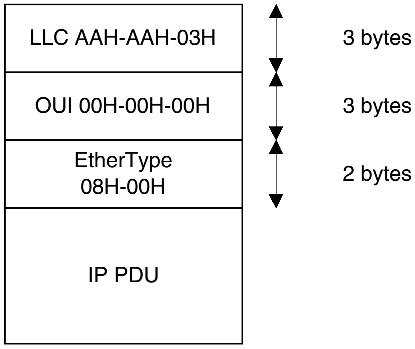
4.8.2.5 Scheduling

ATM Cell Processor holds a scheduling table in which ATM Cell Processor sets the transmitting timings of all active channels. Transmitting timing is recalculated each time ATM Cell Processor transmits a cell. Transmitting timing is calculated using line rate and rate information which is set by VR4120A prior to the Tx_Ready command. Rate information is written in Tx VC table.

4.8.2.6 LLC encapsulation

If LLC encapsulation is indicated in Tx VC table, ATM Cell Processor adds the LLC header to the top of the IP packet. ATM Cell Processor always encapsulates CPCS-PDU as Internet IP PDU.

Figure 4-31. LLC Encapsulation Format



4.8.3 Receiving function

Receiving data structure is described in Section **4.5.2 Rx pool structure**.

4.8.3.1 Receiving procedure

(a) Setting up a receive pool
Before receiving a packet, VR4120A prepares the receive pool to store receive cell data and sets the information about the pool in the pool descriptor in Work RAM.

(b) Opening the receive channel
When VR4120A is going to open a new connection, VR4120A issues Open_Channel command. If Open_Channel command is issued, ATM Cell Processor assigns a VC Table block in Work RAM and reports its start address to VR4120A using a command indication. VR4120A sets the assigned block as a VC table.

(c) Setting the receive VC table
VR4120A sets the 16-word assigned blocks in Work RAM as the receive VC table for each VC.

(d) Setting up the Rx lookup table
The VPI/VCI of the receive cell are set in the Rx lookup table.

(e) Receiving the first cell of the packet
Upon ATM Cell Processor receiving a cell, it checks if the VPI/VCI of the received cell has been registered in Rx lookup table. If not registered, the cell is discarded. If registered, ATM Cell Processor gets the VC number from Rx lookup table. If it is a first cell of a packet, ATM Cell Processor assigns the new buffer directory. The VC is added to the T1 timer list.

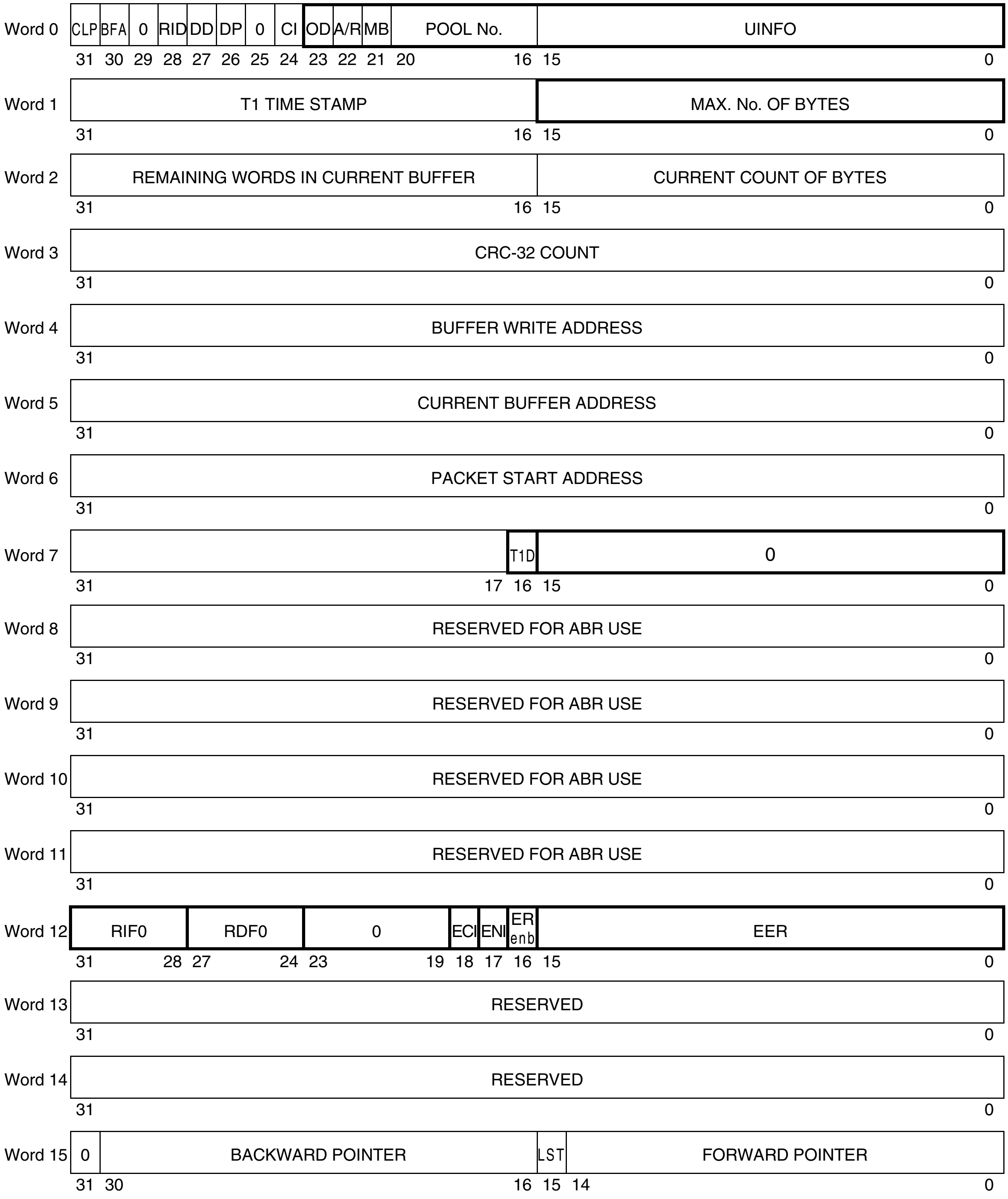
(f) Receiving a cell
ATM Cell Processor transfers the received cell data from SAR FIFO to system memory using Scatter/Gather DMA. In the same way as in transmission, the CRC-32 value, calculated for each received cell data, is stored into the "CRC-32" field of the VC table. The “Current Count of Bytes” and “Remaining words in current buffer” fields, in the VC table, are updated.

(g) Receiving the last cell
<1> The trailer information is checked for errors.
<2> A receive indication is stored into the mailbox and an interruption is issued.

(h) Reading the receive indication
VR4120A reads the receive indication, updates the read pointer in the mailbox, and extract received data from the pool.

(1) Rx VC table

Figure 4-32. Receive VC Table



▭ Fields which are defined by user

CLP	Set to a 1 if the CLP in the header of at least one cell of the packets being received is equal to a 1.
BFA	Set to a 1 if the free buffer assigned to this VC exists.
RID	Set to a 1 if an error occurs while a packet is being received. Then, the subsequent cells of the packet, including the last cell, are discarded.
DD	If a free buffer is not assigned, the cell is discarded and this field is set to a 1.
DP	Always a 0 because it is not used by ATM Cell Processor.
MB	Mailbox number.
POOL No.	Pool number. (ATM Cell Processor supports pool numbers 0 to 7.)
UINFO	User information
T1 TIMESTAMP	Area used for T1 timer calculation
CI	Congestion notification.
OD	OAM cell reception/discard indication bit.
A/R	AAL-5/RAW cell reception indication bit.
MAX. No. OF BYTES	Maximum number of bytes in one packet
REMAINING WORDS IN CURRENT BUFFER	Number of words of the free area remaining in the current buffer
CURRENT COUNT OF BYTES	Number of bytes of the current packet that have been received so far
CRC-32 COUNT	Used for CRC-32 value calculation
BUFFER WRITE ADDRESS	Next address in the currently assigned free buffer that is used for storage
CURRENT BUFFER POINTER	Start address of the free buffer to be assigned next time.
PACKET START ADDRESS	Start address of the packet (start address of the batch assigned first)
WORD 6 to WORD 12	These fields are used internally.
BACKWARD POINTER	VC Number of the VC linked before this VC in the T1 link list
LST	Set to a 1 if the VC is the last one linked in the T1 link list.
FORWARD POINTER	VC Number of the VC linked after this VC in the T1 link list

4.8.3.2 Non AAL-5 traffic support

Every time ATM Cell Processor receives a raw cell, it makes a raw cell data with 53 byte raw cell and 11 byte indication and stores it to the appropriate Rx pool. Then, ATM Cell Processor sets corresponding bits in A_GSR register and issues an interruption if not masked.

Since ATM Cell Processor treats raw cells as a unit of cell not a packet, it doesn’t set rx indications in Rx mailbox. When ATM Cell Processor receives raw cells, CRC-10 verify function is always enable. If ATM Cell Processor detects CRC-10 error, sets error bit in raw cell data.

(1) OAM F5 cell

When OD bit in VC table is a 1 and PTI field in received ATM cell header is “1xx”, ATM Cell Processor generates Raw cell data and stores it in pool 0. It sets PCR0=1 in A_GSR register and issues an interruption to VR4120A, if not masked. ATM Cell Processor always stores these data in Pool 0. If OD bit is set to a 1, Pool 0 has to be set for Raw cell data.

(2) Non AAL-5 traffic

When A/R bit in VC table is a 0, ATM Cell Processor treats received cells that belong to the VC as raw cells. If receives raw cells, ATM Cell Processor has to assign a pool to raw cell data.

(3) Raw cell data

The following is the Raw cell data format in little endian mode.

Figure 4-33. Raw Cell Data Format

<table>
<tr><td>WORD0</td><td colspan="4">CELL HEADER</td></tr>
<tr><td>WORD1</td><td>BYTE2</td><td>BYTE1</td><td>BYTE0</td><td>HEC</td></tr>
<tr><td>:
:
:</td><td></td><td></td><td></td><td></td></tr>
<tr><td>WORD12</td><td>BYTE46</td><td>BYTE45</td><td>BYTE44</td><td>BYTE43</td></tr>
<tr><td>WORD13</td><td colspan="2">UINFO</td><td>0</td><td>BYTE47</td></tr>
<tr><td>WORD14</td><td colspan="4">TIME STAMP</td></tr>
<tr><td>WORD15</td><td>1</td><td>VC NUMBER</td><td>CE</td><td>0</td></tr>
</table>

Cell Header	Header of the cell except HEC.
HEC	HEC field pattern of the cell.
BYTE0 – BYTE47	Payload data of the cell.
UINFO	User information. The pattern which user set in UINFO field in VC table.
TIME STAMP	A_TSR register value when ATM Cell Processor received the cell.
VC NUMBER	VC Number of the cell.
CE	CRC-10 check result 0: No error. 1: CRC-10 error is detected.

4.8.3.3 Receive indication

For each packet, ATM Cell Processor writes a receive indication as the reception completion status in the mailbox. The mailbox used for reception is mailbox 0 and 1. More specifically, ATM Cell Processor writes a receive indication at the following case:

(1) All cell data that belong to a packet are received.

- No error is detected
- "CRC-32 error" or "Length error " is detected

(2) An error is detected before the last cell of a packet is received. The error is other than "CRC-32 error" or "Length error ".

A receive indication contains the start address and size of the batch that ATM Cell Processor uses to store the cell, and other information. By reading the receive indication, the VR4120A can process the received packet that is composed of received cells.

The format of a receive indication is as follows:

Figure 4-34. Receive Indication Format

UINFO	PACKET SIZE
31 16	15 0

TIME STAMP
31 0

PACKET START ADDRESS
31 0

1	VC Number	ERR	CI	CLP	0	ERR STATUS	0	POOL No.
31	30 16	15	14	13	12	11 8	7 5	4 0

UINFO Pattern set by the host in the UINFO field in the VC table

PACKET SIZE Size of the receive packet in cell units

TIME STAMP Value of A_TSR when this indication is issued

PACKET START ADDRESS Start address of the batch used

CHANNEL NUMBER VC Number used by this VC

ERR If 1, indicates that an error occurred while the packet was being received.
If 0, indicates that the packet was received normally.

CI The PTI field in the header of at least one cell of the packet indicated congestion.

CLP The CLP field in the header of at least one cell of the packet was equal to 1.

ERROR STATUS Status of the error that occurred

0000	No error
0001	Free buffer underflow
0011	Max. number of bytes violation
0100	CRC-32 error
0110	Length error
0111	T1 time-out

POOL NUMBER Number of the pool used

4.8.3.4 Receive errors

ATM Cell Processor checks errors while a packet is being received and also after the packet has been received. Upon detecting an error, it reports the type of error, the start address and the amount of data that had been transferred to system memory prior to the error being detected, using the receive indication in the mailbox.

Upon receiving a receive indication containing the error status, the VR4120A takes appropriate action and discards the packet that caused the error. The types of reception errors are described below:

(1) Free buffer underflow

This error occurs if the free area in the free buffer is equal to or less than 48 bytes when a packet is received. When this error occurs, an A_RQU interrupt is generated. If the discarded cell is an intermediate cell or the last cell of the packet, reception of the packet is suspended. A receive indication reporting the free buffer underflow error is issued, and the RID bit in the VC table is set. The remaining cells of the packet, including the last cell, are discarded, and the RID bit is referenced. When the last cell is received, the RID bit is reset. If the cell discarded due to this error is the first cell of the packet, an A_RQU interrupt is generated and the RID bit is set. No receive indication, however, is issued.

(2) Max No. of bytes violation

This error occurs if the last cell of a packet has not been received when the number of cells received has reached the user-specified "Max. No. of bytes" When the next cell is received, the RID bit is set and a receive indication is issued. The subsequent cells of the packet, including the last cell, are discarded.

(3) CRC-32 error

This error occurs if the CRC-32 value does not match the CRC-32 value contained in the receive trailer when all of cell data in the packet has been received. If a check made on the trailer reveals that the received packet has both CRC-32 and "length" errors, the CRC-32 error is reported as the error status in the receive indication.

(4) "length" error

This error is reported if the "length" field contained in the receive trailer and the calculated packet length satisfy either of the following conditions:

("Number of received cells x 48 bytes" – ""length" value in the trailer") > 55 bytes
("Number of received cells x 48 bytes" – ""length" value in the trailer") < 8 bytes

(5) T1 time-out

This error occurs if the last cell of a packet has not been received even after the user-specified A_T1R time has elapsed since the first cell was received. When this error occurs, the RID bit is set and the remaining cells of the packet that caused this error, including the last cell, are discarded.

The table below lists the errors that can occur with any of the first cell, the immediate cells, and the last cell of a packet.

Table 4-4. Reception Errors That Can Occur During Packet Reception

Error	When a cell is discarded	When a receive indication is issued	Handling of other cells after the error occurred
Free buffer underflow	The cell received when the size of the free buffer is insufficient is discarded.	Time at which transfer becomes impossible during segment transfer	The subsequent cells of the packet, including the last cell, are discarded.
MAX No. of Segments error	The cell received after MAX No. of Segments was reached is discarded.	Time at which the next cell is received after MAX No. of Segments was reached	The subsequent cells of the packet, including the last cell, are discarded.
T1 error	The cell received after the T1 time has elapsed is discarded.	Time at which T1 time has elapsed.	The subsequent cells of the packet, including the last cell, are discarded.

Two or more errors may occur at the same time; however, only one is reported with a receive indication. The table below lists the error reporting priorities.

Table 4-5. Error Reporting Priorities

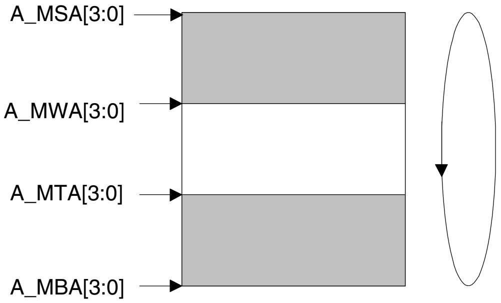
	Underflow	MAX error	CRC error	Length	T1 error
Underflow	-	Underflow	Underflow	Underflow	Underflow
MAX error	-	-	MAX error	MAX error	MAX error
CRC error	-	-	-	CRC error	CRC error
Length	-	-	-	-	Length
T1 error	-	-	-	-	-

4.8.4 Mailbox

ATM Cell Processor uses mailboxes as ring buffers in system memory. The structure of a mailbox and the defined addresses are as follows.

Mailbox start address (A_MSA[3:0]) :The start address of the mailbox
Mailbox bottom address (A_MBA[3:0]) :The bottom address of the mailbox (address following the last address)
Mailbox write address (A_MWA[3:0]) :The write pointer
Mailbox tail address (A_MTA[3:0]) :The tail address that has been read by the host and which is to be updated

Figure 4-35. Mailbox Structure



Upon writing an indication, increments the write pointer (A_MWA[3:0]), sets the MM bit for the corresponding mailbox in the A_GSR register, and issues an interrupt if it is not masked. When updating the write pointer (A_MWA[3:0]), ATM Cell Processor causes A_MWA[3:0] to jump to the start address (A_MSA[3:0]) if A_MWA[3:0] has reached the bottom address (A_MBA[3:0]). To read an indication, VR4120A uses the read pointer (A_MTA[3:0]). A_MTA[3:0] is managed by the VR4120A: Each time VR4120A reads an indication from the mailbox, it writes the address of the next indication to the read pointer (A_MTA[3:0]). If the write pointer (A_MWA[3:0]) points to the same address as that pointed to by the read pointer (A_MTA[3:0]), sets the MF bit of the A_GSR register that corresponds to the mailbox to indicate that the mailbox is full (the MF state), and issues an interrupt if it is not masked. Once the mailbox enters the MF state, ATM Cell Processor does not execute any commands.

CHAPTER 5 ETHERNET CONTROLLER

5.1 Overview

This section describes Ethernet Controller block. This Ethernet Controller block comprises of a 10/100 Mbps Ethernet MAC (Media Access Control), data transmit/receive FIFOs, DMA and internal bus interface.

The μPD98502 implements 2-channel Ethernet Controller.

5.1.1 Features

- IEEE802.3/802.3u/802.3x Compliant:
  - 10/100 Mbps Ethernet MAC
  - Media Independent Interface (MII)
  - Flow Control
- Full duplex operation for 10 Mbps and 100 Mbps
- Address Filtering:
  - unicast
  - multicast
  - broadcast
- Statistics counters for management information
  - RMON
  - SNMP MIB
- Large independent receive and transmit FIFOs
- Direct Memory Access (DMA) with programmable burst size providing for low CPU utilization
- Internal Bus interface (IBUS): 32 bits @66 MHz

5.1.2 Block diagram of Ethernet controller block

The following list describes this block's hardware components, and **Figure 5-1** shows a block diagram of this block:

IBUS Interface - Data transfer is done through this IBUS interface between CPU and Ethernet Controller or memory and Ethernet Controller. This performance is approximately 2 Gbps (32 bits x 66 MHz). Also in this block, IBUS protocol operation is done [retry, disconnect and so on].

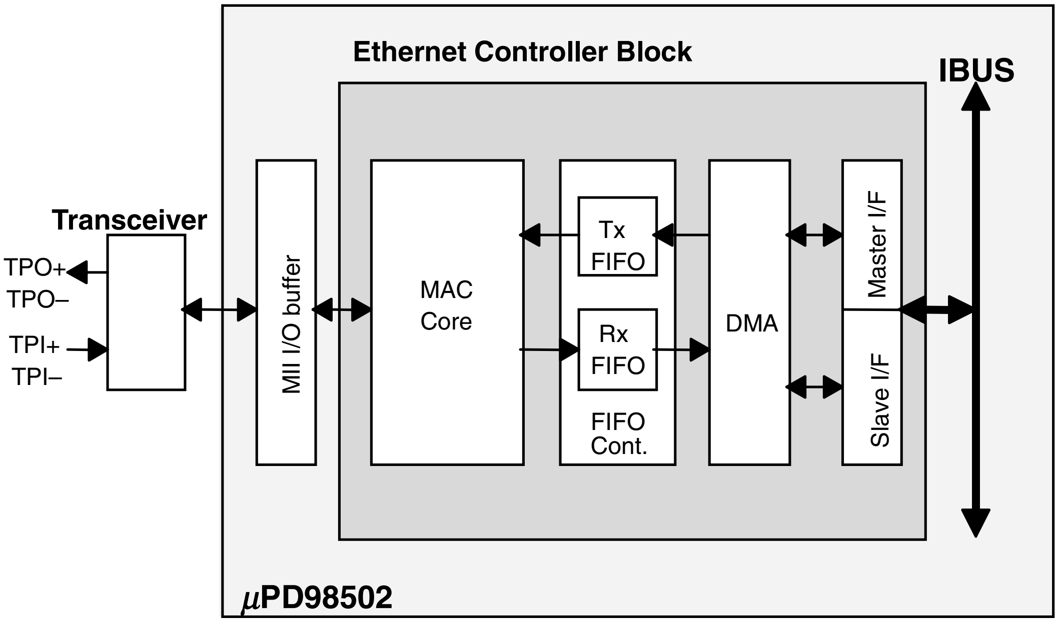
DMA Controller - DMA controller contains dual reception and transmission controller, which handle data transfers between memory and FIFOs. This DMA controller supports a byte alignment.

FIFO Controller - FIFO controller contains two independent FIFOs for reception and transmission. This controller supports automatic packet deletion on reception (after a collision or address filtering) and packet retransmission after a collision on transmission.

MAC Core - MAC Core is fully compliant with IEEE802.3, IEEE802.3u and IEEE802.3x.

MII I/O Buffer - MII I/O buffers are compliant IEEE802.3u and are connect to standard PHY device.

Figure 5-1. Block Diagram of Ethernet Controller



5.2 Registers

Registers of this block are categorized following four categories as shown in **Table 5-1**.

VR4120A controls following registers.

The μPD98502 has 2-channel Ethernet Controller, #1 controller's base address is 1000_2000H, #2 controller's base address is 1000_3000H.

Table 5-1. Ethernet Controller's Register Categories

Offset Address	Register Categories
1000_2000H:1000_213FH (Ethernet Controller #1), 1000_3000H:1000_313FH (Ethernet Controller #2)	MAC Control Registers
1000_2140H:1000_21FFH (Ethernet Controller #1), 1000_3140H:1000_31FFH (Ethernet Controller #2)	Statistics Counter Registers
1000_2200H:1000_2233H (Ethernet Controller #1), 1000_3200H:1000_3233H (Ethernet Controller #2)	DMA and FIFO Management Registers
1000_2234H:1000_223FH (Ethernet Controller #1), 1000_3234H:1000_323FH (Ethernet Controller #2)	Interrupt and Configuration Registers

5.2.1 Register map

5.2.1.1 MAC control registers

MAC Control Registers' map is shown in **Table 5-2**.

Table 5-2. MAC Control Register Map

Offset Address	Register Name	R/W	Access	Description
1000_m000H	En_MACC1	R/W	W	MAC Configuration Register 1
1000_m004H	En_MACC2	R/W	W	MAC Configuration Register 2
1000_m008H	En_IPGT	R/W	W	Back-to-Back IPG Register
1000_m00CH	En_IPGR	R/W	W	Non Back-to-Back IPG Register
1000_m010H	En_CLRT	R/W	W	Collision Register
1000_m014H	En_LMAX	R/W	W	Max Packet Length Register
1000_m018H: 1000_m01CH	N/A	-	-	Reserved for future use
1000_m020H	En_RETX	R/W	W	Retry Count Register
1000_m024H: 1000_m050H	N/A	-	-	Reserved for future use
1000_m054H	En_LSA2	R/W	W	Station Address Register 2
1000_m058H	En_LSA1	R/W	W	Station Address Register 1
1000_m05CH	En_PTVR	R	W	Pause Timer Value Read Register
1000_m060H	N/A	-	-	Reserved for future use
1000_m064H	En_VLTP	R/W	W	VLAN Type Register
1000_m080H	En_MIIC	R/W	W	MII Configuration Register
1000_m084H: 1000_m090H	N/A	-	-	Reserved for future use
1000_m094H	En_MCMD	W	W	MII Command Register
1000_m098H	En_MADR	R/W	W	MII Address Register
1000_m09CH	En_MWTD	R/W	W	MII Write Data Register
1000_m0A0H	En_MRDD	R	W	MII Read Data Register
1000_m0A4H	En_MIND	R	W	MII Indicator Register

Offset Address	Register Name	R/W	Access	Description
1000_m0A8H: 1000_m0C4H	N/A	-	-	Reserved for future use
1000_m0C8H	En_AFR	R/W	W	Address Filtering Register
1000_m0CCH	En_HT1	R/W	W	Hash Table Register 1
1000_m0D0H	En_HT2	R/W	W	Hash Table Register 2
1000_m0D4H: 1000_m0D8H	N/A	-	-	Reserved for future use
1000_m0DCH	En_CAR1	R/W	W	Carry Register 1
1000_m0E0H	En_CAR2	R/W	W	Carry Register 2
1000_m0E4H: 1000_m012CH	N/A	-	-	Reserved for future use
1000_m130H	En_CAM1	R/W	W	Carry Mask Register 1
1000_m134H	En_CAM2	R/W	W	Carry Mask Register 2
1000_m138H: 1000_m13CH	N/A	-	-	Reserved for future use

Remarks 1. In the “Offset Address” field and in the “Register Name” field,
Ethernet Controller #1: m = 2, n = 1,
Ethernet Controller #2: m = 3, n = 2

2. In the “R/W” field,
“W” means “writeable”,
“R” means “readable”,
“RC” means “read-cleared”,
“- “ means “not accessible”.

3. All internal registers are 32-bit word-aligned registers.

4. The burst access to the internal register is prohibited.
If such burst access has been occurred, IRERR bit in NSR is set and NMI will assert to CPU.

5. Read access to the reserved area will set the CBERR bit in the NSR register and the dummy read response data with the data-error bit set on SysCMD [0] is returned.

6. Write access to the reserved area will set the CBERR bit in the NSR register, and the write data is lost.

7. In the “Access” filed,
“W” means that word access is valid,
“H” means that half word access is valid,
“B” means that byte access is valid.

8. Write access to the read-only register cause no error, but the write data is lost.

9. The CPU can access all internal registers, but IBUS master device cannot access them.

5.2.1.2 Statistics counter registers

MAC Control Block gathers statistical information about the receive and transmit operations from the statistics counters.

Each counter consists of 32-bit counter. When a counter overflow condition occurs, the corresponding bit of the En_CAR1 register or En_CAR2 register is set to a 1, and it will generate an interrupt. The interrupt can be masked by setting the bits of En_CAM1 register or En_CAM2 register.

A moment is required to complete the updating of all statistics counters after the change of the status vector (TSV or RSV) because of the count operation.

Table 5-3. Statistics Counter Register Map

Offset Address	Register Name	R/W	Access	Description
1000_m140H	En_RBYT	R/W	W	Receive Byte Counter
1000_m144H	En_RPKT	R/W	W	Receive Packet Counter
1000_m148H	En_RFCS	R/W	W	Receive FCS Error Counter
1000_m14CH	En_RMCA	R/W	W	Receive Multicast Packet Counter
1000_m150H	En_RBCA	R/W	W	Receive Broadcast Packet Counter
1000_m154H	En_RXCF	R/W	W	Receive Control Frame Packet Counter
1000_m158H	En_RXPF	R/W	W	Receive PAUSE Frame Packet Counter
1000_m15CH	En_RXUO	R/W	W	Receive Unknown OP code Counter
1000_m160H	En_RALN	R/W	W	Receive Alignment Error Counter
1000_m164H	En_RFLR	R/W	W	Receive Frame Length Out of Range Counter
1000_m168H	En_RCDE	R/W	W	Receive Code Error Counter
1000_m16CH	En_RFCR	R/W	W	Receive False Carrier Counter
1000_m170H	En_RUND	R/W	W	Receive Undersize Packet Counter This counter is incremented each time a frame is received which is less than 64 bytes in length and contains a valid FCS.
1000_m174H	En_ROVR	R/W	W	Receive Oversize Packet Counter This counter is incremented each time a frame is received which exceeds 1518 bytes length and contains a valid FCS.
1000_m178H	En_RFRG	R/W	W	Receive Error Undersize Packet Counter This counter is incremented each time a frame is received which is less than 64 bytes in length and contains a invalid FCS, includes alignment error.
1000_m17CH	En_RJBR	R/W	W	Receive Error Oversize Packet Counter This counter is incremented each time a frame is received which exceeds 1518 bytes length and contains an invalid FCS or includes an alignment error.
1000_m180H	En_R64	R/W	W	Receive 64 Byte Frame Counter This counter is incremented for each good or bad frame received which is 64 bytes in length.
1000_m184H	En_R127	R/W	W	Receive 65 to 127 Byte Frame Counter This counter is incremented for each good or bad frame received which is 65 to 127 bytes in length.
1000_m188H	En_R255	R/W	W	Receive 128 to 255 Byte Frame Counter This counter is incremented for each good or bad frame received which is 128 to 255 bytes in length.
1000_m18CH	En_R511	R/W	W	Receive 256 to 511 Byte Frame Counter This counter is incremented for each good or bad frame received which is 256 to 511 bytes in length.
1000_m190H	En_R1K	R/W	W	Receive 512 to 1023 Byte Frame Counter This counter is incremented for each good or bad frame received which is 512 to 1023 bytes in length.
1000_m194H	En_RMAX	R/W	W	Receive Over 1023 Byte Frame Counter This counter is incremented for each good or bad frame received which is over 1023 bytes in length.
1000_m198H	En_RVBT	R/W	W	Receive Valid Byte Counter This counter is incremented by the byte count of each valid packet received.
1000_m19CH: 1000_m1BCH	N/A	-		Reserved for future use
1000_m1C0H	En_TBYT	R/W	W	Transmit Byte Counter

Offset Address	Register Name	R/W	Access	Description
1000_m1C4H	En_TPCT	R/W	W	Transmit Packet Counter
1000_m1C8H	En_TFCS	R/W	W	Transmit CRC Error Packet Counter
1000_m1CCH	En_TMCA	R/W	W	Transmit Multicast Packet Counter
1000_m1D0H	En_TBCA	R/W	W	Transmit Broadcast Packet Counter
1000_m1D4H	En_TUCA	R/W	W	Transmit Unicast Packet Counter
1000_m1D8H	En_TXPF	R/W	W	Transmit PAUSE control Frame Counter
1000_m1DCH	En_TDFR	R/W	W	Transmit Single Deferral Packet Counter
1000_m1E0H	En_TXDF	R/W	W	Transmit Excessive Deferral Packet Counter
1000_m1E4H	En_TSCL	R/W	W	Transmit Single Collision Packet Counter
1000_m1E8H	En_TMCL	R/W	W	Transmit Multiple collision Packet Counter
1000_m1ECH	En_TLCL	R/W	W	Transmit Late Collision Packet Counter
1000_m1F0H	En_TXCL	R/W	W	Transmit Excessive Collision Packet Counter
1000_m1F4H	En_TNCL	R/W	W	Transmit Total Collision Counter
1000_m1F8H	En_TCSE	R/W	W	Transmit Carrier Sense Error Counter This counter is incremented for each frame transmitted which carrier sense error occurs during transmission.
1000_m1FCH	En_TIME	R/W	W	Transmit Internal MAC Error Counter This counter is incremented for each frame transmitted in which an internal MAC error occurs during transmission.

Remarks 1. In the “Offset Address” field and in the “Register Name” field,
Ethernet Controller #1: m = 2, n = 1,
Ethernet Controller #2: m = 3, n = 2

2. In the “R/W” field,
“W” means “writeable”,
“R” means “readable”,
“RC” means “read-cleared”,
“- “ means “not accessible”.

3. All internal registers are 32-bit word-aligned registers.

4. The burst access to the internal register is prohibited.
If such burst access has been occurred, IRERR bit in NSR is set and NMI will assert to CPU.

5. Read access to the reserved area will set the CBERR bit in the NSR register and the dummy read response data with the data-error bit set on SysCMD [0] is returned.

6. Write access to the reserved area will set the CBERR bit in the NSR register, and the write data is lost.

7. In the “Access” filed,
“W” means that word access is valid,
“H” means that half word access is valid,
“B” means that byte access is valid.

8. Write access to the read-only register cause no error, but the write data is lost.

9. The CPU can access all internal registers, but IBUS master device cannot access them.

5.2.1.3 DMA and FIFO management registers

These registers control to transfer receive and transmit data by internal DMAC of this block.

Table 5-4. DMA and FIFO Management Registers Map

Offset Address	Register Name	R/W	Access	Description
1000_m200H	En_TXCR	R/W	W	Transmit Configuration Register
1000_m204H	En_TXFCR	R/W	W	Transmit FIFO Control Register
1000_m208H: 1000_m210H	N/A	-	-	Reserved for future use
1000_m214H	En_TXDPR	R/W	W	Transmit Descriptor Register
1000_m218H	En_RXCR	R/W	W	Receive Configuration Register
1000_m21CH	En_RXFCR	R/W	W	Receive FIFO Control Register
1000_m220H: 1000_m228H	N/A	-	-	Reserved for future use
1000_m22CH	En_RXDPR	R/W	W	Receive Descriptor Register
1000_m230H	En_RXPDR	R/W	W	Receive Pool Descriptor Register

Remarks 1. In the “Offset Address” field and in the “Register Name” field,
Ethernet Controller #1: m = 2, n = 1,
Ethernet Controller #2: m = 3, n = 2

2. In the “R/W” field,
“W” means “writeable”,
“R” means “readable”,
“RC” means “read-cleared”,
“- “ means “not accessible”.

3. All internal registers are 32-bit word-aligned registers.

4. The burst access to the internal register is prohibited.
If such burst access has been occurred, IRERR bit in NSR is set and NMI will assert to CPU.

5. Read access to the reserved area will set the CBERR bit in the NSR register and the dummy read response data with the data-error bit set on SysCMD [0] is returned.

6. Write access to the reserved area will set the CBERR bit in the NSR register, and the write data is lost.

7. In the “Access” filed,
“W” means that word access is valid,
“H” means that half word access is valid,
“B” means that byte access is valid.

8. Write access to the read-only register cause no error, but the write data is lost.

9. The CPU can access all internal registers, but IBUS master device cannot access them.

5.2.1.4 Interrupt and configuration registers

These register control interrupt occur and configuration for this block.

Table 5-5. Interrupt and Configuration Registers Map

Offset Address	Register Name	R/W	Access	Description
1000_m234H	En_CCR	R/W	W	Configuration Register
1000_m238H	En_ISR	RC	W	Interrupt Service Register
1000_m23CH	En_MSR	R/W	W	Mask Serves Register

Remarks 1. In the “Offset Address” field and in the “Register Name” field,
Ethernet Controller #1: m = 2, n = 1,
Ethernet Controller #2: m = 3, n = 2

2. In the “R/W” field,
“W” means “writeable”,
“R” means “readable”,
“RC” means “read-cleared”,
“- “ means “not accessible”.

3. All internal registers are 32-bit word-aligned registers.

4. The burst access to the internal register is prohibited.
If such burst access has been occurred, IRERR bit in NSR is set and NMI will assert to CPU.

5. Read access to the reserved area will set the CBERR bit in the NSR register and the dummy read response data with the data-error bit set on SysCMD [0] is returned.

6. Write access to the reserved area will set the CBERR bit in the NSR register, and the write data is lost.

7. In the “Access” filed,
“W” means that word access is valid,
“H” means that half word access is valid,
“B” means that byte access is valid.

8. Write access to the read-only register cause no error, but the write data is lost.

9. The CPU can access all internal registers, but IBUS master device cannot access them.

5.2.2 En_MACC1 (MAC Configuration Register 1)

Bits	Field	R/W	Default	Description
31:12	Reserved	R/W	0	Reserved for future use. Write 0s.
11	TXFC	R/W	0	Transmit flow control enable: Setting this bit to a ‘1’ enables to transmit the pause control frame.
10	RXFC	R/W	0	Receive flow control enable: Setting this bit to a ‘1’ enables MAC Control Block to execute PAUSE operation for the pause time on the setting of the pause timer. The setting of the pause timer must be updated every time a valid PAUSE control frame is received regardless of the setting of this bit.
9	SRXEN	R/W	0	Receive enable: Setting this bit to a ‘1’ enables the function of the MAC Address field filtering.
8	PARF	R/W	0	Control packet pass: Setting this bit to a ‘1’ allows MAC Control Block to check for all received packets including control frames. When this bit is set to a ‘0’, MAC control block does not check the reception of a control frame.
7	PUREP	R/W	0	Pure preamble: Setting this bit to a ‘1’ allows the recognition of start of a new packet only when a pure preamble (“0101” only) is captured.
6	FLCHT	R/W	0	Length field check: When this bit is set to a ‘1’, MAC Control Block compares the length field value in the packet to the actual packet length, and indicates the result in the status vector. When this bit is set 0, MAC does not check the Frame length field.
5	NOBO	R/W	0	No Back Off: When this bit is set to a ‘1’, MAC Control Block always transmits the packet with no back off operation.
4	Reserved	-	-	Reserved for future use. Write a 0.
3	CRCEN	R/W	0	CRC append enable: Setting this bit to a ‘1’ enables MAC to append calculated CRC code to the end of transmitted packet automatically.
2	PADEN	R/W	0	PAD append enable: Setting this bit to a ‘1’ enables MAC control block to append PAD when the transmitted packet length is less than 64 octets because the input data (TPD) length is less than 60 bytes before appending the CRC.
1	FULLD	R/W	0	Full duplex enable: Setting this bit to a ‘1’ enables the full duplex operation.
0	HUGEN	R/W	0	Large packet enable: Setting this bit to a ‘1’ enables MAC to transmit and receive a packet of which length is over the value of En_LMAX register.

5.2.3 En_MACC2 (MAC Configuration Register 2)

Bits	Field	R/W	Default	Description
31:11	Reserved	R/W	0	Reserved for future use. Write 0s.
10	MCRST	R/W	0	MAC Control Block software reset: Setting this bit to a '1' forces MAC Control Block to a software reset operation. In order to complete the software reset state, this bit needs to be cleared.
9	RFRST	R/W	0	Receive Function Block software reset: Setting this bit to a '1' forces the Receive Function Block to a software reset operation. In order to complete the software reset, this bit needs to be cleared.
8	TFRST	R/W	0	Transmit Function Block software reset: Setting this bit to a '1' forces Transmit Function Block to a software reset operation. In order to complete the software reset, this bit needs to be cleared.
7	Reserved	R/W	0	Reserved for future use. Write a 0.
6	BPNB	R/W	0	Back Pressure No Back Off: When this bit is set to a '1', MAC Control Block transmits the packet with no back off operation after the back pressure operation only.
5	APD	R/W	0	Auto VLAN PAD: When this bit is set to a '1', if the ETPID field in the current transmit packet is equal to the value in the En_VLTP register, MAC pads the current packet as a VLAN packet when the current packet should be padded.
4	VPD	R/W	0	VLAN PAD mode: When this bit is set to a '1', MAC Control Block always pads the current packet as a VLAN packet when the current packet should be padded.
3:0	Reserved	R/W	0	Reserved for future use. Write 0s.

5.2.4 En_IPGT (Back-to-Back IPG Register)

Bits	Field	R/W	Default	Description
31:7	Reserved	R/W	0	Reserved for future use. Write 0s.
6:0	IPGT	R/W	13H	Back-to-Back IPG: This field sets IPG for transmit operation with back-to-back mode. The formula for the Back-to-Back IPG is: IPG = (5 + IPGT) × 4 bits time

5.2.5 En_IPGR (Non Back-to-Back IPG Register)

Bits	Field	R/W	Default	Description
31:15	Reserved	R/W	0	Reserved for future use. Write 0s.
14:8	IPGR1	R/W	0EH	Carrier sense time: This field sets the carrier sense time when the transmit operation is executed in non Back-to-Back mode. The formula for the carrier sense time is: Carrier sense time = (2 + IPGR1) × 4 bits time The carrier sense time defined in this field is included in the non Back-to-Back IPG defined in the IPGR2 field.
7	Reserved	-	-	Reserved for future use. Write a 0.
6:0	IPGR2	R/W	13H	Non Back-To-Back IPG: This field sets IPG for transmit operation with non Back-to-Back mode. The formula for the non Back-to-Back IPG is: IPG = (5 + IPGR2) × 4 bits time

5.2.6 En_CLRT (Collision Register)

Bits	Field	R/W	Default	Description
31:14	Reserved	R/W	0	Reserved for future use. Write 0s.
13:8	LCOL	R/W	38H	Late collision window: This field sets collision window size. The formula for the collision window size is: collision window size = (LCOL + 8) × 8 bits time
7:4	Reserved	R/W	0	Reserved for future use. Write 0s.
3:0	RETRY	R/W	0FH	Maximum number of retry: This field sets the maximum number of retries during the transmission. If a retry occurs beyond this value, the current transmission is aborted.

5.2.7 En_LMAX (Maximum Packet Length Register)

Bits	Field	R/W	Default	Description
31:16	Reserved	R/W	0	Reserved for future use. Write 0s.
15:0	MAXF	R/W	600H	Maximum packet length: This field sets the maximum length of transmit and receive packet by the octet when HUGEN bit in En_MACC1 register is set to a '0'. Receive: If the current receive packet length is greater than the value of MAXF, MAC Control Block terminates the reception. Transmit: If the current transmit packet length is greater than the value of MAXF, MAC Control Block aborts the transmission.

5.2.8 En_RETX (Retry Count Register)

Bits	Field	R/W	Default	Description
31:4	Reserved	R/W	0	Reserved for future use. Write 0s.
3:0	RETX	R/W	0	Retry counter: This field indicates the number of the retries during the current transmission.

5.2.9 En_LSA2 (Station Address Register 2)

Bits	Field	R/W	Default	Description
31:16	Reserved	R/W	0	Reserved for future use. Write 0s.
15:0	LSA2	R/W	0	Station address SA (47:32). Please refer to **5.3.6**.

5.2.10 En_LSA1 (Station Address Register 1)

Bits	Field	R/W	Default	Description
31:0	LSA1	R/W	0	Station address SA (31:0): This field sets the station address. The station address: SA (47:0) is used by the MAC Control Block as the source address in the PAUSE control frame, and used for the comparison with the destination address of a received packet. Please refer to **5.3.6**.

5.2.11 En_PTVR (Pause Timer Value Read Register)

Bits	Field	R/W	Default	Description
31:16	Reserved	R	0	Reserved for future use.
15:0	PTCT	R	0	Pause timer counter: This field indicates the current pause timer value.

5.2.12 En_VLTP (VLAN Type Register)

Bits	Field	R/W	Default	Description
31:16	Reserved	R/W	0	Reserved for future use. Write 0s.
15:0	VLTP	R/W	0	VLAN type: This field sets ETPID value. Receive: MAC Control Block detects a VLAN frame by comparing this field with the ETPID field in a received packet. Transmit: If APD bit in En_MACC2 register is set to a ‘1’ and the ETPID field in the current transmit packet is equal to the value in this field, MAC pads as a VLAN packet when the current packet should be padded.

5.2.13 En_MIIC (MII Configuration Register)

Bits	Field	R/W	Default	Description
31:16	Reserved	R/W	0	Reserved for future use. Write 0s.
15	MIRST	R/W	0	MII Management Interface Block software reset: Setting this bit to a ‘1’ forces the MII Management Interface Block to transit to a software reset operation. In order to complete the software reset, this bit needs to be cleared.
14:4	Reserved	R/W	0	Reserved for future use. Write 0s.
3:2	CLKS	R/W	0	Select frequency range: This field sets the frequency range of the internal clock. MDC is generated by dividing down the internal clock and the clock division is set by these bits. The settings of these bits are: 00: the internal clock is equal to 25 MHz 01: the internal clock is less than or equal to 33 MHz 10: the internal clock is less than or equal to 50 MHz 11: the internal clock is less than or equal to 66 MHz (normal case) MDC is set less than or equal to 2.5MHz. by the setting of this bits.
1:0	Reserved	R/W	0	Reserved for future use. Write 0s.

5.2.14 En_MCMD (MII Command Register)

Bits	Field	R/W	Default	Description
31:2	Reserved	W	0	Reserved for future use. Write 0s.
1	SCANC	W	0	SCAN command: When this bit is set to a ‘1’, MAC Control Block executes the SCAN command.
0	RSTAT	W	0	MII management read: When this bit is set to a ‘1’, MAC Control Block executes the read access through MII management interface.

5.2.15 En_MADR (MII Address Register)

Bits	Field	R/W	Default	Description
31:13	Reserved	R/W	0	Reserved for future use. Write 0s.
12:8	FIAD	R/W	0	MII PHY address: This field sets PHY address to be selected during the management access.
7:5	Reserved	R/W	0	Reserved for future use. Write 0s.
4:0	RGAD	R/W	0	MII register address: This field sets register address to be accessed during the management access.

5.2.16 En_MWTD (MII Write Data Register)

Bits	Field	R/W	Default	Description
31:16	Reserved	R/W	0	Reserved for future use. Write 0s.
15:0	CTLD	R/W	0	MII write data: This field sets the MII management write data for write access through the MII management interface.

5.2.17 En_MRDD (MII Read Data Register)

Bits	Field	R/W	Default	Description
31:16	Reserved	R	0	Reserved for future use.
15:0	CTLD	R	0	MII read data: This field indicates the MII management read data for read access through the MII management interface.

5.2.18 En_MIND (MII Indicate Register)

Bits	Field	R/W	Default	Description
31:3	Reserved	R	0	Reserved for future use.
2	NVALID	R	0	SCAN command start status: This bit is set to a ‘1’ when SCANC bit in En_MCMD register is set to 1. When the first read access using the SCAN command is completed, this bit is cleared.
1	SCANA	R	0	SCAN command active: This bit is set to a ‘1’ while SCANC bit in En_MCMD register is set to 1.
0	BUSY	R	0	BUSY: This bit indicates that the MAC Control Block is executing a management access to a external PHY device through the MII management interface. This bit is set to a ‘1’ during the management access.

5.2.19 En_AFR (Address Filtering Register)

Bits	Field	R/W	Default	Description
31:4	Reserved	R/W	0	Reserved for future use. Write 0s.
3	PRO	R/W	0	Promiscuous mode: When this bit is set to a ‘1’, all receive packets are accepted. Please refer to **5.3.6**.
2	PRM	R/W	0	Accept Multicast: When this bit is set to a ‘1’, all multicast packets are accepted. Please refer to **5.3.6**.
1	AMC	R/W	0	Accept Multicast (qualified): When this bit is set to a ‘1’, multicast packets that match the conditions using hash table are accepted. Please refer to **5.3.6**.
0	ABC	R/W	0	Accept Broadcast: When this bit is set to a ‘1’, all broadcast packets are accepted. Please refer to **5.3.6**.

5.2.20 En_HT1 (Hash Table Register 1)

Bits	Field	R/W	Default	Description
31:0	HT1	R/W	0	Hash table 1: This register is used with the HT2 register as the hash table. It is used for detection of qualified multicast packets. This register sets the upper 32 bits of the hash table, HT (63:32). Please refer to **5.3.6**.

5.2.21 En_HT2 (Hash Table Register 2)

Bits	Field	R/W	Default	Description
31:0	HT2	R/W	0	Hash table 2: This register is used with the HT1 register as the hash table. It is used for detection of qualified multicast packets. This register sets the lower 32 bits of the hash table, HT (31:0). Please refer to **5.3.6**.

5.2.22 En_CAR1 (Carry Register 1)

The bits of this register indicate that an overflow event has occurred in statistics counters. Each bit corresponds to a counter, and the bit is set to a ‘1’ when the corresponding statistics counter overflow event occurs.

Bits	Field	R/W	Default	Description
31:16	Reserved	R/W	0	Reserved for future use. Write 0s.
15	C1VT	R/W	0	En_RVBT counter carry bit
14	C1UT	R/W	0	En_TUCA counter carry bit
13	C1BT	R/W	0	En_TBCA counter carry bit
12	C1MT	R/W	0	En_TMCA counter carry bit
11	C1PT	R/W	0	En_TPCT counter carry bit
10	C1TB	R/W	0	En_TBYT counter carry bit
9	C1MX	R/W	0	En_RMAX counter carry bit
8	C11K	R/W	0	En_R1K counter carry bit
7	C1FE	R/W	0	En_R511 counter carry bit
6	C1TF	R/W	0	En_R255 counter carry bit
5	C1OT	R/W	0	En_R127 counter carry bit
4	C1SF	R/W	0	En_R64 counter carry bit
3	C1BR	R/W	0	En_RBCA counter carry bit
2	C1MR	R/W	0	En_RMCA counter carry bit
1	C1PR	R/W	0	En_RPKT counter carry bit
0	C1RB	R/W	0	En_RBYT counter carry bit

5.2.23 En_CAR2 (Carry Register 2)

The bits of this register indicate that an overflow event has occurred in statistics counters. Each bit corresponds to a counter, and the bit is set to a '1' when the corresponding statistics counter overflow event occurs.

Bits	Field	R/W	Default	Description
31	C2XD	R/W	0	Status vector overrun bit
30:23	Reserved	R/W	0	Reserved for future use. Write 0s.
22	C2IM	R/W	0	En_TIME counter carry bit
21	C2CS	R/W	0	En_TCSE counter carry bit
20	C2BC	R/W	0	En_TNCL counter carry bit
19	C2XC	R/W	0	En_TXCL counter carry bit
18	C2LC	R/W	0	En_TLCL counter carry bit
17	C2MC	R/W	0	En_TMCL counter carry bit
16	C2SC	R/W	0	En_TSCL counter carry bit
15	C2XD	R/W	0	En_TXDF counter carry bit
14	C2DF	R/W	0	En_TDFR counter carry bit
13	C2XF	R/W	0	En_TXPF counter carry bit
12	C2TE	R/W	0	En_TFCS counter carry bit
11	C2JB	R/W	0	En_RJBR counter carry bit
10	C2FG	R/W	0	En_RFRG counter carry bit
9	C2OV	R/W	0	En_ROVR counter carry bit
8	C2UN	R/W	0	En_RUND counter carry bit
7	C2FC	R/W	0	En_RFCR counter carry bit
6	C2CD	R/W	0	En_RCDE counter carry bit
5	C2FO	R/W	0	En_RFLR counter carry bit
4	C2AL	R/W	0	En_RALN counter carry bit
3	C2UO	R/W	0	En_RXUO counter carry bit
2	C2PF	R/W	0	En_RXPF counter carry bit
1	C2CF	R/W	0	En_RXCF counter carry bit
0	C2RE	R/W	0	En_RFCS counter carry bit

5.2.24 En_CAM1 (Carry Register 1 Mask Register)

This register masks the Interrupt that is generated from the setting of the bits in the En_CAR1 register. Each mask bit can be enabled independently.

Bits	Field	R/W	Default	Description
31:16	Reserved	R/W	0	Reserved for future use. Write 0s.
15	M1VT	R/W	0	En_RVBT counter carry mask bit
14	M1UT	R/W	0	En_TUCA counter carry mask bit
13	M1BT	R/W	0	En_TBCA counter carry mask bit
12	M1MT	R/W	0	En_TMCA counter carry mask bit
11	M1PT	R/W	0	En_TPCT counter carry mask bit
10	M1TB	R/W	0	En_TBYT counter carry mask bit
9	M1MX	R/W	0	En_RMAX counter carry mask bit
8	M11K	R/W	0	En_R1K counter carry mask bit
7	M1FE	R/W	0	En_R511 counter carry mask bit
6	M1TF	R/W	0	En_R255 counter carry mask bit
5	M1OT	R/W	0	En_R127 counter carry mask bit
4	M1SF	R/W	0	En_R64 counter carry mask bit
3	M1BR	R/W	0	En_RBCA counter carry mask bit
2	M1MR	R/W	0	En_RMCA counter carry mask bit
1	M1PR	R/W	0	En_RPKT counter carry mask bit
0	M1RB	R/W	0	En_RBYT counter carry mask bit

5.2.25 En_CAM2 (Carry Register 2 Mask Register)

This register masks the Interrupt that is generated from the setting of the bits in the En_CAR2 register.
Each mask bit can be enabled independently.

Bits	Field	R/W	Default	Description
31	M2XD	R/W	0	Status vector overrun mask bit
30:23	Reserved	R/W	0	Reserved for future use. Write 0s.
22	M2IM	R/W	0	En_TIME counter carry mask bit
21	M2CS	R/W	0	En_TCSE counter carry mask bit
20	M2BC	R/W	0	En_TNCL counter carry mask bit
19	M2XC	R/W	0	En_TXCL counter carry mask bit
18	M2LC	R/W	0	En_TLCL counter carry mask bit
17	M2MC	R/W	0	En_TMCL counter carry mask bit
16	M2SC	R/W	0	En_TSCL counter carry mask bit
15	M2XD	R/W	0	En_TXDF counter carry mask bit
14	M2DF	R/W	0	En_TDFR counter carry mask bit
13	M2XF	R/W	0	En_TXPF counter carry mask bit
12	M2TE	R/W	0	En_TFCS counter carry mask bit
11	M2JB	R/W	0	En_RJBR counter carry mask bit
10	M2FG	R/W	0	En_RFRG counter carry mask bit
9	M2OV	R/W	0	En_ROVR counter carry mask bit
8	M2UN	R/W	0	En_RUND counter carry mask bit
7	M2FC	R/W	0	En_RFCR counter carry mask bit
6	M2CD	R/W	0	En_RCDE counter carry mask bit
5	M2FO	R/W	0	En_RFLR counter carry mask bit
4	M2AL	R/W	0	En_RALN counter carry mask bit
3	M2UO	R/W	0	En_RXUO counter carry mask bit
2	M2PF	R/W	0	En_RXPF counter carry mask bit
1	M2CF	R/W	0	En_RXCF counter carry mask bit
0	M2RE	R/W	0	En_RFCS counter carry mask bit

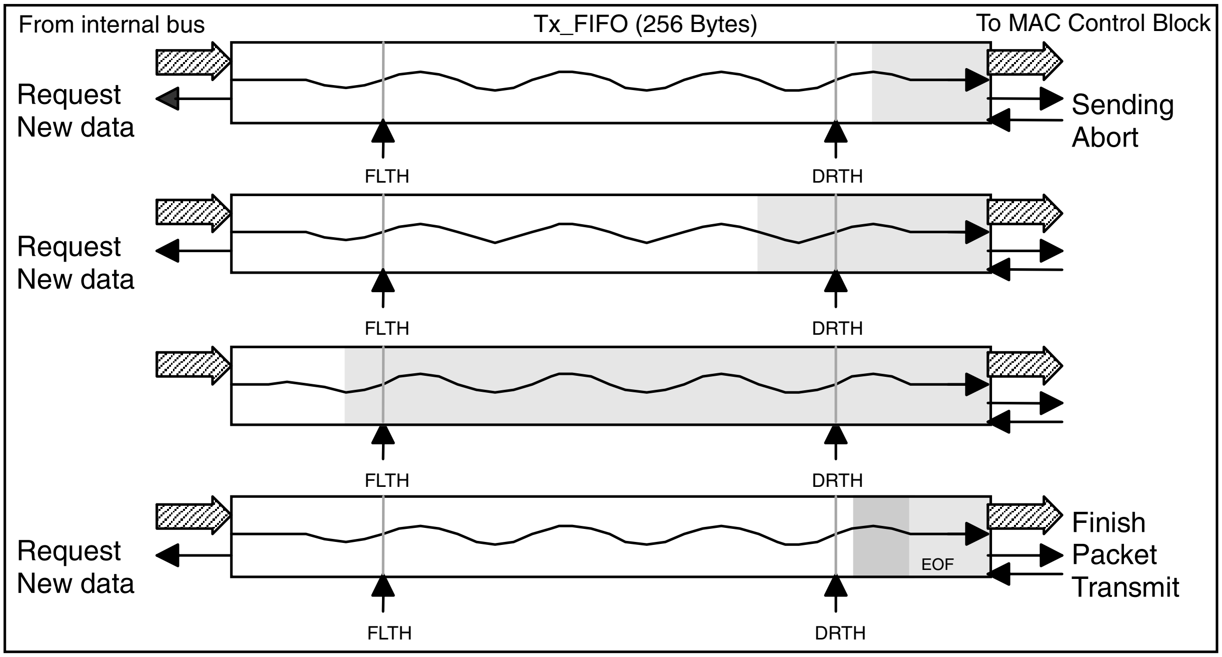
5.2.26 En_TXCR (Transmit Configuration Register)

Bits	Field	R/W	Default	Description
31	TXE	R/W	0	Transmit Enable: 0: Disable 1: Enable
30:19	Reserved	R/W	0	Reserved for future use. Write 0s.
18:16	DTBS [2:0]	R/W	0	DMA Transmit Burst Size: 000: 1 Word (4 bytes) 001: 2 Words (8 bytes) 010: 4 Words (16 bytes) 011: 8 Words (32 bytes) 100: 16 Words (64 bytes) 101: 32 Words (128 bytes) 110: 64 Words (256 bytes) 111: Reserved for future use
15:1	Reserved	R/W	0	Reserved for future use. Write 0s.
0	AFCE	R/W	0	Auto Flow Control Enable: 0: Disable 1: Enable

5.2.27 En_TXFCR (Transmit FIFO Control Register)

Bits	Field	R/W	Default	Description
31:16	TPTV	R/W	FFFFH	Transmit Pause Timer Value:
15:10	TX_DRTH	R/W	10H	Transmit Drain Threshold Level: This threshold is enable to the transmit data to the MAC Control Block form the Tx-FIFO. If the transfer data is not completed by DMAC and the buffer empty pointer exceed this pointer, this MAC Control Block sends an Abort Packet. Please see the **Figure 5-2**. This is a word pointer.
9:8	Reserved	R/W	0	Reserved for future use. Write 0s.
7:2	TX_FRTH	R/W	30H	Transmit Fill Threshold Level: This threshold is enable to transmit data to the FIFO from the internal bus through the DMAC of this block. Please see the **Figure 5-2**. This is a word pointer.
1:0	Reserved	R/W	0	Reserved for future use. Write 0s.

Figure 5-2. Tx FIFO Control Mechanism



5.2.28 En_TXDPR (Transmit Descriptor Pointer)

Bits	Field	R/W	Default	Description
31:2	XMTDP	R/W	0	Transmit Descriptor Please see the Section **5.3.4**
1:0	Reserved	R/W	0	Reserved for future use. Write 0s.

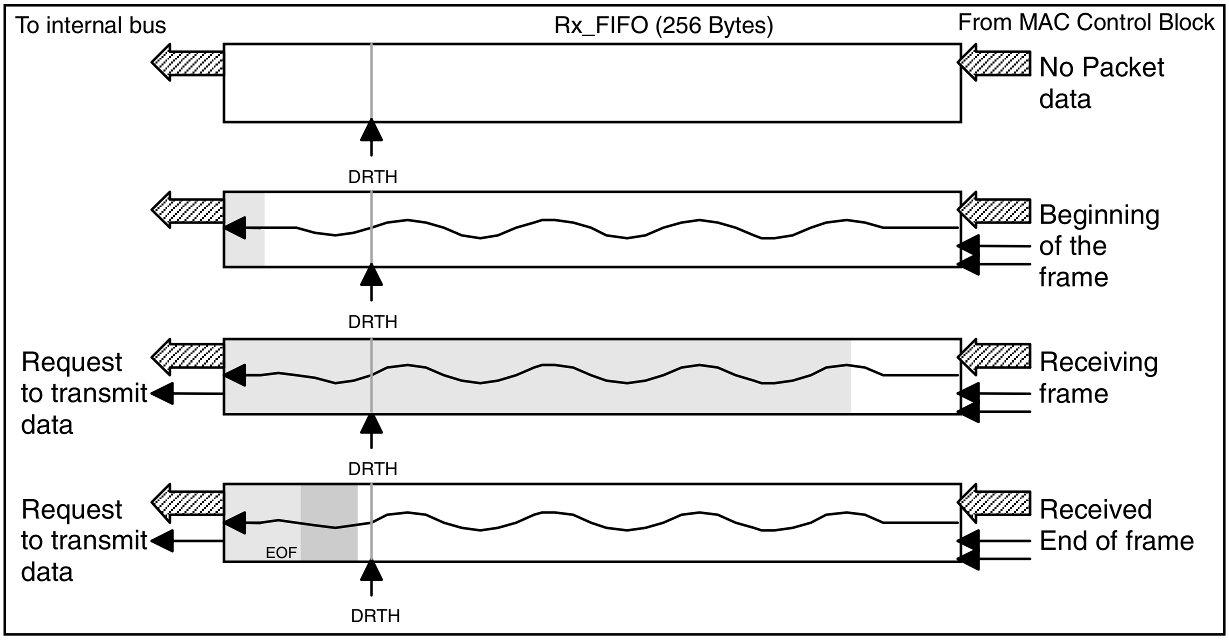
5.2.29 En_RXCR (Receive Configuration Register)

Bits	Field	R/W	Default	Description
31	RXE	R/W	0	Receive Enable: 0: Disable 1: Enable
30:19	Reserved	R/W	0	Reserved for future use. Write 0s.
18:16	DRBS [2:0]	R/W	0	DMA Receive Burst Size: 000: 1 Word (4 bytes) 001: 2 Word (8 bytes) 010: 4 Word (16 bytes) 011: 8 Word (32 bytes) 100: 16 Word (64 bytes) 101: 32 Word (128 bytes) 110: 64 Word (256 bytes) 111: Reserved for future use
15:0	Reserved	R/W	0	Reserved for future use. Write 0s.

5.2.30 En_RXFCR (Receive FIFO Control Register)

Bits	Field	R/W	Default	Description
31:26	UWM [7:2]	R/W	30H	Upper Water Mark: This pointer is used with Auto Flow Control Enable bit in En_TXCR. When the receiving data fill level exceeds this pointer, the transmit module generates a flow control frame automatically.
25:24	Reserved	R/W	0	Reserved for future use. Write 0s.
23:18	LWN [7:2]	R/W	10H	Lower Water Mark:
17:8	Reserved	R/W	0	Reserved for future use. Write 0s.
7:2	RX_DRTH	R/W	10H	Receive Drain Threshold Level This threshold is enable to the transmit data to the IBUS via internal DMAC form the FIFO. Please see the **Figure 5-3**. This pointer is a word pointer.
1:0	Reserved	R/W	0	Reserved for future use. Write 0s.

Figure 5-3. Rx FIFO Control Mechanism



5.2.31 En_RXDPR (Receive Descriptor Pointer)

Bits	Field	R/W	Default	Description
31:2	RCVDP	R/W	0	Receive Descriptor Pointer: Please see the Section **5.3.5**.
1:0	Reserved	R/W	0	Reserved for future use. Write 0s.

5.2.32 En_RXPDR (Receive Pool Descriptor Pointer)

Bits	Field	R/W	Default	Description
31	Reserved	R/W	0	Reserved for future use. Write a 0.
30:28	AL[2:0]	R/W	0	Alert Level
27:16	Reserved	R/W	0	Reserved for future use. Write 0s.
15:0	RNOD [15:0]	R/W	0	Remaining Number of Descriptor

5.2.33 En_CCR (Configuration Register)

Bits	Field	R/W	Default	Description
31:1	Reserved	R/W	0	Reserved for future use. Write 0s.
0	SRT	R/W	0	Software Reset

5.2.34 En_ISR (Interrupt Serves Register)

Bits	Field	R/W	Default	Description
31:16	Reserved	RC	0	Reserved for future use.
15	XMTDN	RC	0	Transmit Done
14	TBDR	RC	0	Transmit Buffer Descriptor Request at Null
13	TFLE	RC	0	Transmit Frame Length Exceed
12	UR	RC	0	Underrun
11	TABR	RC	0	Transmit Aborted
9:8	Reserved	RC	0	Reserved for future use.
10	TCFRI	RC	0	Control Frame Transmit
7	RCVDN	RC	0	Receive Done
6	RBDRS	RC	0	Receive Buffer Descriptor Request at alert level
5	RBDRU	RC	0	Receive Buffer Descriptor Request at zero
4	OF	RC	0	Overflow
3	LFAL	RC	0	Link Failed
2:1	Reserved	RC	0	Reserved for future use.
0	CARRY	RC	0	Carry Flag: CARRY indicates an overflow of the statistics counters

5.2.35 En_MSR (Mask Serves Register)

Each interrupt source is maskable. En_MSR register shows which interrupts are enable.

Default value is all “0” which means all interrupt sources are disable.

Bits	Field	R/W	Default	Description
31:16	Reserved	R/W	0	Reserved for future use. Write 0s.
15	XMTDN	R/W	0	Transmit Done
14	TBDR	R/W	0	Transmit Buffer Descriptor Request at Null
13	TFLE	R/W	0	Transmit Frame Length Exceed
12	UR	R/W	0	Underrun
11	TABR	R/W	0	Transmit Aborted
9:8	Reserved	R/W	0	Reserved for future use. Write 0s.
10	TCFRI	R/W	0	Control Frame Transmit
7	RCVDN	R/W	0	Receive Done
6	RBDRS	R/W	0	Receive Buffer Descriptor Request at alert level
5	RBDRU	R/W	0	Receive Buffer Descriptor Request at zero
4	OF	R/W	0	Overflow
3	LFAL	R/W	0	Link Failed
2:1	Reserved	R/W	0	Reserved for future use. Write 0s.
0	CARRY	R/W	0	Carry Flag: CARRY indicates an overflow of the statistics counters

5.3 Operation

5.3.1 Initialization

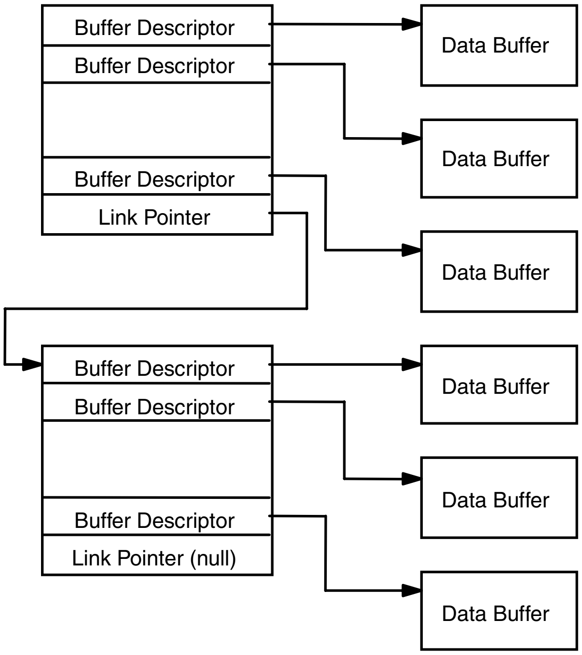
After a power on reset or a software reset, VR4120A has to set the following registers:

i) Interrupt Mask Registers
ii) Configuration Registers
iii) MII Management Registers
iv) Pool/Buffer Descriptor Registers

5.3.2 Buffer structure for Ethernet Controller block

The data buffer structure for Ethernet Controller is shown in **Figure 5-4**.

Figure 5-4. Buffer Structure for Ethernet Block



5.3.3 Buffer descriptor format

The Transmit Descriptor format is shown in **Figure 5-5** and the description is shown in **Table 5-6.**

Figure 5-5. Transmit Descriptor Format

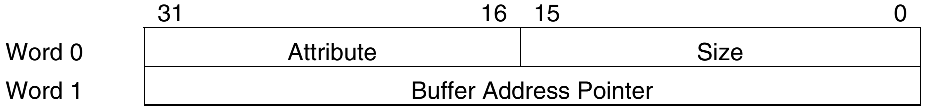


Table 5-6. Attribute for Transmit Descriptor

Attribute & Size	Bit Name	Status
31	L	Last Descriptor
30	D/L	Data Buffer / Link Pointer
29	OWN	Owner 1:Ethernet Controller 0: V$_R$4120A Ethernet Controller sets this bit after it began to transfer data into each descriptor.
28	DBRE	Data Buffer Read Error
27	TUDR	Transmit Underrun Error
26	CSE	Carrier Sense Lost Error
25	LCOL	Late Collision
24	ECOL	Excessive Collision
23	EDFR	Excessive Deferral
22:19	-	Reserved for future use. Write 0s.
18	TGNT	Transmit Giant Frame
17	-	Reserved for future use. Write 0s.
16	TOK	Transmit OK
15:0	SIZE	Transmit Byte Count

The Receive Descriptor format is shown in **Figure 5-6** and the description is shown in **Table 5-7**.

Figure 5-6. Receive Descriptor Format

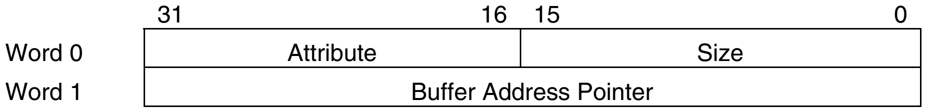


Table 5-7. Attribute for Receive Descriptor

Attribute & Size	Bit Name	Status
31	L	Last Descriptor
30	D/L	Data Buffer / Link Pointer
29	OWN	Owner bit 1:Ethernet Controller 0: VR4120A Ethernet Controller sets this bit after it began to transfer data into each descriptor.
28	DBWE	Data Buffer Write Error
27:25	FTYP	Frame Type[2:0]: 000 Broadcast Frame 001 Multicast Frame 010 Unicast Frame 011 VLAN Frame 100 PAUSE control frame 101 Control Frame (except pause) 11x Reserved for future use
24	OVRN	Overrun Error
23	0	Reserved for future use.
22	0	Reserved for future use.
21	RCV	Detects RXER
20	FC	False Carrier
19	CRCE	CRC Error
18	FAE	Frame Alignment Error
17	RFLE	Receive Frame Length Error
16	RXOK	Receive OK
15:0	SIZE	Receive Byte Count

Remark RUNT packet: less than 64 bytes packet with a good FCS

FRAGMENT packet: less than 64 bytes packet with either a bad FCS or a bad FCS with an alignment error

Dribble Error: When a dribble error is occurred, both of RXOK and FAD will be set.

5.3.4 Frame transmission

The transmitter is designed to work with almost no intervention from the VR4120A. Once the VR4120A enables the transmitter by setting the Transmit Descriptor Pointer Register (En_TXDPR) and the Transmit Enable (TXE), Ethernet Controller fetches the first Transmit Data Buffer from Buffer Descriptor.

When the drain threshold level of the transmit FIFO was over, the MAC Controller Block transmit logic will start transmitting the preamble sequence, the start frame delimiter, and then the frame information. However, the controller defers the transmission if the line is busy (carrier sense is active). Before transmitting, the controller has to wait for carrier sense to become inactive. Once carrier sense is inactive, the controller determines if carrier sense stays inactive for IPGR1 bit time in En_IPGR register. If so, then the transmission begins after waiting an additional IPGR2 – IPGR1 bit times (i.e., IPG is generally 96 bit times).

If a collision occurs during the transmit frame, Ethernet Controller follows the specified back-off procedures and attempts to re-transmit the frame until the retry limit threshold is reached (RETRY in En_CLRT register). Ethernet Controller holds the first 64 bytes of the transmit frame in the transmit FIFO, so that Ethernet Controller does not have to be retrieved from system memory in case of a collision. This improves bus utilization and latency.

When Ethernet Controller reads the Transmit Buffer Descriptor, and it shows the end of data buffer “L bit is set to a ‘1’, Ethernet Controller adds the FCS after the end of data if CRCEN in En_MACC1 register is enable.

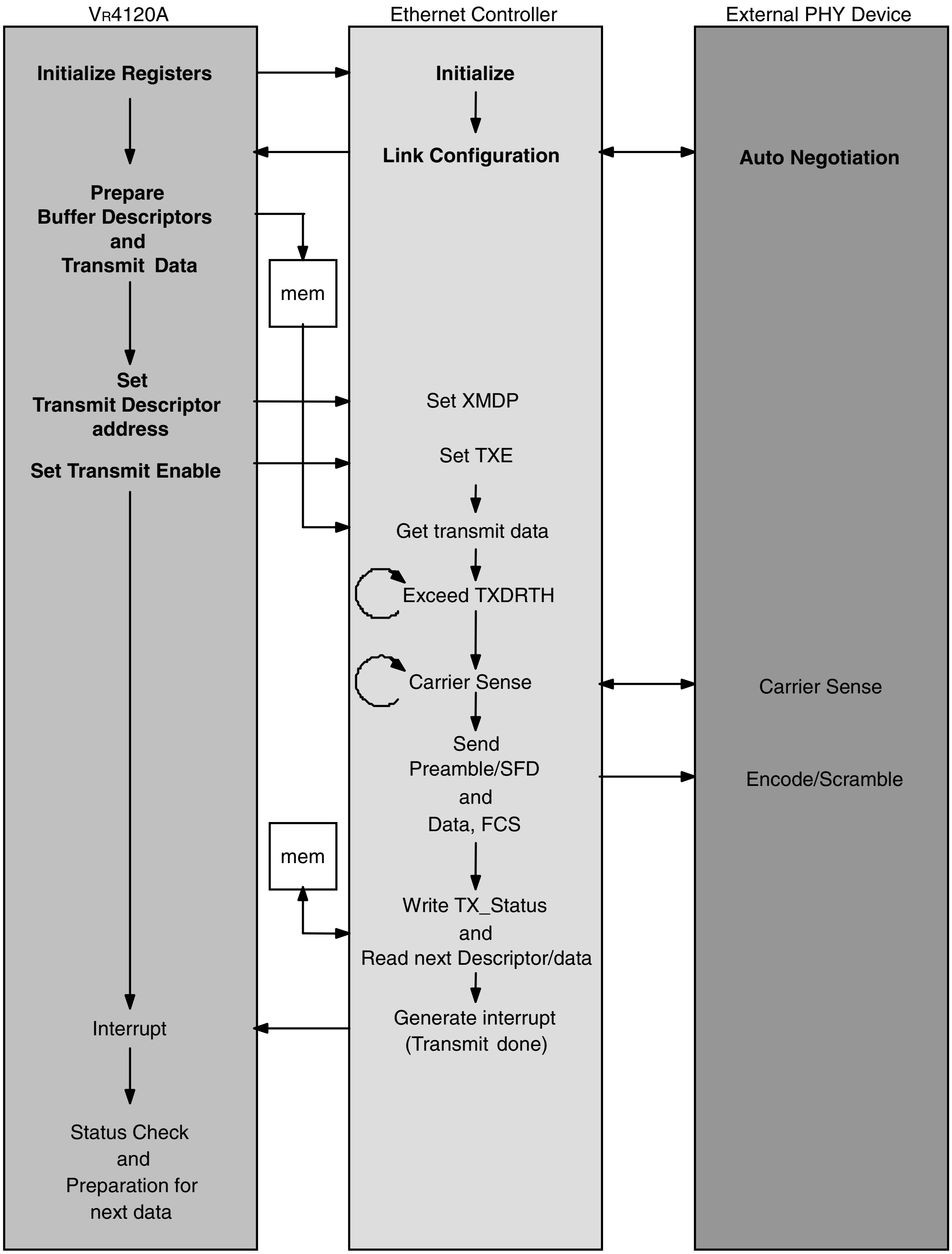
Short frames are automatically padded by the transmit logic if PADEN bit in En_MACC1 register is set. If the transmit frame length exceeds 1518 bytes, Ethernet Controller will assert an interrupt. However, the entire frame will be transmitted (no truncation).

If the current descriptor does not contain the end of frame, Ethernet Controller reads next buffer descriptor, and then reads the continuous data from the data buffer. After Ethernet Controller sent out the whole packet,. Ethernet Controller writes the transmission status into the last descriptor (L=1), and generates an interrupt to indicates the end of transmission. After this, Ethernet Controller fetches the next Transmit Buffer Descriptor, and then if the next data is available, it will be sent out in the same manner.

When Ethernet Controller received the pause control frame and if it is active, Ethernet Controller transmitter stops immediately if no transmission is in progress or continues transmission until the current frame either finishes or terminates with a collision. When the pause timer was expired or Ethernet Controller received a zero value of pause control frame, Ethernet Controller resumes transmission with the next frame.

Transmit procedure is as follows: (**Figure 5-7**)

Figure 5-7. Transmit Procedure



Operation flow for transmit packet

i) Prepares transmit data in data buffer
ii) Initializes registers (XMDP, TXE)
iii) Reads buffer descriptor for transmission from SDRAM
iv) Reads transmit data from data buffer by using master DMA burst operation
v) Waits for exceeding of transmit drain threshold (TXDRTH)
Senses carrier
Transmits data (Preamble. SFD, data)
vi) Reads continuous data?
If the current buffer descriptor does not show a last packet (L=0), it reads continuous data.
Increments current Transmit Descriptor Pointer
vii) Reads next buffer descriptor
viii) Reads continuous data from data buffer again
ix) Stores the transmit status in the last buffer descriptor (L = 1)
x) Generates an interrupt
xi) Reads next buffer descriptor and data, if available

Remark When a transmit abort, like an underrun or an excessive collision occurs, the XMTDP has to be set again after checking the status in the buffer descriptor.

5.3.5 Frame reception

The receiver is designed to work with almost no intervention from the host processor and can perform address recognition, CRC checking and maximum frame length checking.

When the driver enables the receiver by setting Receive Descriptor Pointer Register (En_RXDPR) and Receive Enable (RXE), it will immediately start processing receive frames. The receiver will first check for a valid preamble (PA)/start frame delimiter (SFD) header at the beginning packet. If the PA/SFD is valid, it will be stripped and the frame will be processed by the receiver. If a valid PA/SFD is not found the frame will be ignored.

Once a collision window (64 bytes) of data has been received and if address recognition has not rejected the frame, Ethernet Controller starts transferring the incoming frame to the receive data buffer. If the frame is a runt (due to collision) or is rejected by address recognition, no receive buffers are filled. Thus, no collision frames are presented to the user except late collisions, which indicate serious LAN problems.

It has no matter since after the reception it writes the receive status into the descriptor even if the received data were gone out to SDRAM.

If the incoming frame exceeds the length of the data buffer, Ethernet Controller fetches the next Receive Descriptor Buffer in the table and, if it is empty, continues transferring the rest of the frame to this data buffer.
If the remaining number of descriptors is under four times of the alert level, Ethernet Controller generates an interrupt to request new additional descriptors.

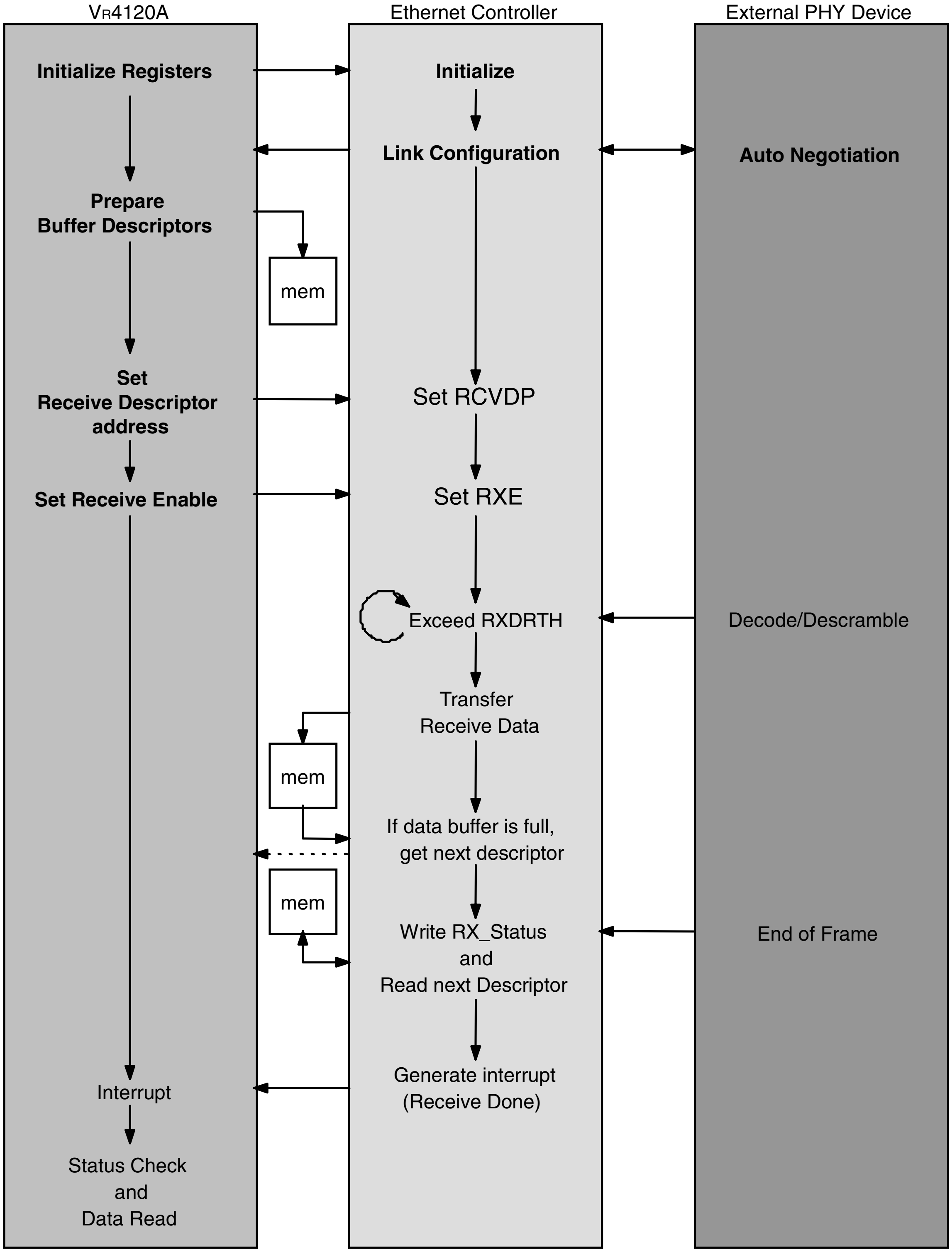
During reception, Ethernet Controller checks for a frame that is either too short or too long. When the frame ends (carrier sense is negated), the receive CRC field is checked out and written to the data buffer. The data length written to the last data Buffer in the Ethernet frame is the length of the entire frame. Frames that are less then 64 bytes in length are not DMA’d (transferred) and, are rejected in hardware with no impact on system bus utilization if the data is in the Rx FIFO.

Caution Recommend a high (over 16 words) drain threshold.

When the receive frame is complete, Ethernet Controller sets the L-bit in the Receive Descriptor, writes the frame status bits into the Receive Descriptor, and sets the OWN-bit. Ethernet Controller generates a maskable interrupt, indicating that a frame has been received and is in memory. Ethernet Controller then waits for a new frame.

Receive procedure is as follows: (**Figure 5-8**)

Figure 5-8. Receive Procedure



Operation flow for receive packet

i) Prepares the receive buffer descriptors
ii) Initializes registers (RXVDP, RXE)
iii) Reads the receive buffer descriptor
iv) Waits for exceeding of receive drain threshold (RXDRTH)
v) Writes receive data to data buffer by using master DMA burst operation
vi) Increments the Receive Descriptor Pointer if the current data buffer is full
vii) Check out the RNOD
If the remaining number of descriptors is less than four times of the alert level, generates an interrupt to request an adding descriptor.
viii) Reads the next buffer descriptor
ix) Stores the received data
x) Stores the receive status in the last buffer descriptor (L = 1)
xi) Generates an interrupt for the end of reception
xii) Reads next receive descriptor if available

How to add the receive buffer descriptors

i) Prepares the receive buffer descriptors
ii) Sets the number of buffer descriptors in En_RXPDR as well as the alert level
iii) Generates an interrupt by this Ethernet Controller
iv) Adds the receive buffer descriptors in the memory
v) Sets the number of buffer descriptors in En_RXPDR as well as the alert level

5.3.6 Address Filtering

The Ethernet Controller can parse a destination address in a received packet. The destination address is filtered using the condition in En_AFR register set by VR4120. The condition for unicast, multicast and broadcast can be configured independently.

(1) Unicast address filtering

The destination address in a received packet is compared with the station address in En_LSA1 and En_LSA2 registers. When both the addresses are equal, the received packet is accepted. The comparison is executed for every received packet.

(2) Multicast address filtering

Two filtering methods are supported. With one method, all of received multicast packets are accepted when PRM bit in En_AFR register is set to a ‘1’.

With the other method, received multicast packets are filtered, using a hash table configured by the values in En_HT1 and En_HT2 registers. At first, the CRC is executed against the multicast destination address in the received packet using following polynomial expression.

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X + 1$$

Bits [28:23] of the CRC calculation result are decoded. When the bits in En_HT1 and En_HT2 registers pointed by the decoded result are equal to ‘1’, the received packet with the multicast destination address is accepted. In order to set the En_HT1 and En_HT2 registers, CRCs for multicast destination address to be received should be calculated before receiving any multicast packet.

(3) Broadcast address filtering

All of received packets with broadcast destination address are received when ABC bit in En_AFR register is set to a ‘1’.

(4) Promiscuous mode

Setting PRO bit in En_AFR register to a ‘1’ caused all of received packets to be received.

Filtering procedure is as follows:

At first, SRXEN bit in En_MACC1 register is set to a ‘1’. In this case, the received data interface is disabled. Then, the station address is set in En_LSA1 and En_LSA2 registers. En_AFR register is also set for enabling of unicast, multicast and broadcast reception. In addition, En_HT1 and En_HT2 registers should be set when multicast address filtering with the hash table is used. After these procedures, SRXEN is set to a ‘1’ in order to enable the received data interface.

CHAPTER 6 USB CONTROLLER

6.1 Overview

The USB Controller handles the data communication through USB. The following lists the features of USB Controller.

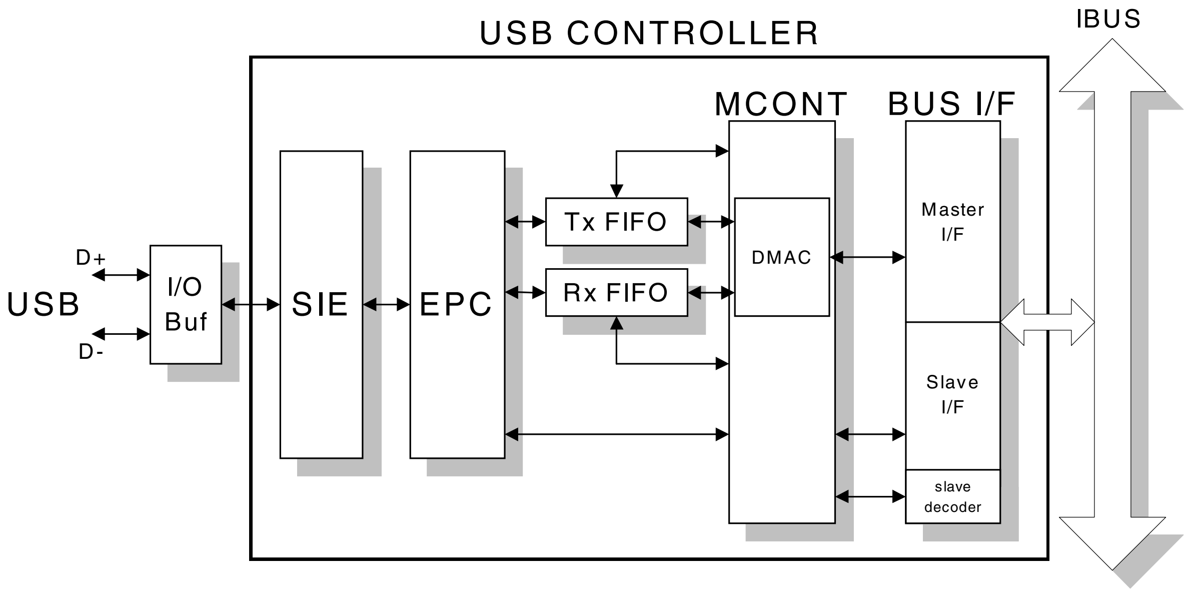
6.1.1 Features

- Conforms to Universal Serial Bus Specification Rev 1.1
- Supports operation conforming to the USB Communication Device Class Specification
- Supports data transfer at full speed (12 Mbps)
- In addition to the control Endpoint, a further six Endpoints are built in
  (Interrupt in/out, Isochronous in/out and Bulk in/out)
- Supports a built-in 64-byte Tx FIFO for Control transfer
- Supports a built-in 128-byte Tx FIFO for Isochronous transfer
- Supports a built-in 128-byte Tx FIFO for Bulk transfer
- Supports a built-in 64-byte Tx FIFO for Interrupt transfer
- Supports a built-in 128-byte shared Rx FIFO for Control/Isochronous/Bulk/Interrupt transfer
- Supports a DMA function for transferring transmit/receive data
- Supports Control/Status registers
- Compatible with the Suspend and Resume signaling issued from the Host PC
  (Processing by the V_R4120A is required)
- Supports Remote Wake-up (Processing by the V_R4120A is required)
- Supports Direct connect to Internal BUS (IBUS) Master and Slave Interface block
- Supports the counters required to indicate the USB status

6.1.2 Internal block diagram

USB Controller internal block diagram is as shown below.

Figure 6-1. USB Controller Internal Configuration



USB Controller's configuration features the following blocks.

SIE (Serial Interface Engine): Performs Serial/Parallel conversion, NRZI encoding/decoding, CRC calculation, etc.
EPC (EndPoints Controller): Performs data transmission/reception for each Endpoint.
Tx FIFO (Transmit FIFO): FIFO for transmitting data
Rx FIFO (Receive FIFO): FIFO for receiving data
MCONT (Main Controller): Block for controlling transmission and reception.
DMAC (DMA Controller): Block for controlling DMA transfer.
Master_if (Master Interface): Master section of the Internal BUS interface.
Slave_if (Slave Interface): Slave section of the Internal BUS interface.
I/O Buf (I/O Buffer): I/O buffer that satisfies the electrical specifications of the USB.
Internal BUS: The internal bus of the μPD98502.

6.2 Registers

This section explains the mapping of those registers that can be accessed from IBUS. USB base address is 1000_1000H

6.2.1 Register map

Offset Address	Register Name	R/W	Access	Description
1000_1000H	U_GMR	R/W	W/H/B	USB General Mode Register
1000_1004H	U_VER	R	W/H/B	USB Frame Number/Version Register
1000_1008H: 1000_100CH	N/A	-	-	Reserved for future use
1000_1010H	U_GSR1	RC	W	USB General Status Register 1
1000_1014H	U_IMR1	R/W	W/H/B	USB Interrupt Mask Register 1
1000_1018H	U_GSR2	RC	W	USB General Status Resister 2
1000_101CH	U_IMR2	R/W	W/H/B	USB Interrupt Mask Register 2
1000_1020H	U_EP0CR	R/W	W/H/B	USB EP0 Control Register
1000_1024H	U_EP1CR	R/W	W/H/B	USB EP1 Control Register
1000_1028H	U_EP2CR	R/W	W/H/B	USB EP2 Control Register
1000_102CH	U_EP3CR	R/W	W/H/B	USB EP3 Control Register
1000_1030H	U_EP4CR	R/W	W/H/B	USB EP4 Control Register
1000_1034H	U_EP5CR	R/W	W/H/B	USB EP5 Control Register
1000_1038H	U_EP6CR	R/W	W/H/B	USB EP6 Control Register
1000_103CH	N/A	-	-	Reserved for future use
1000_1040H	U_CMR	R/W	W	USB Command Register
1000_1044H	U_CA	R/W	W/H/B	USB Command Address Register
1000_1048H	U_TEPSR	R	W/H/B	USB Tx EndPoint Status Register
1000_104CH	N/A	-	-	Reserved for future use
1000_1050H	U_RP0IR	R/W	W/H/B	USB Rx Pool0 Information Register
1000_1054H	U_RP0AR	R	W/H/B	USB Rx Pool0 Address Register
1000_1058H	U_RP1IR	R/W	W/H/B	USB Rx Pool1 Information Register
1000_105CH	U_RP1AR	R	W/H/B	USB Rx Pool1 Address Register
1000_1060H	U_RP2IR	R/W	W/H/B	USB Rx Pool2 Information Register
1000_1064H	U_RP2AR	R	W/H/B	USB Rx Pool2 Address Register
1000_1068H: 1000_106CH	N/A	-	-	Reserved for future use
1000_1070H	U_TMSA	R/W	W/H/B	USB Tx MailBox Start Address Register
1000_1074H	U_TMBA	R/W	W/H/B	USB Tx MailBox Bottom Address Register
1000_1078H	U_TMRA	R/W	W/H/B	USB Tx MailBox Read Address Register
1000_107CH	U_TMWA	R	W/H/B	USB Tx MailBox Write Address Register
1000_1080H	U_RMSA	R/W	W/H/B	USB Rx MailBox Start Address Register
1000_1084H	U_RMBA	R/W	W/H/B	USB Rx MailBox Bottom Address Register
1000_1088H	U_RMRA	R/W	W/H/B	USB Rx MailBox Read Address Register
1000_108CH	U_RMWA	R	W/H/B	USB Rx MailBox Write Address Register
1000_1090H: 1000_1FFCH	N/A	-	-	Reserved for future use

Remarks 1. In the “R/W” field,
“W” means “writeable”,
“R” means “readable”,
“RC” means “read-cleared”,
“- “ means “not accessible”.

2. All internal registers are 32-bit word-aligned registers.
3. The burst access to the internal register is prohibited.
   If such burst access has been occurred, IRERR bit in NSR is set and NMI will assert to CPU.
4. Read access to the reserved area will set the CBERR bit in the NSR register and the dummy read response data with the data-error bit set on SysCMD [0] is returned.
5. Write access to the reserved area will set the CBERR bit in the NSR register, and the write data is lost.
6. In the "Access" filed,
   "W" means that word access is valid,
   "H" means that half word access is valid,
   "B" means that byte access is valid.
7. Write access to the read-only register cause no error, but the write data is lost.
8. The CPU can access all internal registers, but IBUS master device cannot access them.

6.2.2 U_GMR (USB General Mode Register)

This register is used for setting the operation of USB Controller. The low-order sixteen bits except for RR bit can be written only when the device is being initialized. If the values of these bits are changed while transmission or reception is being performed, the operation of USB Controller may become unpredictable.

Bits	Field	R/W	Default	Description
31:24	Reserved	R/W	0	Reserved for future use. Writes '0's.
23	VT	R/W	0	Function Address Valid Timing: When this bit is set to a '1', FA becomes valid immediately. When this bit is set to a '0', FA will become valid after USB Controller receives subsequent ACK packet on EndPoint0.
22:16	FA	R/W	0	Function Address: Register that stores the USB Function Address. This is allocated by the Host PC as part of the USB configuration process. The VR4120A should set the allocated address in this register.
15:8	SOFINTVL	R/W	18H	SOF Interval: This value is used to define the allowable skew for SOF packet. The default value should be 18H. When '00H' is set, the USB Controller does not care the timing between two consecutive SOF packets.
7:3	Reserved	R/W	0	Reserved for future use. Writes '0's.
2	AU	R/W	0	Auto Update: Frame Number auto updating enable. To set to a '1' causes Frame Number Register to be updated though a received SOF packet is corrupted.
1	LE	R/W	0	Loopback Enable: Bit for enabling internal loopback mode. When this bit is set to a '1', USB Controller operates in loopback mode. Setting loopback mode enables the testing of the internal DMA controller. In addition, USB packets are not transmitted, and USB packets are not received. For a detailed explanation of loopback mode, see Section **6.9**.
0	RR	R/W	0	Remote Resume: When Remote Resume is to be performed, the VR4120A will set this bit. Once this bit is set to a '1', USB Controller issues Resume Signaling to the USB for a period of 5 ms. Upon the completion of Resume Signaling, this bit is automatically reset to a '0'.

6.2.3 U_VER (USB Frame Number/Version Register)

Register that stores the current Frame Number of the USB and version of the USB Controller block.

Bits	Field	R/W	Default	Description
31:16	UVER	R	0201H	USB Version: Hardwired to '0201H'. The Revision Number of the USB Controller block is stored into this register.
15:11	Reserved	R	0	Reserved for future use
10:0	UFNR	R	0	USB Frame Number: Register that stores the Frame Number of the USB.

6.2.4 U_GSR1 (USB General Status Register 1)

This register indicates the current status of USB Controller.

Bits	Field	R/W	Default	Description
31	GSR2	RC	0	If some bits of General Status Register 2 are set to ‘1’s and the corresponding bits in Interrupt Mask Register 2 are set to ‘1’s, this GSR2 bit will be set to a ‘1’.
30:24	Reserved	R	0	Reserved for future use
23	TMF	RC	0	Tx MailBox Full: Bit that indicates transmit MailBox area is full. This bit is set to a ‘1’ when the USB Tx MailBox Read Address and the USB Tx MailBox Write Address get equal. This bit is reset to a ‘0’ when the VR4120A reads this register.
22	RMF	RC	0	Rx MailBox Full: Bit that indicates receive MailBox area is full. This bit is set to a ‘1’ when the USB Rx MailBox Read Address and the USB Rx MailBox Write Address get equal. This bit is reset to a ‘0’ when the VR4120A reads this register.
21	RPE2	RC	0	Rx Pool2 Empty: Bit that indicates receive Pool2 is empty. This bit is reset to a ‘0’ when the VR4120A reads this register.
20	RPE1	RC	0	Rx Pool1 Empty: Bit that indicates receive Pool1 is empty. This bit is reset to a ‘0’ when the VR4120A reads this register.
19	RPE0	RC	0	Rx Pool0 Empty: Bit that indicates receive Pool0 is empty. This bit is reset to a ‘0’ when the VR4120A reads this register.
18	RPA2	RC	0	Rx Pool2 Alert: This bit is set to a ‘1’ when the number of Buffer Directories remaining in receive Pool2 gets equal to 4 times of the AL field value in the Rx Pool2 Information Register. This bit is reset to a ‘0’ when the VR4120A reads this register.
17	RPA1	RC	0	Rx Pool1 Alert: This bit is set to a ‘1’ when the number of Buffer Directories remaining in receive Pool1 gets equal to 4 times of the AL field value in the Rx Pool1 Information Register. This bit is reset to a ‘0’ when the VR4120A reads this register.
16	RPA0	RC	0	Rx Pool0 Alert: This bit is set to a ‘1’ when the number of Buffer Directories remaining in receive Pool0 gets equal to 4 times of the AL field value in the Rx Pool0 Information Register. This bit is reset to a ‘0’ when the VR4120A reads this register.
15:11	Reserved	R	0	Reserved for future use
10	DER	RC	0	DMA Error: Bit that indicates that an error occurred during DMA transfer. This bit is set to a ‘1’ if an error occurs on the Internal BUS during DMA transfer. This bit is reset to a ‘0’ when the VR4120A reads this register.
9	EP2FO	RC	0	EP2 FIFO Error: Bit that indicates that an overrun has occurred for the FIFO of EndPoint2 (Isochronous OUT). When the FIFO becomes full while EndPoint2 is performing a transaction, USB Controller can no longer receive data and all data subsequent is discarded. Should this occur, this bit is set to a ‘1’. This bit is reset to a ‘0’ when the VR4120A reads this register.

Bits	Field	R/W	Default	Description
8	EP1FU	RC	0	EP1 FIFO Error: Bit that indicates that an underrun has occurred for the FIFO of EndPoint1 (Isochronous IN). When the FIFO empties while EndPoint1 is performing a transaction, this bit is set to a ‘1’. This bit is reset to a ‘0’ when the VR4120A reads this register.
7	EP6RF	RC	0	EP6 Rx Finished: Bit that indicates that EndPoint6 (Interrupt OUT) has completed the receiving of a data segment and issued the Rx Indication. This bit is reset to a ‘0’ when the VR4120A reads this register.
6	EP5TF	RC	0	EP5 Tx Finished: Bit that indicates that EndPoint5 (Interrupt IN) has completed the transmitting of a data segment and issued the Tx Indication. This bit is reset to a ‘0’ when the VR4120A reads this register.
5	EP4RF	RC	0	EP4 Rx Finished: Bit that indicates that EndPoint4 (Bulk OUT) has completed the receiving of a data segment and issued the Rx Indication. This bit is reset to a ‘0’ when the VR4120A reads this register.
4	EP3TF	RC	0	EP3 Tx Finished: Bit that indicates that EndPoint3 (Bulk IN) has completed the transmitting of a data segment and issued the Tx Indication. This bit is reset to a ‘0’ when the VR4120A reads this register.
3	EP2RF	RC	0	EP2 Rx Finished: Bit that indicates that EndPoint2 (Isochronous OUT) has completed the receiving of a data segment and issued the Rx Indication. The timing when this bit will be set varies with Rx Mode. This bit is reset to a ‘0’ when the VR4120A reads this register.
2	EP1TF	RC	0	EP1 Tx Finished: Bit that indicates that EndPoint1 (Isochronous IN) has completed the transmitting of a data segment and issued the Tx Indication. This bit is reset to a ‘0’ when the VR4120A reads this register.
1	EP0RF	RC	0	EP0 Rx Finished: Bit that indicates that EndPoint0 (Control) has completed the receiving of a data segment and issued the Rx Indication. This bit is reset to a ‘0’ when the VR4120A reads this register.
0	EP0TF	RC	0	EP0 Tx Finished: Bit that indicates that EndPoint0 (Control) has completed the transmitting of a data segment and issued the Tx Indication. This bit is reset to a ‘0’ when the VR4120A reads this register.

6.2.5 U_IMR1 (USB Interrupt Mask Register 1)

This register is used to mask interrupts.

When a bit in this register is set to a ‘1’ and the corresponding bit in the USB General Status Register 1 (Address: 10H) is set to a ‘1’, an interrupt is issued.

Bits	Field	R/W	Default	Description
31	GSR2	R/W	0	General Status Register 2 Interrupt: 1 = unmask. 0 = mask.
30:24	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
23	TMF	R/W	0	Tx MailBox Full: 1 = unmask. 0 = mask.
22	RMF	R/W	0	Rx MailBox Full: 1 = unmask. 0 = mask.
21	RPE2	R/W	0	Rx Pool2 Empty: 1 = unmask. 0 = mask.
20	RPE1	R/W	0	Rx Pool1 Empty: 1 = unmask. 0 = mask.
19	RPE0	R/W	0	Rx Pool0 Empty: 1 = unmask. 0 = mask.
18	RPA2	R/W	0	Rx Pool2 Alert: 1 = unmask. 0 = mask.
17	RPA1	R/W	0	Rx Pool1 Alert: 1 = unmask. 0 = mask.
16	RPA0	R/W	0	Rx Pool0 Alert: 1 = unmask. 0 = mask.
15:11	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
10	DER	R/W	0	DMA Error: 1 = unmask. 0 = mask.
9	EP2FO	R/W	0	EP2 FIFO Error: 1 = unmask. 0 = mask.
8	EP1FU	R/W	0	EP1 FIFO Error: 1 = unmask. 0 = mask.
7	EP6RF	R/W	0	EP6 Rx Finished: 1 = unmask. 0 = mask.
6	EP5TF	R/W	0	EP5 Tx Finished: 1 = unmask. 0 = mask.
5	EP4RF	R/W	0	EP4 Rx Finished: 1 = unmask. 0 = mask.

Bits	Field	R/W	Default	Description
4	EP3TF	R/W	0	EP3 Tx Finished: 1 = unmask. 0 = mask.
3	EP2RF	R/W	0	EP2 Rx Finished: 1 = unmask. 0 = mask.
2	EP1TF	R/W	0	EP1 Tx Finished: 1 = unmask. 0 = mask.
1	EP0RF	R/W	0	EP0 Rx Finished: 1 = unmask. 0 = mask.
0	EP0TF	R/W	0	EP0 Tx Finished: 1 = unmask. 0 = mask.

6.2.6 U_GSR2 (USB General Status Register 2)

This register indicates the current status of USB Controller. Reading this register clears all bits in this register.

Bits	Field	R/W	Default	Description
31:21	Reserved	R	0	Reserved for future use
21	FW	RC	0	Frame Number Written: This bit is set to a ‘1’ when Frame Number is written to USB Frame Number/Version Register (04H).
20	IFN	RC	0	Incorrect Frame Number: This bit is set to a ‘1’ when USB Controller receives a SOF packet with Incorrect Frame Number, or with CRC/Bit Stuff error.
19	IEA	RC	0	Incorrect EndPoint Access: This bit is set to a ‘1’ when USB Controller received an IN or OUT Token with Incorrect EndPoint Number.
18	URSM	RC	0	USB Resume: This bit is set to a ‘1’ when USB Controller has received a Resume Signaling from the Host PC.
17	URST	RC	0	USB Reset: This bit is set to a ‘1’ when USB Controller has received a Reset Signaling from the Host PC.
16	USPD	RC	0	USB Suspend: This bit is set to a ‘1’ when USB Controller detects that USB enters Suspend state.
15:8	Reserved	R	0	Reserved for future use
7	EP2OS	RC	0	Over Size on EndPoint2: This bit is set to a ‘1’ when the received data size is over Max Packet Size on EP2.
6	EP2ED	RC	0	Extra Data on EndPoint2: This bit is set to a ‘1’ when an extra data packet is detected on Isochronous EndPoint (EP2). In the case that EP2EN bit in U_EP2CR is set to a ‘0’, this bit will not be set even if the USB Controller detects an extra data on EP2.
5	EP2ND	RC	0	Isochronous Data Corrupted on EndPoint2: Bit that indicates that Isochronous data is corrupted on EP2. In the case that EP2EN bit in U_EP2CR is set to a ‘0’, this bit will not be set even if the USB Controller detects data corruption on EP2.
4	EP1NT	RC	0	No Token on EndPoint1: This bit is set to a ‘1’ when IN Token packet does not come on EP1 between two SOFs. In the case that EP1EN bit in U_EP1CR is set to a ‘0’, this bit will not be set even if the USB Controller detects no token packet on EP1.
3	EP1ET	RC	0	Extra Token on EndPoint1: This bit is set to a ‘1’ when two IN Token packets are received on EP1 between two SOFs. In the case that EP1EN bit in U_EP1CR is set to a ‘0’, this bit will not be set even if the USB Controller detects an extra token packet on EP1.
2	EP1ND	RC	0	No Data on EndPoint1: This bit is set to a ‘1’ when IN Token packet comes but any data is not ready on EP1. In the case that EP1EN bit in U_EP1CR is set to a ‘0’, this bit will not be set even if the USB Controller detects no data condition on EP1.
1	ES	RC	0	Extra SOF: This bit is set to a ‘1’ when extra SOF packet is detected.
0	SL	RC	0	SOF Loss: This bit is set to a ‘1’ when USB Controller doesn’t receive any SOF packet.

6.2.7 U_IMR2 (USB Interrupt Mask Register 2)

This register is used to mask interrupts.

When a bit in this register is set to a ‘1’ and the corresponding bit in the USB General Status Register 2 (Address: 18H) is set to a ‘1’, GSR2 bit in the U_GSR1 will be set to a ‘1’.

Bits	Field	R/W	Default	Description
31:22	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
21	FW	R/W	0	Frame Number Written: 1 = unmask. 0 = mask.
20	IFN	R/W	0	Incorrect Frame Number: 1 = unmask. 0 = mask.
19	IEA	R/W	0	Incorrect EndPoint Access: 1 = unmask. 0 = mask.
18	URSM	R/W	0	USB Resume: 1 = unmask. 0 = mask.
17	URST	R/W	0	USB Reset: 1 = unmask. 0 = mask.
16	USPD	R/W	0	USB Suspend: 1 = unmask. 0 = mask.
15:8	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
7	EP2OS	R/W	0	Over Size: 1 = unmask. 0 = mask.
6	EP2ED	R/W	0	Extra Data on EndPoint2: 1 = unmask. 0 = mask.
5	EP2ND	R/W	0	Isochronous Data Corrupted: 1 = unmask. 0 = mask.
4	EP1NT	R/W	0	No Token on EndPoint 1: 1 = unmask. 0 = mask.
3	EP1ET	R/W	0	Extra Token on EndPoint 1: 1 = unmask. 0 = mask.
2	EP1ND	R/W	0	No Data on EndPoint 1: 1 = unmask. 0 = mask.
1	ES	R/W	0	Extra SOF: 1 = unmask. 0 = mask.
0	SL	R/W	0	SOF Loss: 1 = unmask. 0 = mask.

6.2.8 U_EP0CR (USB EP0 Control Register)

This register is used for setting the operation of EndPoint0.

If the value in the MAXP field is rewritten during transmitting or receiving operation, the operation of USB Controller may become unpredictable. Therefore, the MAXP can be written only when initial setting is being performed.

Bits	Field	R/W	Default	Description
31	EP0EN	R/W	0	EndPoint Enable: When the VR4120A sets this bit to a ‘1’, EndPoint0 is enabled for transmitting and receiving data to and from USB.
30:19	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
20	ISS	R/W	0	IN Transmit Stall: When the VR4120A sets this bit to a ‘1’, a STALL packet is sent at the Data Phase. Receiving a SETUP packet causes this bit to be a ‘0’.
19	INAK	R/W	0	IN NAK: When the VR4120A sets this bit to a ‘1’, a NAK packet is sent at the Data Phase.
18	OSS	R/W	0	OUT Transmit Stall: When the VR4120A sets this bit to a ‘1’, a STALL handshake is performed at the Handshake Phase. Receiving a SETUP packet causes this bit to be a ‘0’
17	NHSK0	R/W	0	No Handshake: When the VR4120A sets this bit to a ‘1’, No Handshake is performed at the Handshake Phase.
16	ONAK	R/W	0	OUT NAK: When the VR4120A sets this bit to a ‘1,’ NAK Handshake is performed at the Handshake Phase.
15:7	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
6:0	MAXP0	R/W	0	MAX Packet size: The Max Packet Size for EndPoint0. Prior to the start of a USB transaction, the VR4120A must set an appropriate value into this register.

Remark When a STALL handshake is sent by a control endpoint in either the Data or Status stage of a control transfer, a STALL handshake must be returned on all succeeding access to that endpoint until a SETUP PID is received. The endpoint is not required to return a STALL handshake after it receives a subsequent SETUP PID (referred from USB1.1 Specification).

6.2.9 U_EP1CR (USB EP1 Control Register)

This register is used for setting the operation of EndPoint1.

If the value in the MAXP field is rewritten during transmitting operation, the operation of USB Controller may become unpredictable. Therefore, the MAXP can be written only when initial setting is being performed.

Bits	Field	R/W	Default	Description
31	EP1EN	R/W	0	EndPoint Enable: When the VR4120A sets this bit to a ‘1’, EndPoint1 is enabled to transmit data.
30:20	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
19	TM1	R/W	0	Tx Mode: Bit for setting the transmit mode. When this bit is set to a ‘0’, transmitting is performed in SZLP Mode. When this bit is set to a ‘1’, transmitting is performed in NZLP Mode. For a detailed explanation of the transmit modes, see Section **6.5.3**.
18:10	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
9:0	MAXP1	R/W	0	MAX Packet size: The Max Packet Size of EndPoint1. Prior to the start of a USB transaction, the VR4120A must write an appropriate value into this register.

6.2.10 U_EP2CR (USB EP2 Control Register)

This register is used for setting the operation of EndPoint2.

If the value in the MAXP field is rewritten during receiving operation, the operation of USB Controller may become unpredictable. Therefore, the MAXP can be written only when initial setting is being performed.

Bits	Field	R/W	Default	Description
31	EP2EN	R/W	0	EndPoint Enable: When the VR4120A sets this bit to a ‘1’, EndPoint2 is enabled to receive data.
30:21	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
20:19	RM2	R/W	00	Rx Mode: Bit for setting the receive mode. When these bits are set to ‘00’ or ‘01’, data receiving is performed in Normal Mode. When these bits are set to ‘10’, data receiving is performed in Assemble Mode. When these bits are set to ‘11’, data receiving is performed in Separate Mode. For a detailed explanation of the receive modes, see Section **6.6.4**.
18:10	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
9:0	MAXP2	R/W	0	MAX Packet size: The Max Packet Size of EndPoint2. Prior to the start of a USB transaction, the VR4120A must set an appropriate value into this register.

Remark In Normal Mode, indication is issued every received packet so that error status for every packet can be noticed. In the other modes, error status is noticed for all received packets, not for every packet.

6.2.11 U_EP3CR (USB EP3 Control Register)

This register is used for setting the operation of EndPoint3.

If the value in the MAXP field is rewritten during transmitting operation, the operation of USB Controller may become unpredictable. Therefore, the MAXP can be written only when initial setting is being performed.

Bits	Field	R/W	Default	Description
31	EP3EN	R/W	0	EndPoint Enable: When the VR4120A sets this bit to a ‘1’, EndPoint3 is enabled for transmit data.
30:20	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
19	TM3	R/W	0	Tx Mode: Bit for setting the transmit mode. When this bit is set to a ‘0’, transmitting is performed in SZLP Mode. When this bit is set to a ‘1’, transmitting is performed in NZLP Mode. For a detailed explanation of the transmit modes, see Section **6.5.3**.
18	SS3	R/W	0	Transmit Stall: When the VR4120A sets this bit to a ‘1’, a STALL packet is sent from EndPoint3.
17	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
16	NAK3	R/W	0	When the VR4120A sets this bit to a ‘1’, a NAK packet is sent from EndPoint3.
15:7	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
6:0	MAXP3	R/W	0	MAX Packet size: The Max Packet Size for EndPoint3. Prior to the start of a USB transaction, the VR4120A must write an appropriate value into this register.

6.2.12 U_EP4CR (USB EP4 Control Register)

This register is used for setting the operation of EndPoint4.

If the value in the MAXP field is rewritten during receiving operation, the operation of USB Controller may become unpredictable. Therefore, the MAXP can be written only when initial setting is being performed.

Bits	Field	R/W	Default	Description
31	EP4EN	R/W	0	EndPoint Enable: When the VR4120A sets this bit to a ‘1’, EndPoint4 is enabled to receive data.
30:21	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
20:19	RM4	R/W	0	Rx Mode: Bit for setting the receive mode. When these bits are set to ‘00’ or ‘01’, data receiving is performed in Normal Mode. When these bits are set to ‘10’, data receiving is performed in Assemble Mode. When these bits are set to ‘11’, data receiving is performed in Separate Mode. For a detailed explanation of the receive modes, see Section **6.6.4**.
18	SS4	R/W	0	Transmit Stall: When the VR4120A sets this bit to a ‘1’, a STALL handshake is performed at EndPoint4.
17	NHSK4	R/W	0	No Handshake: When the VR4120A sets this bit to a ‘1’, No Handshake is performed at EndPoint4.
16	NAK4	R/W	0	When the VR4120A sets this bit to a ‘1’, a NAK Handshake is performed at EndPoint4.
15:7	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
6:0	MAXP4	R/W	0	MAX Packet size: The Max Packet Size for EndPoint4. Prior to the start of a USB transaction, the VR4120A must set an appropriate value into this register.

6.2.13 U_EP5CR (USB EP5 Control Register)

This register is used for setting the operation of EndPoint5.

If the value in the MAXP field is rewritten during transmitting operation, the operation of USB Controller may become unpredictable. Therefore, the MAXP can be written only when initial setting is being performed.

Bits	Field	R/W	Default	Description
31	EP5EN	R/W	0	EndPoint Enable: When the VR4120A sets this bit to a ‘1’, EndPoint5 is enabled to transmit data.
30:20	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
19	FM	R/W	0	Feedback Mode: When the VR4120A sets this bit to a ‘1’, EndPoint5 performs in feedback mode. (For further information about Feedback mode, please refer to USB Specification 1.1)
18	SS5	R/W	0	Transmit Stall: When the VR4120A sets this bit to a ‘1’, a STALL handshake is performed at EndPoint5.
17	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
16	NAK5	R/W	0	When the VR4120A sets this bit to a ‘1’, a NAK Handshake is performed at EndPoint5.
15:7	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
6:0	MAXP5	R/W	0	MAX Packet size: The Max Packet Size for EndPoint5. Prior to the start of a USB transaction, the VR4120A must set an appropriate value into this register.

6.2.14 U_EP6CR (USB EP6 Control Register)

This register is used for setting the operation of EndPoint6.

If the value in the MAXP field is rewritten during receiving operation, the operation of USB Controller may become unpredictable. Therefore, the MAXP can be written only when initial setting is being performed.

Bits	Field	R/W	Default	Description
31	EP6EN	R/W	0	EndPoint Enable: When the VR4120A sets this bit to a ‘1’, EndPoint6 is enabled to receive data.
30:19	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
18	SS6	R/W	0	Transmit Stall: When the VR4120A sets this bit to a ‘1’, a STALL handshake is performed at EndPoint6.
17	NHSK6	R/W	0	No Handshake: When the VR4120A sets this bit to a ‘1’, No Handshake is performed at EndPoint6.
16	NAK6	R/W	0	When the VR4120A sets this bit to a ‘1’, a NAK Handshake is performed at EndPoint6.
15:7	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
6:0	MAXP6	R/W	0	MAX Packet size: The Max Packet Size for EndPoint6. Prior to the start of a USB transaction, the VR4120A must set an appropriate value into this register.

6.2.15 U_CMR (USB Command Register)

This register is used for issuing Tx request or adding Rx Buffer Directories to Pool.

The VR4120A writes commands into this register.

Whenever B bit (Bit 31) is set, the value will not change even if the VR4120A writes commands into this register.

Bits	Field	R/W	Default	Description
31	B	R/W	0	Busy: Bit that indicates whether the interpretation of an issued command has terminated. When the execution of the command has not yet completed, this bit will be set to a ‘1’. When the execution of a command has completed, this bit will be set to a ‘0’. When the VR4120A tries to issue one command immediately after another, it is necessary to confirm that this bit is set to a ‘0’.
30:27	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
26:24	Command	R/W	000	Field for specifying the type of a command. USB Controller's internal processing varies depending on the value written into this field. 000: Data transmitting at EndPoint0 001: Data transmitting at EndPoint1 010: Data transmitting at EndPoint3 011: Data transmitting at EndPoint5 100: Addition of Buffer Directories to Pool0 101: Addition of Buffer Directories to Pool1 110: Addition of Buffer Directories to Pool2 111: Reserved (Don't Use)
23:16	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
15:0	Data Size/NOD	R/W	0	Data Size/Number Of Buffer Directory: The meaning of this field depends on the value written into the Command field. Command = 0xx: VR4120A has to write the size of the transmitting data in this field. Command = 100, 101, 110: Indicates the number of Buffer Directories added to a pool. Command = 111: This field has no meaning.

6.2.16 U_CA (USB Command Extension Register)

This register is used for issuing Tx request or adding Rx Buffer Directories to Pool.

The VR4120A writes the start address of either the Tx or the Rx Buffer Directory into this register.

Bits	Field	R/W	Default	Description
31:0	Address	R/W	0	The meaning of this field varies depending on the value written into the Command field of the USB Command Register. Command = 0xx: Start address of the transmit packet Command = 100, 101, 110: Start address of the Buffer Directory to be added to the receive pool Command = 111: This register has no meaning.

6.2.17 U_TEPSR (USB Tx EndPoint Status Register)

This register is used for indicate the status of the EndPoint being used for data transmitting.

Bits	Field	R/W	Default	Description
31:26	Reserved	R	0	Reserved for future use
25:24	EP5TS	R	0	EP5 Tx Status: Register that indicates the transmit status of EndPoint5 This register is not cleared, even if read. 00: There is no data scheduled to be sent (Idle) 01: There is one data item scheduled to be sent 10: There are two data items that are scheduled to be sent (Busy)
23:18	Reserved	R	0	Reserved for future use
17:16	EP3TS	R	0	EP3 Tx Status: Register that indicates the transmit status of EndPoint3 This register is not cleared, even if read. 00: There is no data scheduled to be sent (Idle) 01: There is one data item scheduled to be sent 10: There are two data items that are scheduled to be sent (Busy)
15:10	Reserved	R	0	Reserved for future use
9:8	EP1TS	R	0	EP1 Tx Status: Register that indicates the transmit status of EndPoint1 This register is not cleared, even if read. 00: There is no data scheduled to be sent (Idle) 01: There is one data item scheduled to be sent 10: There are two data items that are scheduled to be sent (Busy)
7:2	Reserved	R	0	Reserved for future use
1:0	EP0TS	R	0	EP0 Tx Status: Register that indicates the transmit status of EndPoint0 This register is not cleared, even if read. 00: There is no data scheduled to be sent (Idle) 01: There is one data item scheduled to be sent 10: There are two data items that are scheduled to be sent (Busy)

6.2.18 U_RP0IR (USB Rx Pool0 Information Register)

This register indicates the information of Receive Pool0.

The V_R4120A writes to this register only when the device is being initialized.

Bits	Field	R/W	Default	Description
31	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
30:28	AL	R/W	000	Alert Level: Sets the warning level for Pool0. When the number of Buffer Directories remaining in this pool equals the value set in this field, USB Controller sets the RPA0 bit in the USB General Status Register1 to a ‘1’. Writing N in this field is equivalent to specifying N x 4 (remaining number of Buffer Directories = 4, 8, 12, …, 28). When 000 is written into this field, this function is disabled and no notification is posted to the V_R4120A.
27:16	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
15:0	RNOD	R	0	Remaining Number of Buffer Directory: Indicates the number of Buffer Directories remaining in Pool0. The V_R4120A can only read this field. Buffer Directory addition is performed entirely using the USB Command Register.

6.2.19 U_RP0AR (USB Rx Pool0 Address Register)

This register indicates the start address of Buffer Directory which is currently used.

The way to set up Rx Pool is described at Section **6.6.3 Receive pool settings**.

Bits	Field	R/W	Default	Description
31:0	Address	R	0	Buffer Directory Address: Register that indicates the start address of the first Buffer Directory in Pool0. The VR4120A can only read this register. Buffer Directory addition is performed entirely using the USB Command Register.

6.2.20 U_RP1IR (USB Rx Pool1 Information Register)

This register indicates the information of Receive Pool1.

The VR4120A writes to this register only when the device is being initialized.

Bits	Field	R/W	Default	Description
31	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
30:28	AL	R/W	000	Alert Level: Sets the warning level for Pool1. When the number of Buffer Directories remaining in this pool equals the value set in this field, USB Controller sets the RPA1 bit in the USB General Status Register1 to a ‘1’. Writing N in this field is equivalent to specifying N x 4 (remaining number of Buffer Directories = 4, 8, 12, …, 28). When 000 is written into this field, this function is disabled and no notification is posted to the VR4120A.
27:16	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
15:0	RNOD	R	0	Remaining Number of Buffer Directory: Indicates the number of Buffer Directories remaining in Pool1. The VR4120A can only read this field. Buffer Directory addition is performed entirely using the USB Command Register.

6.2.21 U_RP1AR (USB Rx Pool1 Address Register)

This register indicates the start address of Buffer Directory which is currently used.

The way to set up Rx Pool is described at Section **6.6.3 Receive pool settings**.

Bits	Field	R/W	Default	Description
31:0	Address	R	0	Buffer Directory Address: Register that indicates the start address of the first Buffer Directory in Pool1. The VR4120A can only read this register. Buffer Directory addition is performed entirely using the USB Command Register.

6.2.22 U_RP2IR (USB Rx Pool2 Information Register)

This register indicates the information of Receive Pool2.

The VR4120A writes to this register only when the device is being initialized.

Bits	Field	R/W	Default	Description
31	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
30:28	AL	R/W	000	Alert Level: Sets the warning level for Pool0. When the number of Buffer Directories remaining in this pool equals the value set in this field, USB Controller sets the RPA2 bit in the USB General Status Register1 to a ‘1’. Writing N in this field is equivalent to specifying N x 4 (remaining number of Buffer Directories = 4, 8, 12, …, 28). When 000 is written into this field, this function is disabled and no notification is posted to the VR4120A.
27:16	Reserved	R/W	0	Reserved for future use. Writes ‘0’s.
15:0	RNOD	R	0	Remaining Number of Buffer Directory: Indicates the number of Buffer Directories remaining in Pool2. The VR4120A can only read this field. Buffer Directory addition is performed entirely using the USB Command Register.

6.2.23 U_RP2AR (USB Rx Pool2 Address Register)

This register indicates the start address of Buffer Directory which is currently used.

The way to set up Rx Pool is described at Section **6.6.3 Receive pool settings**.

Bits	Field	R/W	Default	Description
31:0	Address	R	0	Buffer Directory Address: Register that indicates the start address of the first Buffer Directory in Pool2. The VR4120A can only read this register. Buffer Directory addition is performed entirely using the USB Command Register.

6.2.24 U_TMSA (USB Tx MailBox Start Address Register)

Bits	Field	R/W	Default	Description
31:0	Address	R/W	0	Register that indicates the start address of the transmit MailBox area. The VR4120A must set a value in this field only at initialization.

6.2.25 U_TMBA (USB Tx MailBox Bottom Address Register)

Bits	Field	R/W	Default	Description
31:0	Address	R/W	0	Register that indicates the end address of the transmit MailBox area. The VR4120A must set a value in this field only at initialization.

6.2.26 U_TMRA (USB Tx MailBox Read Address Register)

Bits	Field	R/W	Default	Description
31:0	Address	R/W	0	Register that indicates the address of the area that will be read next by the VR4120A. After the VR4120A reads the contents of a MailBox, the value set in this register must be changed by VR4120A.

6.2.27 U_TMWA (USB Tx MailBox Write Address Register)

Bits	Field	R/W	Default	Description
31:0	Address	R	0	Register that indicates the address in the transmit MailBox area to which USB Controller will write next time.

6.2.28 U_RMSA (USB Rx MailBox Start Address Register)

Bits	Field	R/W	Default	Description
31:0	Address	R/W	0	Register that indicates the start address of the receive MailBox area. The VR4120A must set a value in this field only at initialization.

6.2.29 U_RMBA (USB Rx MailBox Bottom Address Register)

Bits	Field	R/W	Default	Description
31:0	Address	R/W	0	Register that indicates the end address of the receive MailBox area. The VR4120A must set a value in this field only at initialization.

6.2.30 U_RMRA (USB Rx MailBox Read Address Register)

Bits	Field	R/W	Default	Description
31:0	Address	R/W	0	Register that indicates the address of the area that will be read next by the VR4120A. After the VR4120A reads the contents of a MailBox, the value set in this register must be changed by VR4120A RISC Processor.

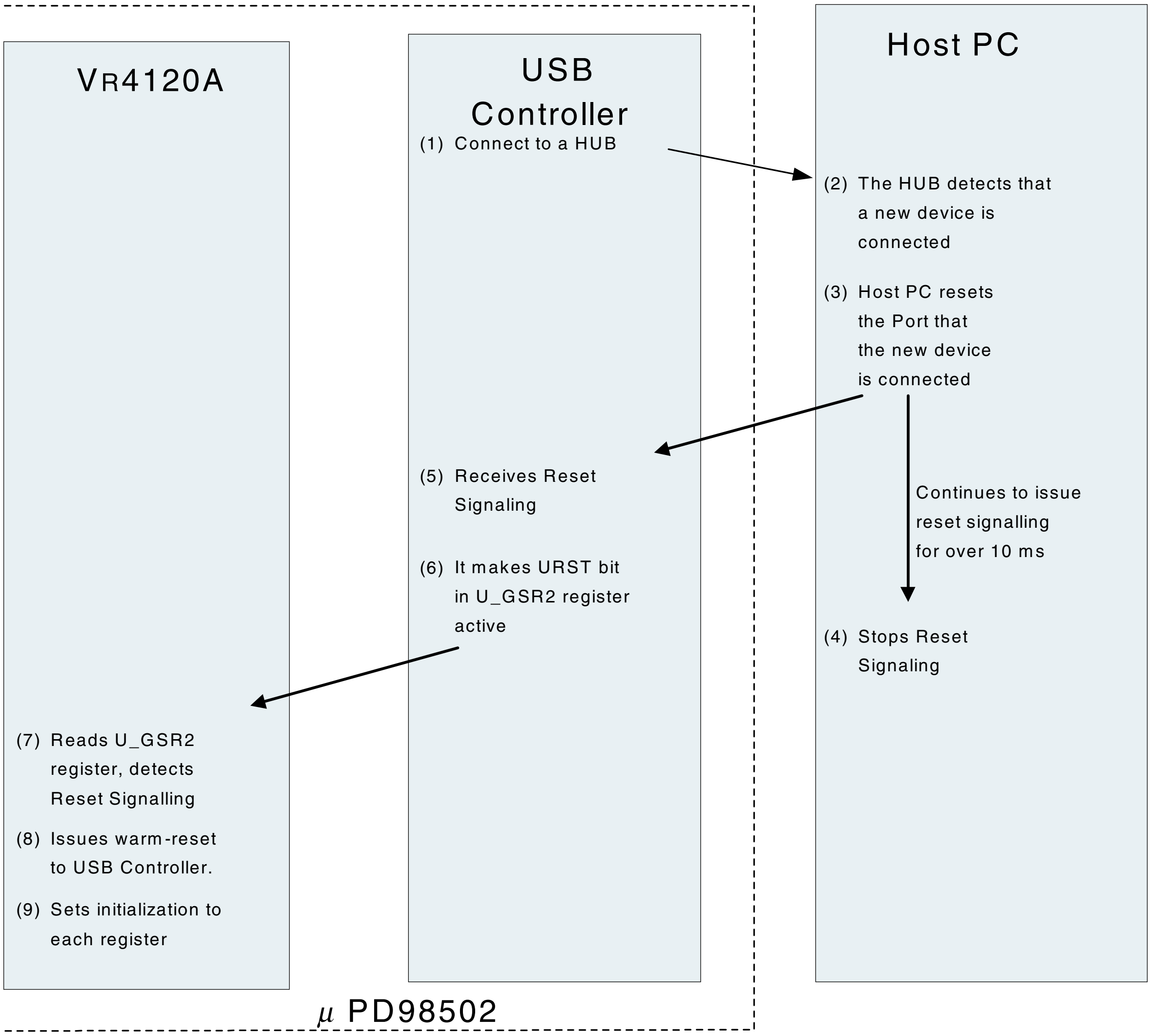
6.2.31 U_RMWA (USB Rx MailBox Write Address Register)

Bits	Field	R/W	Default	Description
31:0	Address	R	0	Register that indicates the address in the receive MailBox area to which USB Controller will write next time.

6.3 USB Attachment Sequence

This section describes the sequence that is followed when the μPD98502 is attached to a USB hub.

Figure 6-2. USB Attachment Sequence


VR4120A
USB Controller
Host PC
(1) Connect to a HUB
(2) The HUB detects that a new device is connected
(3) Host PC resets the Port that the new device is connected
Continues to issue reset signalling for over 10 ms
(4) Stops Reset Signaling
(5) Receives Reset Signaling
(6) It makes URST bit in U_GSR2 register active
(7) Reads U_GSR2 register, detects Reset Signalling
(8) Issues warm-reset to USB Controller.
(9) Sets initialization to each register
μ PD98502

(1) USB port of the μPD98502 is attached to a HUB.

(2) Notification that a new device (μPD98502) has been connected to the HUB is posted to a Host PC.

(3) The Host PC issues Reset Signaling to reset the port to which the device has been connected.

(4) Once 10 ms have elapsed, the Host PC halts the issue of the Reset Signaling.

(5) Notification of the Reset Signaling performed by the Host PC is posted to USB Controller.

(6) To post notification of the Reset by the Host PC to the VR4120A, the URST bit of the U_GSR2 register is made active.

(7) The VR4120A reads the U_GSR2 register and, upon finding that the URST bit is active, detects that Reset Signaling has been issued.

(8) To initialize USB Controller, the VR4120A has to issue the warm-reset to USB Controller.

(9) Upon the completion of the reset, USB Controller performs initialization by writing a value into each of USB Controller's internal registers.

6.4 Initialization

After USB Controller has been reset, the VR4120A must set several USB Controller registers. The initialization sequence is listed below.

(1) A desired mode is set into the USB General Mode Register.

(2) The receive pools are placed in system memory, and the information they contain are set in the following registers:

USB Rx Pool0 Information Register:	(Address: 1000_1050H)
USB Rx Pool0 Address Register:	(Address: 1000_1054H)
USB Rx Pool1 Information Register:	(Address: 1000_1058H)
USB Rx Pool1 Address Register:	(Address: 1000_105CH)
USB Rx Pool2 Information Register:	(Address: 1000_1060H)
USB Rx Pool2 Address Register:	(Address: 1000_1064H)

(3) The transmit/receive MailBoxes are placed in system memory, and the information they contain are set in the following registers:

USB Tx MailBox Start Address Register	(Address: 1000_1070H)
USB Tx MailBox Bottom Address Register	(Address: 1000_1074H)
USB Rx MailBox Start Address Register	(Address: 1000_1080H)
USB Rx MailBox Bottom Address Register	(Address: 1000_1084H)

When Tx MailBox Start Address Register is set, the value in the register is copied to Tx MailBox Read Address Register (Address: 1000_1078H) and Tx MailBox Write Address Register (Address: 1000_107CH).

In same way, when Rx MailBox Start Address Register is set, the value in the register is copied to Rx MailBox Read Address Register (Address: 1000_1088H) and Rx MailBox Write Address Register (Address: 1000_108CH).

(4) A value is written into the MAXP field of the USB EP0 Control Register, the EP1-2 Control Register, the EP3-4 Control Register, and the EP5-6 Control Register. Then, each EndPoint is enabled. The settings made up to this point enable the start of data transmitting/receiving, as well as the ability to respond to Device Configuration from the Host PC.

(5) Device Configuration is started through EndPoint0. Therefore, a response must be returned to EndPoint0. At this stage, the USB Function Address is allocated. The VR4120A must set that value into the Function Address field in USB General Mode Register (Address: 1000_1000H).

The initialization procedure is completed.

In addition to the above, it may be necessary to write a value into the USB Interrupt Mask Register (Address: 1000_1014H or 1000_101CH) and enable the interrupt.

6.4.1 Receive pool settings

For details of the receive pool settings, see Section **6.6.3 Receive pool settings**.

6.4.2 Transmit/receive MailBox settings

After USB Controller transmits a data segment, it indicates the status by writing a transmit indication in ‘MailBox’ in system memory. During the initialization, the VR4120A must set the MailBoxes. USB Controller uses the MailBoxes as a ring buffer. This buffer is set using four registers for each of transmitting and receiving.

Registers for setting a transmit MailBox

U_TMSA (USB Tx MailBox Start Address Register: 1000_1070H)
U_TMBA (USB Tx MailBox Bottom Address Register: 1000_1074H)
U_TMRA (USB Tx MailBox Read Address Register: 1000_1078H)
U_TMWA (USB Tx MailBox Write Address Register: 1000_107CH)

Registers for setting a receive MailBox

U_RMSA (USB Rx MailBox Start Address Register: 1000_1080H)
U_RMBA (USB Rx MailBox Bottom Address Register: 1000_1084H)
U_RMRA (USB Rx MailBox Read Address Register: 1000_1088H)
U_RMWA (USB Rx MailBox Write Address Register: 1000_108CH)

Remarks 1. When the VR4120A writes the value to U_TMSA firstly after reset, USB Controller automatically copies the value to U_TMRA and U_TMWA internally. Similarly, when VR4120A writes the value to RMSA firstly after reset, USB Controller automatically copies the value to U_RMRA and U_RMWA internally.

2. Do not set the same values for U_TMSA and U_TMBA.

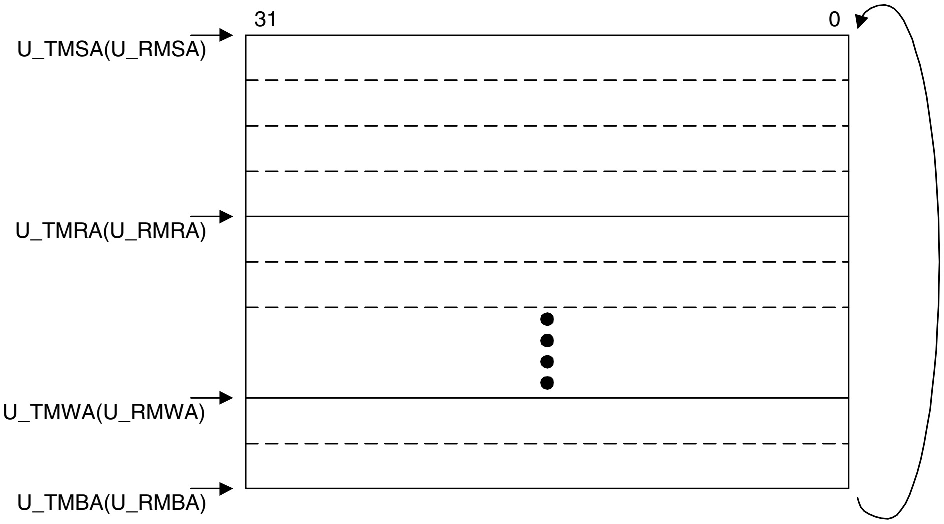
3. Among those registers used to set the MailBoxes, the VR4120A updates only U_TMRA and U_RMRA. The other registers are written to only as part of initialization. They are not to be written to at any other time.

4. Each receive indication has a two-word configuration. Therefore, the size of the receive MailBox must be an integer multiple of two words.

5. The MailBox areas must be word-aligned.

The configuration of the MailBoxes in system memory is as shown in the Following **Figure 6-3**.

Figure 6-3. Mailbox Configuration



When USB Controller writes an indication, the write pointer (U_TMWA or U_RMWA) is incremented. Every time that USB Controller writes an indication, it also sets the transmit/receive finish bit of the corresponding EndPoint and, issues an interrupt if it is not masked.

The write pointer is forced to jump to the start address (U_TMSA or U_RMSA) when it reaches the bottom address (U_TMBA or U_RMBA). USB Controller uses the read pointer (U_TMRA or U_RMRA) to prevent the overwriting of those indications that the VR4120A has not yet read out. The read pointer (U_TMRA or U_RMRA) is managed by the VR4120A. Each time the VR4120A reads an indication from a MailBox, it writes the address to be read next time into the read pointer register (U_TMRA or U_RMRA).
When both the write pointer (U_TMWA or U_RMWA) and read pointer (U_TMRA or U_RMRA) point to the same address, USB Controller sets the TMF bit (transmit MailBox full) or RMF bit (receive MailBox full) of the USB General Status Register 1 to indicate the MailBox full state and issues an interrupt if it is not masked.

In the MailBox full status, USB Controller will not issue the next indication. The VR4120A must read an indication from the full MailBox and update the read pointer (U_TMRA or U_RMRA).

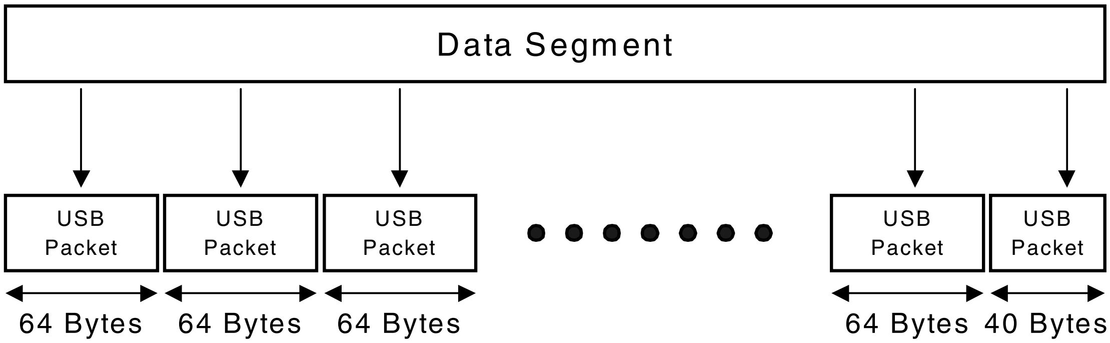
6.5 Data Transmit Function

This section explains USB Controller's data transmit function.

6.5.1 Overview of transmit processing

USB Controller divides the data segments in system memory, into USB packets, then transmits them to the Host PC. The VR4120A sets the size of USB packet in the MAXP field of the EP0 Control Register, the EP1-2 Control Register, the EP3-4 Control Register, and the EP5-6 Control Register (in the example shown below, a value of 64 bytes has been set).

Figure 6-4. Division of Data into USB Packets



Last USB packet in a data segment will be smaller than the value set in the MAXP field (40 bytes in the example shown above). As a result, the Host PC can identify the boundary between data segments. When the last USB packet size is equal to the value in the MAXP field, a zero-length USB packet will be transmitted after the last item of the divided data in transmit SZLP Mode. In transmit NZLP Mode, a zero-length USB packet is not transmitted. For an explanation of the transmit modes, see Section **6.5.3 Data transmit modes**.

After all the data segments have been transferred, USB Controller writes the "transmit indication" which has transmit status information into the MailBox in system memory.

For an explanation of the transmit indication, see Section **6.5.6 Tx indication**.

Data segments transmission time at a given EndPoint can be scheduled upon the issue of the corresponding transmit command. The current transmit status can be determined by reading the contents of the USB Tx EndPoint Status Register (Address: 1000_1048H).

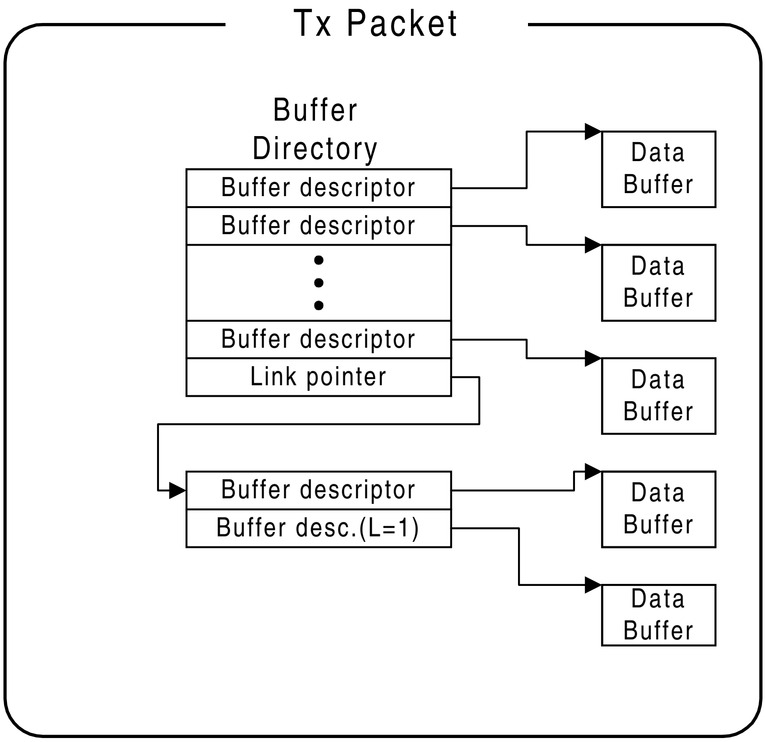
For an explanation of how to issue the transmit command, see Section **6.5.4 VR4120A processing at data transmitting**.

6.5.2 Tx buffer configuration

The VR4120A creates a Tx buffer in system memory, then informs USB Controller to transmit data to the Host PC.

The configuration of the Tx buffer is as shown below.

Figure 6-5. Tx Buffer Configuration



A transmit packet is configured by breaking up multiple data buffers in system memory. These data buffers are bundled together in the buffer directory.

The formats of the Buffer directory, Buffer descriptor, and Link Pointer in the Tx buffer are as shown below.

Figure 6-6. Configuration of Transmit Buffer Directory

-Tx Buffer Directory

<table>
<tr><th>31</th><th>0</th></tr>
<tr><td colspan="2">Buffer Descriptor 0</td></tr>
<tr><td colspan="2">Buffer Descriptor 1</td></tr>
<tr><td colspan="2">Buffer Descriptor 2</td></tr>
<tr><td colspan="2">Buffer Descriptor 3</td></tr>
<tr><td colspan="2">Buffer Descriptor 4</td></tr>
<tr><td colspan="2">l</td></tr>
<tr><td colspan="2">Buffer Descriptor N</td></tr>
<tr><td colspan="2">Link Pointer</td></tr>
</table>

-Tx Buffer Descriptor

<table>
<tr><th>31</th><th>30</th><th>29 16</th><th>15 0</th></tr>
<tr><td>L</td><td>1</td><td>Reserved</td><td>Size</td></tr>
<tr><td colspan="4">Buffer Address</td></tr>
</table>

-Tx Link Pointer

<table>
<tr><th>31</th><th>30</th><th>29 0</th></tr>
<tr><td>0</td><td>0</td><td>Reserved</td></tr>
<tr><td colspan="3">Directory Address</td></tr>
</table>

Tx Buffer Directory: This is the Tx buffer directory. It contains the buffer descriptor and the link pointer. A single Tx Buffer Directory can accommodate up to 255 buffer descriptors.

Tx Buffer Descriptor: This is the Tx buffer descriptor. It maintains the data in the Tx BUFFER.

When Bit 31 (Last bit) is set, the Buffer Descriptor indicates the last buffer in a packet.

Bit30 is used to discriminate between the Buffer Descriptor and Link Pointer. When set to 1, this bit indicates the Buffer Descriptor.

The "Size" field indicates the buffer size. As the buffer size, a value between 1 and 65535 bytes can be set. The "Buffer Address" field indicates the start address of the buffer.

Tx Link Pointer: This is the link pointer. It points to the next packet directory.

Bit31 is usually set to 0.

Bit30 is used to discriminate between the Buffer Descriptor and Link Pointer. When set to 0, this bit indicates the Link Pointer.

The "Directory Address" field indicates the start address of the next Buffer Directory.

6.5.3 Data transmit modes

USB Controller supports two transmit modes. These modes differ only in whether a zero-length USB packet is transmitted after the last USB packet of a data segment. In all other aspects, they are identical. The transmit mode is switched using the TM bit (Bit 19) of the USB EP1 EndPoint Control Register (Address: 1000_1024H) and USB EP3 EndPoint Control Register (Address: 1000_102CH).

Cautions 1. The setting of the TM bit applies only to EndPoint1 and EndPoint3.

2. When EndPoint0 and EndPoint5 are used to transmit data, only NZLP mode is used.

(1) SZLP (Transmit Zero Length Packet) mode

In this mode, when the last USB packet size of a data segment is equal to the value in the MAXP field, a zero-length packet is transmitted after the completion of data segment transmitting.

When the last packet size is less than the value in the MAXP field, the last Short Packet is transmitted and any zero-length packet is not transmitted.

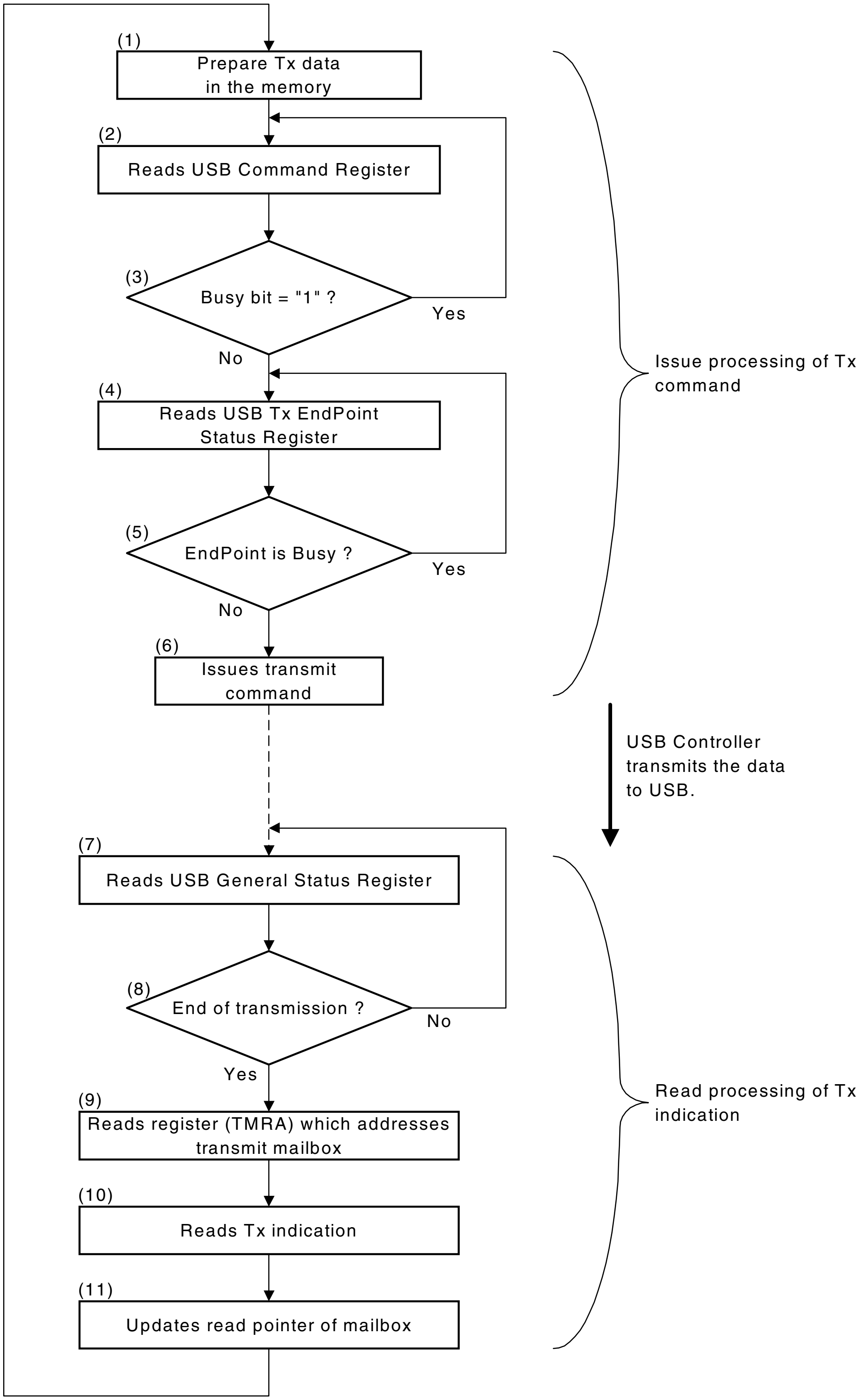
(2) NZLP (Non Zero Length Packet) mode

In this mode, even when the last USB packet size of a data segment is equal to the value in the MAXP field, any zero-length packet is not transmitted after the last packet transmission.

6.5.4 VR4120A processing at data transmitting

This section explains the processing performed by the VR4120A when transmitting data.

Figure 6-7. VR4120A Processing at Data Transmitting


(1)
Prepare Tx data in the memory
(2)
Reads USB Command Register
(3)
Busy bit = "1" ?
Yes
No
(4)
Reads USB Tx EndPoint Status Register
(5)
EndPoint is Busy ?
Yes
No
(6)
Issues transmit command
Issue processing of Tx command
USB Controller transmits the data to USB.
(7)
Reads USB General Status Register
(8)
End of transmission ?
No
Yes
(9)
Reads register (TMRA) which addresses transmit mailbox
(10)
Reads Tx indication
(11)
Updates read pointer of mailbox
Read processing of Tx indication

(1) First, the VR4120A prepares the data to be transmitted in system memory.
(2) The VR4120A reads the USB Command Register.
(3) The VR4120A checks whether the Busy bit of the USB Command Register is set. If the Busy bit is set, it indicates that USB Controller is still executing the previous command. Thus the VR4120A can not issue a new command.
(4) The VR4120A reads the USB Tx EndPoint Status Register.
(5) The VR4120A checks whether the EndPoint that is to perform transmission next is in the Busy status. If the EndPoint is Busy, the VR4120A repeats the processing from the reading of the USB Tx EndPoint Status Register.
(6) The VR4120A issues the Tx command.
(7) The VR4120A reads the USB General Status Registers 1.
(8) The VR4120A checks whether transmission has completed.
(9) The VR4120A reads the contents of the USB Tx Mailbox Read Address Register.
(10) The VR4120A reads the transmit Indication.
(11) The VR4120A updates the contents of the USB Tx MailBox Read Address Register.

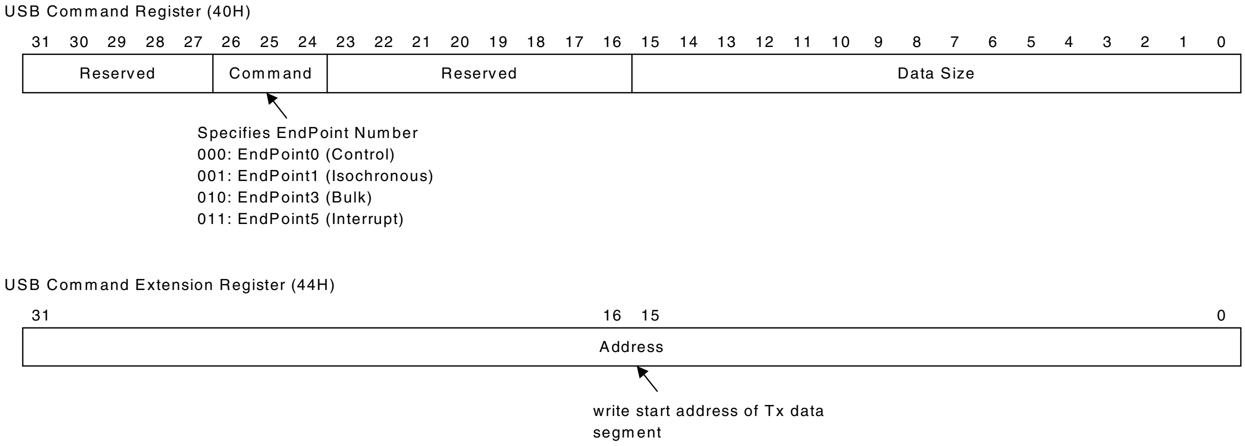
By issuing a transmit command, the transmission of a data segment can be scheduled.

A transmit command is issued by writing a value into the registers listed below. When writing, it is necessary to write first to the USB Command Extension Register, then the USB Command Register.

If data size field of USB Command Register is “0”, USB Controller transmits Zero-Length Packet.

The size of data set by the transmission command and the total size of data buffer to be transmitted have to be the same value.

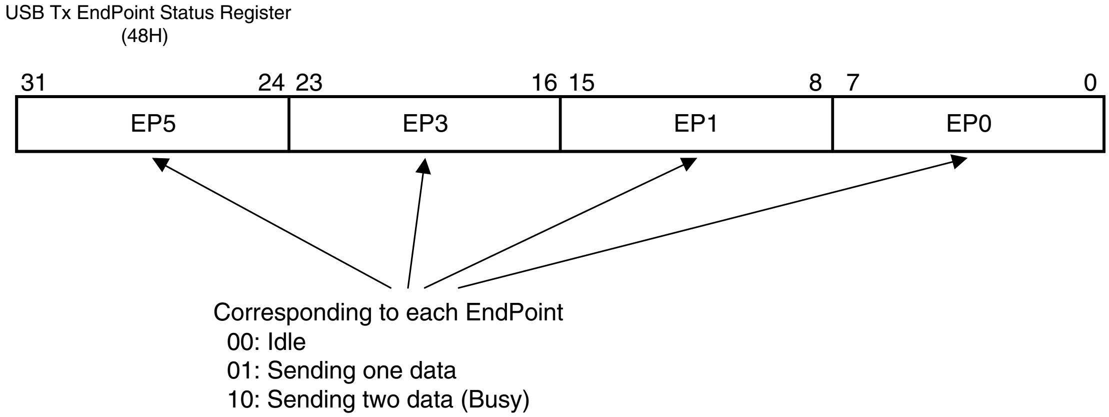
Figure 6-8. Transmit Command Issue


USB Command Register (40H)
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
Command
Reserved
Data Size
Specifies EndPoint Number
000: EndPoint0 (Control)
001: EndPoint1 (Isochronous)
010: EndPoint3 (Bulk)
011: EndPoint5 (Interrupt)
USB Command Extension Register (44H)
31
16 15
0
Address
write start address of Tx data segment

For a given EndPoint, it is possible to schedule the transmitting of up to two data items. Once two data have been scheduled, even if the VR4120A writes a transmit command into the USB Command Register to transmit a third data, that command cannot be accepted until the first scheduled data is transmitted.

The number of data items that are scheduled for transmission can be determined by reading the USB Tx EndPoint Status Register.

Figure 6-9. Transmit Status Register

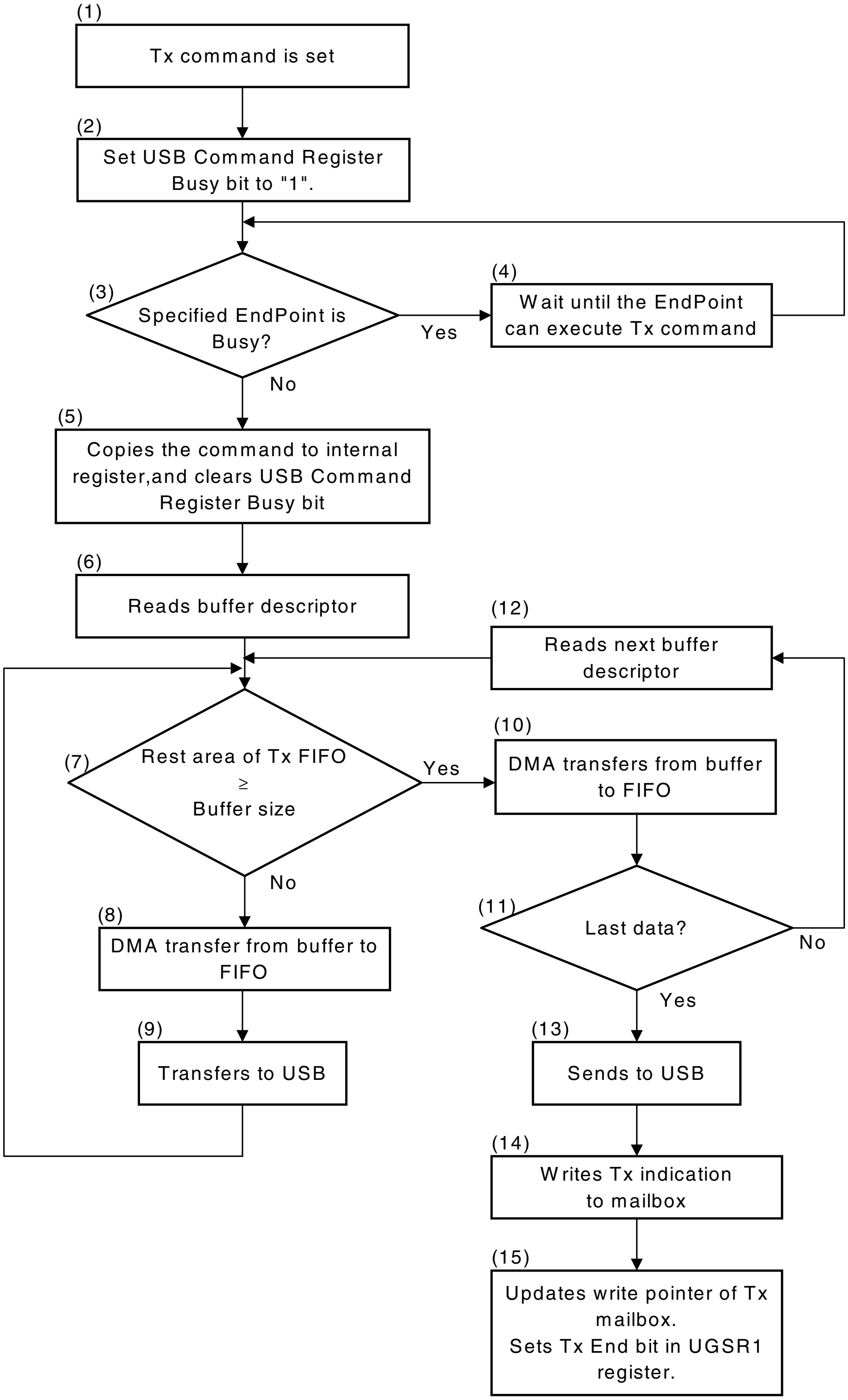


Corresponding to each EndPoint
00: Idle
01: Sending one data
10: Sending two data (Busy)

6.5.5 USB controller processing at data transmitting

This section presents all of the processing performed by USB Controller at data transmitting.

Figure 6-10. USB Controller Transmit Operation Flow Chart


(1)
Tx command is set
(2)
Set USB Command Register Busy bit to "1".
(3)
Specified EndPoint is Busy?
Yes
(4)
Wait until the EndPoint can execute Tx command
No
(5)
Copies the command to internal register,and clears USB Command Register Busy bit
(6)
Reads buffer descriptor
(12)
Reads next buffer descriptor
(7)
Rest area of Tx FIFO ≥ Buffer size
Yes
(10)
DMA transfers from buffer to FIFO
No
(8)
DMA transfer from buffer to FIFO
(11)
Last data?
No
Yes
(9)
Transfers to USB
(13)
Sends to USB
(14)
Writes Tx indication to mailbox
(15)
Updates write pointer of Tx mailbox.
Sets Tx End bit in UGSR1 register.

Numbers (1) to (15) do not indicate the order in which USB Controller must perform processing. Instead, these numbers correspond to those in the following explanation.

(1) USB Controller starts transmit processing upon receiving a transmit command from the VR4120A.
(2) When the command is written, USB Controller sets the Busy bit in USB Command Register to a ‘1’.
(3) USB Controller checks whether the EndPoint specified with the transmit command is currently in the Busy status (two data are scheduled to be transmitted).
(4) If the EndPoint specified with the transmit command is found to be in the Busy status, the command written at (1) is not executed until the specified EndPoint can accept a new Tx command.
(5) The command written into the USB Command Register and USB Command Extension Register in (1) is copied to an internal register and the Busy bit of the USB Command Register is returned to a ‘0’.
(6) USB Controller reads the buffer descriptor.
(7) USB Controller compares the size of the area remaining in the Tx FIFO with the buffer size of the buffer descriptor read in the previous step.
(8) If step (7) reveals that the area remaining in the Tx FIFO is smaller, USB Controller transfers the data from the buffer until the Tx FIFO is full by DMA.
(9) Once the Tx FIFO is full, USB Controller transfers the data to the USB.
(10) If step (7) reveals that the area remaining in the Tx FIFO is larger, USB Controller transfers all the data in the buffer to the Tx FIFO by DMA.
(11) USB Controller checks whether the data transferred by DMA is the last data of the data segments to be transmitted to a Host.
(12) If the data transferred by DMA is not the last to be transmitted, it indicates that the buffer is empty. Therefore, USB Controller reads the next buffer descriptor.
(13) If the data transferred by DMA is the last to be transmitted, USB Controller transmits the data.
(14) USB Controller writes a Tx indication in the MailBox.
(15) USB Controller updates the MailBox write pointer (USB Tx MailBox Write Address Register). It also sets the transmit finish bit of the USB General Status Register 1, and if it is not masked, issues an interrupt to the VR4120A.

6.5.6 Tx indication

For every data segment to be transmitted, USB Controller writes a Tx indication into the Tx MailBox. After writing a Tx indication, USB Controller sets the transmit completion bit of USB General Status Register1 to 1 and, provided it is not masked, issues an interrupt to the VR4120A.

The format of the transmit indication is as shown below.

Figure 6-11. Transmit Indication Format

31 16 15 11	10 8	7 3	2 1 0
Reserved	Status	Reserved	EPN

Status: Field that indicates the status upon data transmitting.

Bit10: When set to a ‘0’, indicates that an IBUS error has not occurred.
When set to a ‘1’, indicates that processing is terminated abnormally due to an IBUS error.

Bit9: When set to a ‘0’, indicates that a buffer underrun did not occur during data transmitting.
When set to a ‘1’, indicates that a buffer underrun occurred.
This bit is set only when transmitting data to EndPoint1.

Bit8: When set to a ‘0’, data transmitting is performed in SZLP mode.
When set to a ‘1’, data transmitting is performed in NZLP mode.
For EndPoint0 and EndPoint5, this bit is usually set to a ‘1’.

EPN: Field that indicates the EndPoint number.

000:	EndPoint0
010:	EndPoint1
100:	EndPoint3
110:	EndPoint5

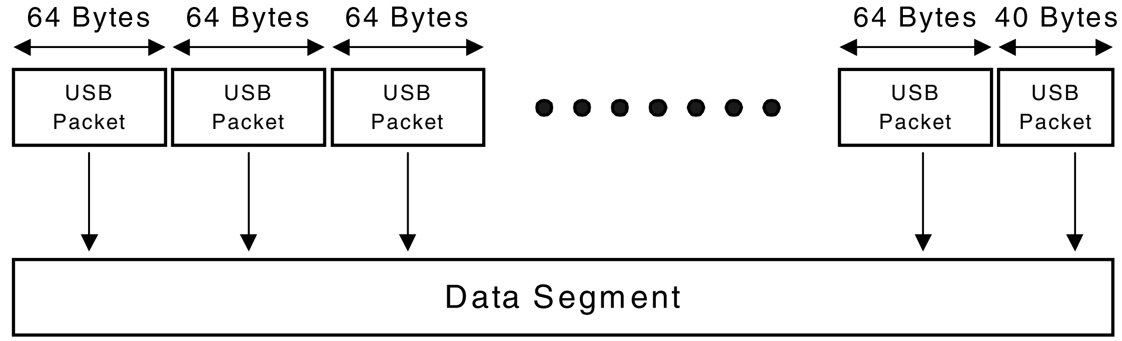
6.6 Data Receive Function

This section explains USB Controller's data receive function.

6.6.1 Overview of receive processing

USB Controller receives USB packets from the USB, stores them into system memory, and then assembles a single data segment. The VR4120A sets the size of a single USB packet in the MAXP field of the EP0 Control Register, EP1-2 Control Register, EP3-4 Control Register, and EP5-6 Control Register. (The figure shown below is an example when the packet size is set to 64 bytes.)

Figure 6-12. Division of Data into USB Packets



When the data segments are divided to USB packets with the same byte size as a value set in the MAXP field, the last USB packet of the data segment will be smaller than the value set in the MAXP field (40 bytes in the example shown above). As a result, USB Controller can identify the boundary between data segments. If the last USB buffer size is equal to the value in the MAXP field, a zero-length USB packet will be transmitted from the Host PC to USB Controller after the last USB packet of the data segment.

When placing data received from the USB to system memory, an area to store received USB packet is required in system memory. This area is referred to as the Rx buffer. It must be secured by the VR4120A. For an explanation of the Rx buffer, see Section **6.6.2 Rx Buffer configuration**.

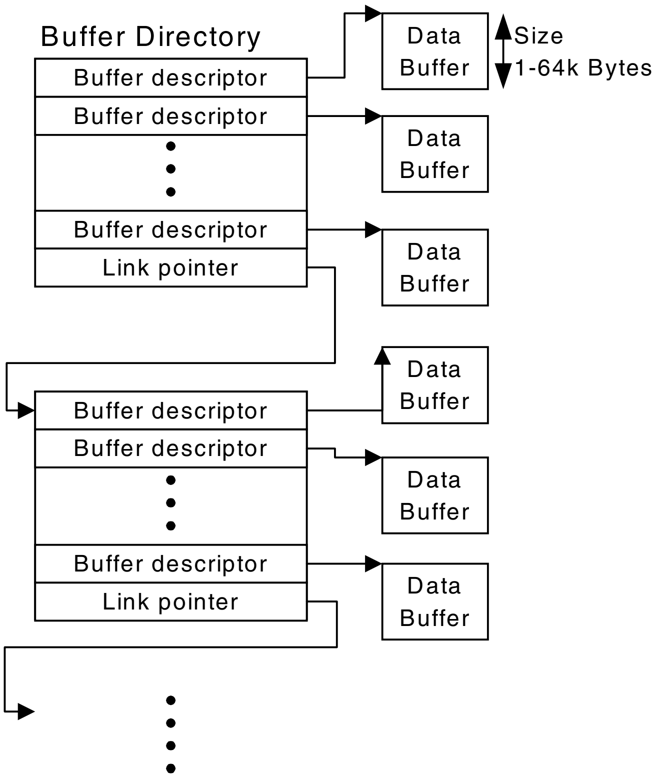
Upon the completion of data segment transfer, USB Controller writes the "Rx indication" into a MailBox in system memory. For an explanation of the Rx modes, see Section **6.6.4 Data receive mode**.

6.6.2 Rx Buffer configuration

Data received from the USB is stored into a receive pool in system memory.

USB Controller uses three receive pools. The configuration of the receive pools is shown below.

Figure 6-13. Receive Buffer Configuration



The receive buffer is composed of the Buffer Directory and Data Buffer. The receive buffer is prepared in system memory by the VR4120A.

The formats of the Buffer Directory, Buffer descriptor, and Link Pointer are each described below.

Figure 6-14. Receive Descriptor Configuration

-Rx Buffer Directory

<table>
<tr><th>31 … 0</th></tr>
<tr><td>Buffer Desciptor 0</td></tr>
<tr><td>Buffer Desciptor 1</td></tr>
<tr><td>Buffer Desciptor 2</td></tr>
<tr><td>Buffer Desciptor 3</td></tr>
<tr><td>Buffer Desciptor 4</td></tr>
<tr><td>|</td></tr>
<tr><td>Buffer Desciptor N</td></tr>
<tr><td>Link Pointer</td></tr>
</table>

-Rx Buffer Descriptor

<table>
<tr><th>31</th><th>30</th><th>29 … 16</th><th>15 … 0</th></tr>
<tr><td>L</td><td>1</td><td>Reserved</td><td>Size</td></tr>
<tr><td colspan="4">Buffer Address</td></tr>
</table>

-Rx Link Pointer

<table>
<tr><th>31</th><th>30</th><th>29 … 16</th><th>15 … 0</th></tr>
<tr><td>0</td><td>0</td><td colspan="2">Reserved</td></tr>
<tr><td colspan="4">Buffer Directory Address</td></tr>
</table>

Rx Buffer Directory: A Rx Buffer Directory. This is composed of buffer descriptors and a link pointer. A single Rx Buffer Directory can contain up to 255 buffer descriptors.

Rx Buffer Descriptor: Contains the Rx buffer data.

When Bit31 (Last bit) is set to a ‘1’, that Buffer Descriptor indicates the last buffer in the pool.

Bit30 is used to discriminate between the Buffer Descriptor and Link Pointer. When set to 1, this bit indicates the Buffer Descriptor.

The "Size" field indicates the buffer size. As the buffer size, a value between 1 and 65535 bytes can be set. The "Buffer Address" field indicates the start address of the buffer.

Rx Link Pointer: This is the link pointer. It indicates the last Buffer Directory.

Bit31 is set to 0.

Bit30 is used to discriminate between the Buffer Descriptor and Link Pointer. When set to 0, this bit indicates the Link Pointer.

The "Buffer Directory Address" field indicates the start address of the next Buffer Directory.

6.6.3 Receive pool settings

USB Controller uses three receive pools.

Pool0	**For EndPoint0 (Control) and EndPoint6 (Interrupt)**
Pool1	**For EndPoint2 (Isochronous)**
Pool2	**For EndPoint4 (Bulk)**

The data in each of these three pools is written into the corresponding registers.

Pool0	**USB Rx Pool0 Information Register**	**(Address: 1000_1050H)**
	USB Rx Pool0 Address Register	**(Address: 1000_1054H)**
Pool1	**USB Rx Pool1 Information Register**	**(Address: 1000_1058H)**
	USB Rx Pool1 Address Register	**(Address: 1000_105CH)**
Pool2	**USB Rx Pool2 Information Register**	**(Address: 1000_1060H)**
	USB Rx Pool2 Address Register	**(Address: 1000_1064H)**

The VR4120A can know the current status of each pool by reading these registers.

The VR4120A can write values only into the Alert field of three Information Registers above. Other filed must be set using USB Command Register and USB Command Extension Register.

The VR4120A adds Buffer Directories to each pool by using the USB Command Register (Address: 1000_1040H) and the USB Command Extension Register (Address: 1000_1044H).

To add Buffer Directories to a receive pool, the VR4120A performs the following processing.

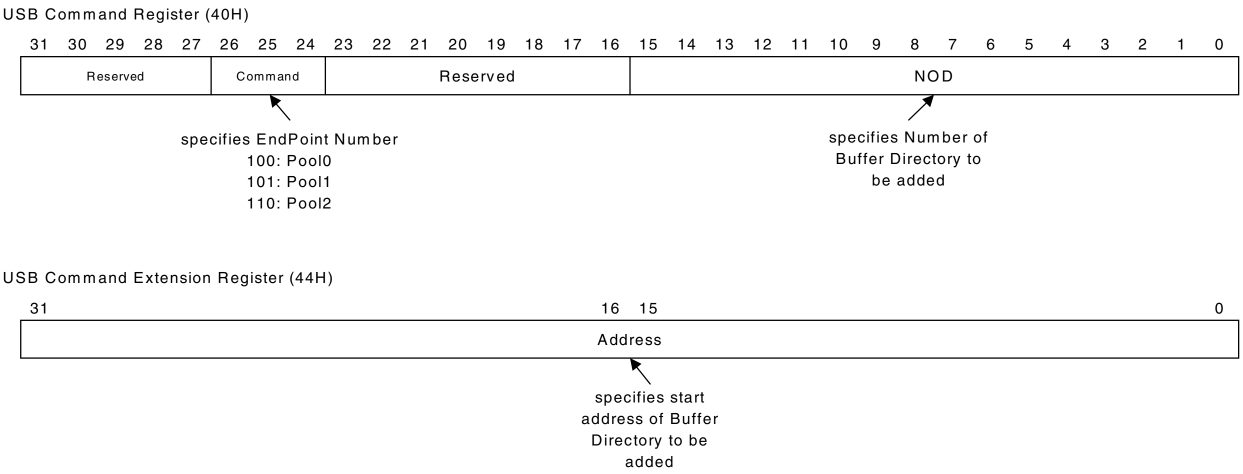
(1) The VR4120A places the Buffer Directory to be added to the pool, and the buffer, in system memory. When multiple Buffer Directories are to be added, they are linked in advance.

(2) The VR4120A sets the start address of the Buffer Directory to be added into the link pointer to the last Buffer Directory in the list of dependent Buffer Directories in the pool.

(3) The VR4120A sets the start address of the Buffer Directory to be added into the USB Command Extension Register (Address: 1000_1044H).

(4) The VR4120A sets the pool number and size of the Buffer Directory to be added into the USB Command Register (Address: 1000_1040H).

Figure 6-15. Buffer Directory Addition Command


USB Command Register (40H)
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
Command
Reserved
NOD
specifies EndPoint Number
100: Pool0
101: Pool1
110: Pool2
specifies Number of Buffer Directory to be added
USB Command Extension Register (44H)
31
16 15
0
Address
specifies start address of Buffer Directory to be added

The operation of USB Controller varies with whether any unused Buffer Directories remain in the corresponding pool when the Buffer Directory addition command is written into the USB Command Register.

(a) If any unused Buffer Directories remain in the pool (when the RNOD field in the Pool Information Register is set to grater than 0), USB Controller adds the number in the NOD field of the command to the RNOD field of the Pool Information Register.

(b) When the pool is empty (when the RNOD field in the Pool Information Register is 0), USB Controller loads the value set in the NOD field of the command into the RNOD field of the Pool Information Register. Furthermore, it loads the value written in the USB Command Extension Register into the Pool Address Register.

6.6.4 Data receive mode

USB Controller has different receive processing every EndPoint and receive mode.

The receive mode is determined by RM field (Bits 20:19) in USB EP2 Control Register (Address 1000_1028H) and USB EP4 Control Register (Address 1000_1030H). There are four kinds of receive processing.

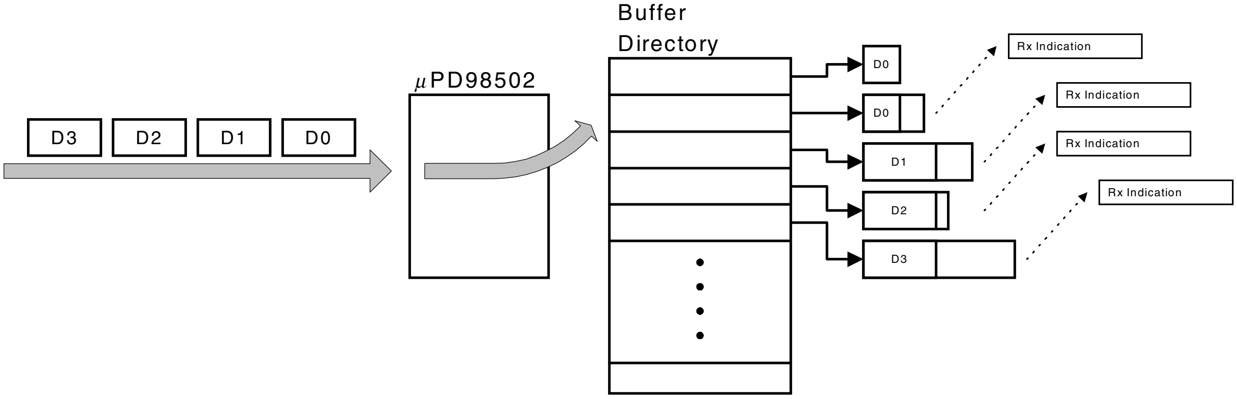
(1) EndPoint0, EndPoint6
(2) EndPoint2, EndPoint4 Normal Mode
(3) EndPoint2, EndPoint4 Assemble Mode
(4) EndPoint2, EndPoint4 Separate Mode

Each processing is explained below.

(1) Reception in EndPoint0, EndPoint6

Same processing is executed without relations in receive mode in EndPoint0, EndPoint6 every time.

Figure 6-16. Data Receiving in EndPoint0, EndPoint6

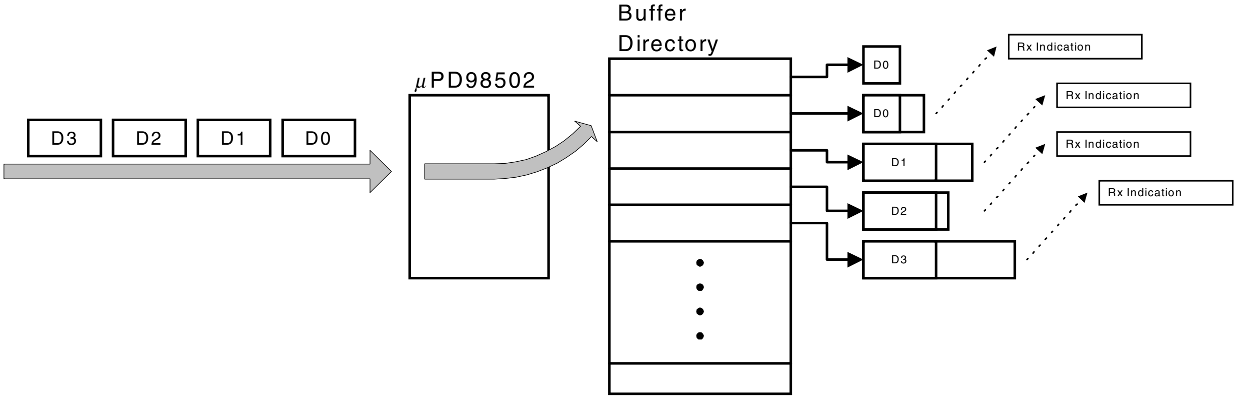


When USB Controller receives one USB packet, stores it in Data Buffer and write Rx indication to the Mailbox. USB Controller updates the size field and Last field in Buffer Descriptor every USB packet before writing Rx Indication.

(2) EndPoint2, EndPoint4, normal mode

The processing in EndPoint2, EndPoint4 receive Normal mode is explained below.

Figure 6-17. EndPoint2, EndPoint4 Receive Normal Mode

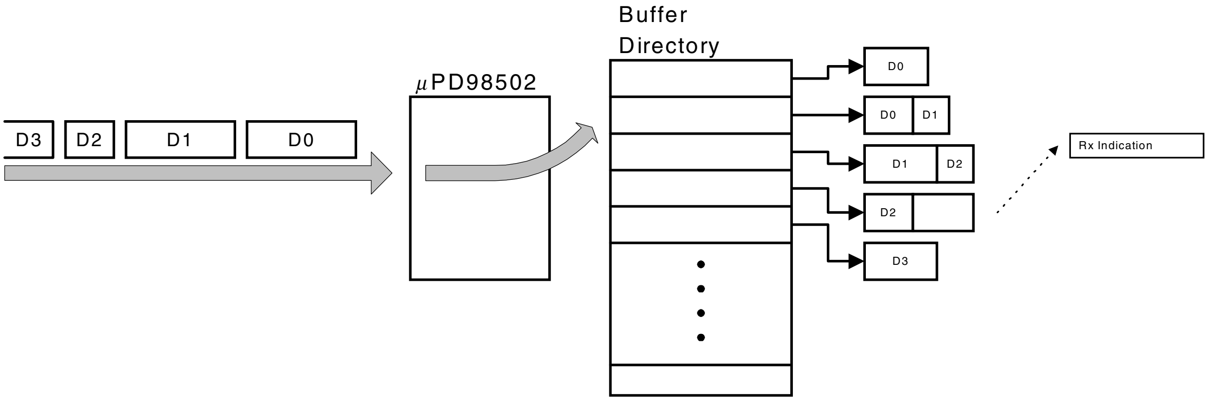


When USB Controller receives one USB packet, stores it in Data Buffer and write Rx indication to the Mailbox. USB Controller updates the size field and Last field in Buffer Descriptor every USB packet before writing Rx Indication.

(3) EndPoint2, EndPoint4, assemble mode

The processing in EndPoint2, EndPoint4 receive Assemble mode is explained below.

Figure 6-18. EndPoint2, EndPoint4 Receive Assemble Mode



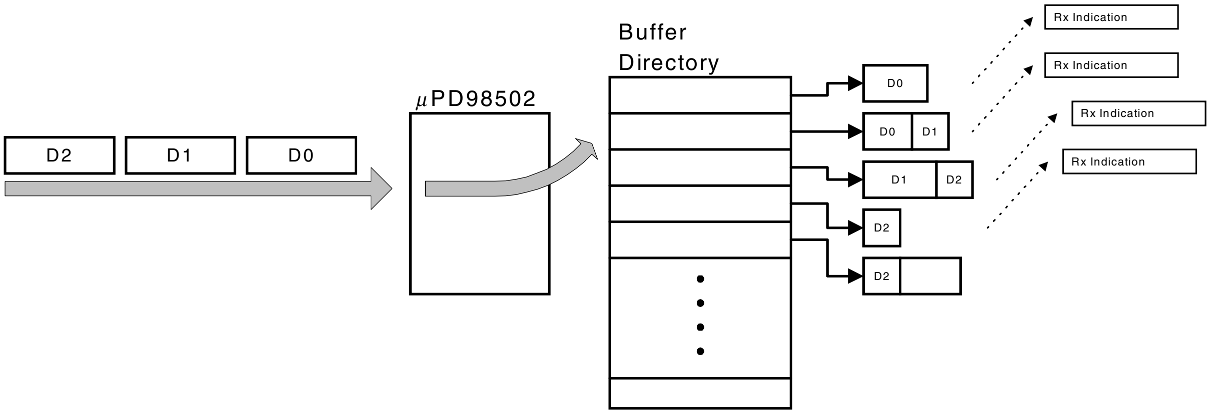
In this mode USB Controller issues Rx indication after receiving one data segment.

In other word, after USB Controller writes the Short packet or Zero-Length Packet received from USB to buffer in system memory, USB Controller updates Size field, Last bit in last Buffer Descriptor and issues Rx indication.

(4) EndPoint2, EndPoint4, separate mode

The processing in EndPoint2, EndPoint4 receive separate mode is explained below.

Figure 6-19. EndPoint2, EndPoint4 Receive Separate Mode



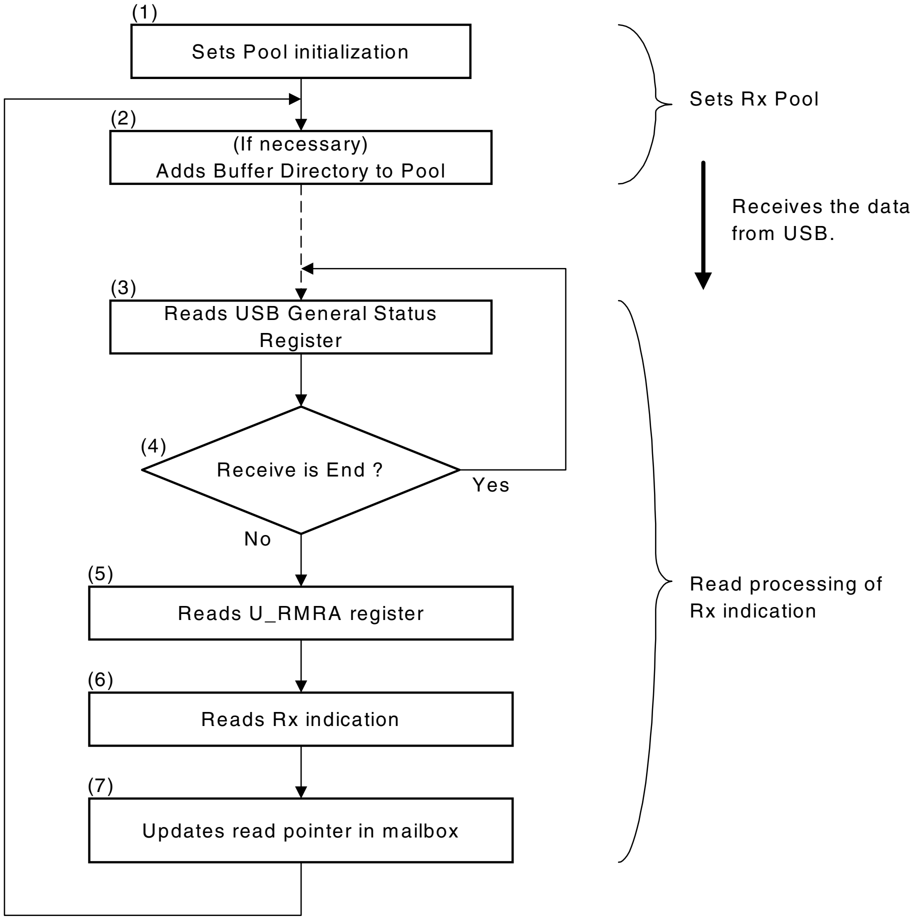
In this mode, after USB Controller receives USB packet and stores the data in Rx buffer, it issues Rx indication when a buffer is full. Even if a buffer is full in the middle of USB packets, it issues indication. After that, processing of storing the USB packet to next buffer continues.

It does not execute to update Size field, Last bit in Buffer Descriptor.

6.6.5 VR4120A receive processing

This section explains the processing that the VR4120A must perform when data is being received.

Figure 6-20. VR4120A Receive Processing


(1)
Sets Pool initialization
(2)
(If necessary)
Adds Buffer Directory to Pool
Sets Rx Pool
Receives the data from USB.
(3)
Reads USB General Status Register
(4)
Receive is End ?
Yes
No
(5)
Reads U_RMRA register
(6)
Reads Rx indication
(7)
Updates read pointer in mailbox
Read processing of Rx indication

Numbers (1) to (7) do not indicate the order in which the VR4120A must perform processing. Instead, these numbers correspond to those in the following explanation.

(1) First, as part of initialization, the VR4120A must set Pool configuration.
(2) For receiving, the VR4120A must add Buffer Directories to the Pool, if necessary.
(3) The VR4120A reads the USB General Status Register.
(4) The VR4120A checks whether receiving has ended.
(5) If receiving has ended, the VR4120A reads USB Rx MailBox Read Address Register (Address: 1000_1088H) to determine the address of MailBox VR4120A must read in the next time.
(6) Then, the VR4120A reads the Rx indication from the indicated MailBox.
(7) The VR4120A updates the USB Rx MailBox Read Address Register.

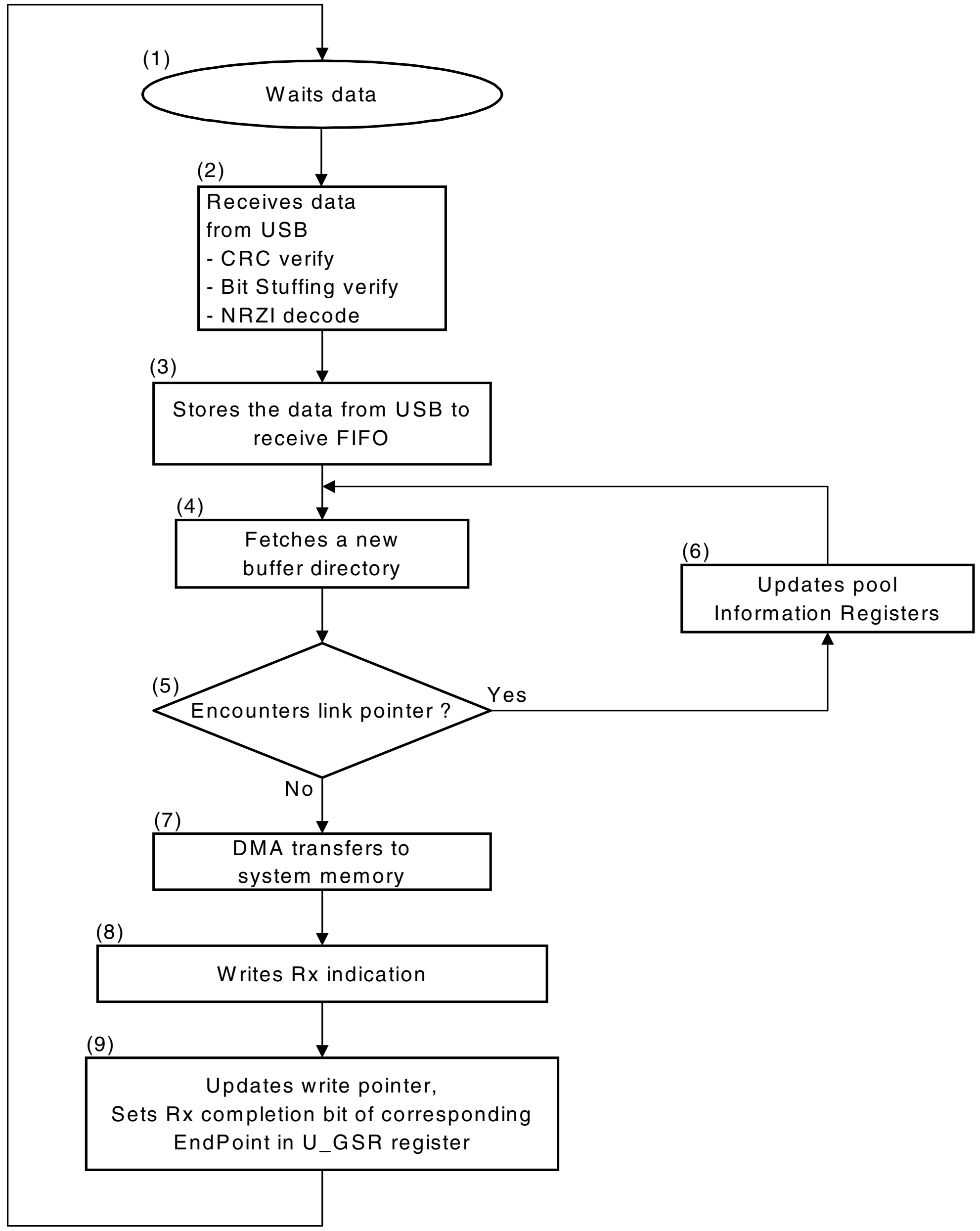
6.6.6 USB controller receive processing

This section presents all of the processing performed by USB Controller at data receiving.

6.6.6.1 Normal mode

The following figure illustrates the receive operations performed by USB Controller in Normal Mode.

Figure 6-21. USB Controller Receive Operations (Normal Mode)



Numbers (1) to (9) do not indicate the order in which USB Controller must perform processing. Instead, these numbers correspond to those in the following explanation.

(1) USB Controller is in the status where it waits to receive data (USB Packets) from the USB.

(2) USB Controller receives data (USB Packets) from the USB. As it is receiving the data, USB Controller performs NRZI decoding, CRC check, and Bit Stuffing Error check.

(3) USB Controller stores the received data into the FIFO.

(4) USB Controller starts to fetch a new buffer descriptor.

(5) USB Controller checks whether the fetched buffer descriptor is a link pointer or not.

(6) If the fetched buffer descriptor is a link pointer, USB Controller updates the Pool Information Registers and restarts to fetch a new buffer descriptor.

(7) USB Controller then DMA-transfers data from the FIFO to system memory.

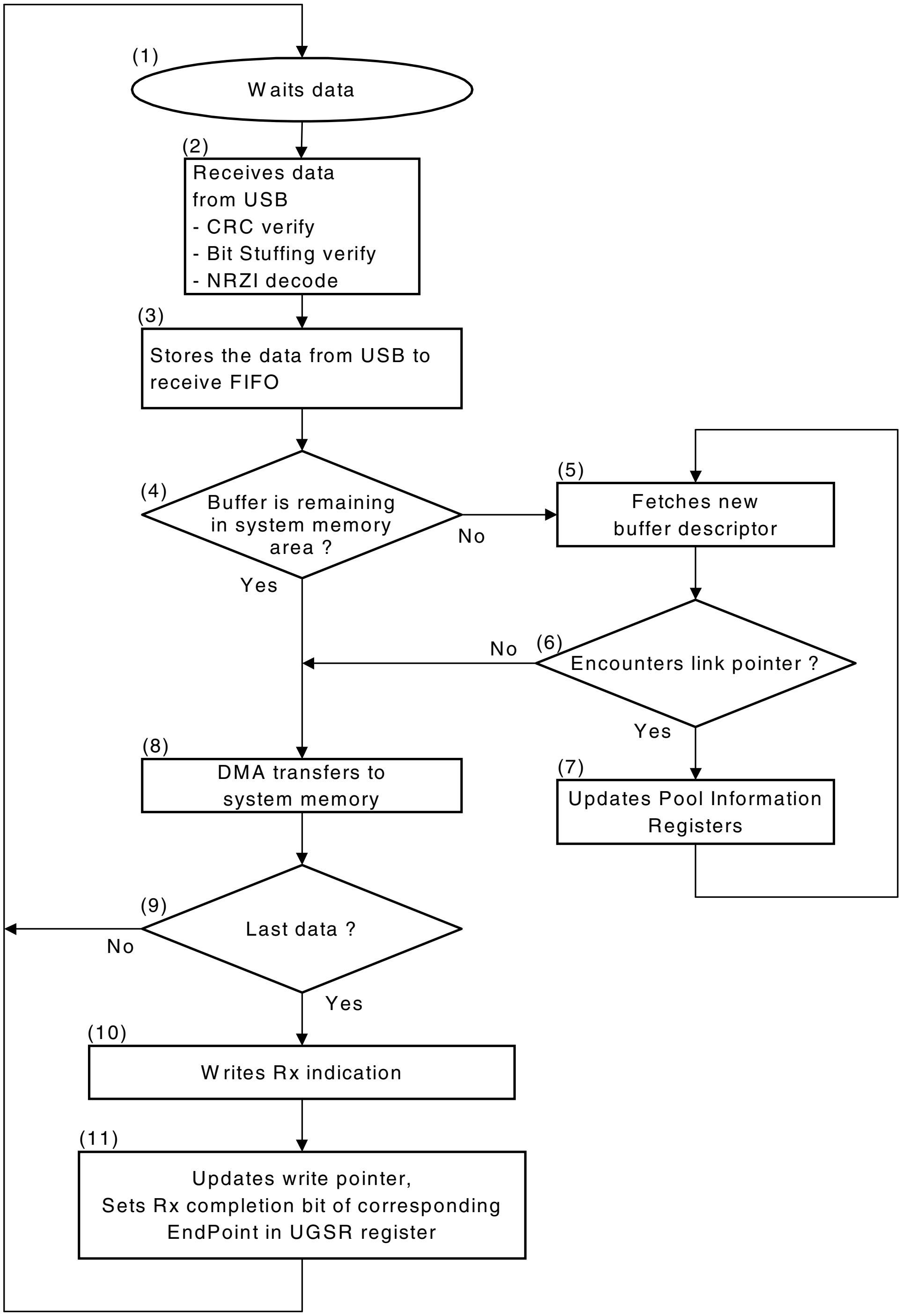
(8) If USB Controller finds that the transferred data is the last data, renews the Size field and Last bit of Buffer Descriptor and writes the Rx indication into the prepared Mailbox.

(9) USB Controller updates the write pointer of the MailBox (Rx MailBox Write Address Register Address: 1000_108CH). Also, it sets the receive completion bit of the USB General Status Register 1 and issues an interrupt to the VR4120A if it is not masked.

6.6.6.2 Assemble mode

The following figure illustrates the receive operations performed by USB Controller in Assemble Mode.

Figure 6-22. USB Controller Receive Operations (Assemble Mode)


(1)
Waits data
(2)
Receives data from USB
- CRC verify
- Bit Stuffing verify
- NRZI decode
(3)
Stores the data from USB to receive FIFO
(4)
Buffer is remaining in system memory area ?
No
Yes
(5)
Fetches new buffer descriptor
(6)
Encounters link pointer ?
No
Yes
(7)
Updates Pool Information Registers
(8)
DMA transfers to system memory
(9)
Last data ?
No
Yes
(10)
Writes Rx indication
(11)
Updates write pointer,
Sets Rx completion bit of corresponding EndPoint in UGSR register

Numbers (1) to (11) do not indicate the order in which USB Controller must perform processing. Instead, these numbers correspond to those in the following explanation.

(1) USB Controller is in the status where it waits to receive data (USB Packets) from the USB.

(2) USB Controller receives data (USB Packets) from the USB. As it is receiving the data, USB Controller performs NRZI decoding, CRC check, and Bit Stuffing Error check.

(3) USB Controller stores the received data into the FIFO.

(4) USB Controller checks whether there is buffer remaining in system memory area or not (checks if USB Controller should fetch new buffer descriptor or not).

(5) If the buffer is not remaining in system memory area, USB Controller starts to fetch new buffer descriptor.

(6) USB Controller checks whether the fetched buffer descriptor is a link pointer or not.

(7) If the fetched buffer descriptor is a link pointer, USB Controller updates the Pool Information Registers and restarts to fetch a new buffer descriptor.

(8) USB Controller then DMA-transfers data from the FIFO to system memory.

(9) USB Controller checks whether the DMA-transferred data is the last data of a segment.

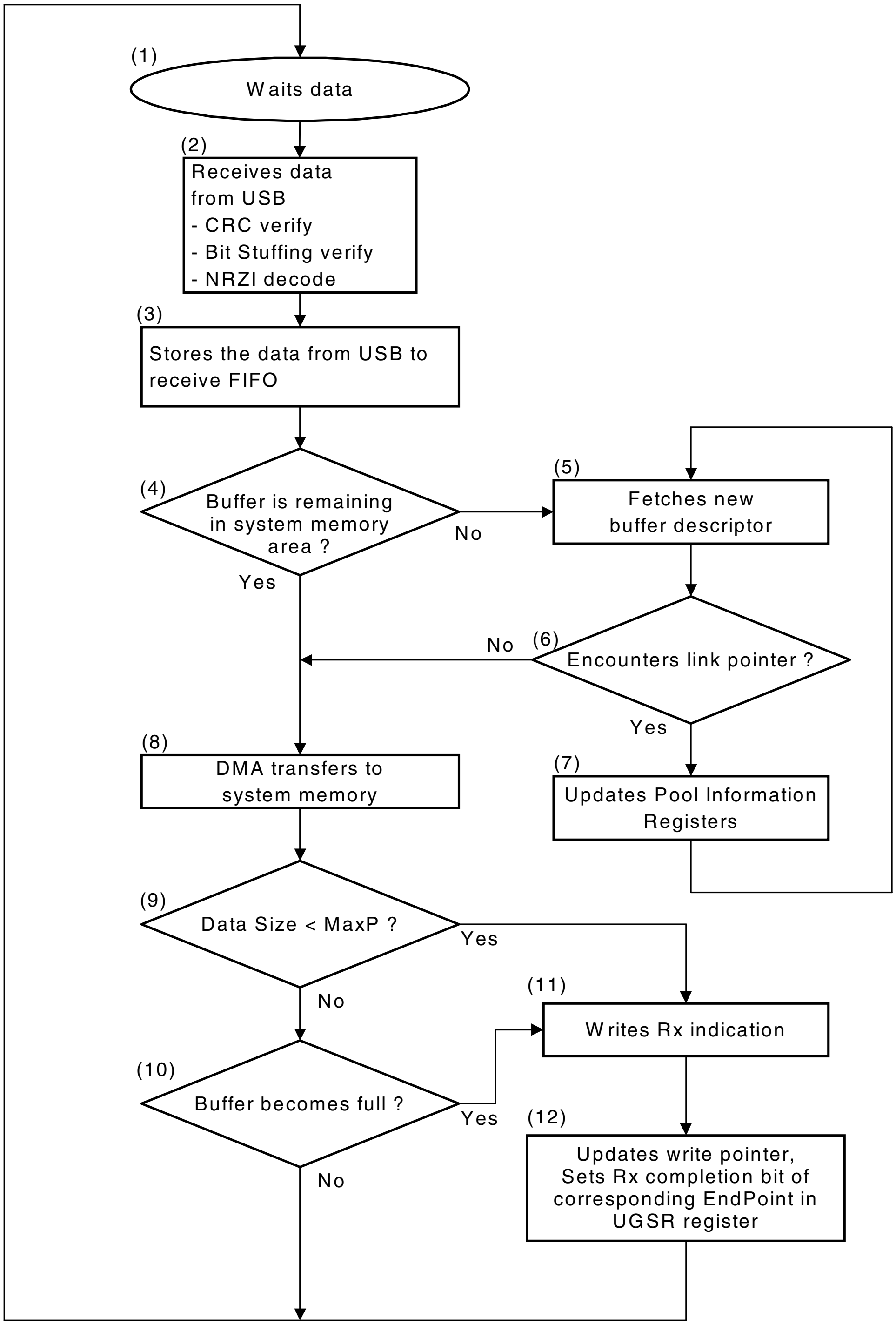
(10) If USB Controller finds that the transferred data is the last data, updates the Size field and Last bit of Buffer Descriptor and writes the Rx indication into the prepared Mailbox.

(11) USB Controller updates the write pointer of the MailBox (Rx MailBox Write Address Register Address: 1000_108CH). Also, it sets the receive completion bit of the USB General Status Register 1 and issues an interrupt to the V$_R$4120A if it is not masked.

6.6.6.3 Separate mode

The following figure illustrates the receive operations performed by USB Controller in Separate Mode.

Figure 6-23. USB Controller Receive Operation Sequence (Separate Mode)


(1)
Waits data
(2)
Receives data from USB
- CRC verify
- Bit Stuffing verify
- NRZI decode
(3)
Stores the data from USB to receive FIFO
(4)
Buffer is remaining in system memory area ?
No
Yes
(5)
Fetches new buffer descriptor
(6)
Encounters link pointer ?
No
Yes
(7)
Updates Pool Information Registers
(8)
DMA transfers to system memory
(9)
Data Size < MaxP ?
Yes
No
(10)
Buffer becomes full ?
Yes
No
(11)
Writes Rx indication
(12)
Updates write pointer, Sets Rx completion bit of corresponding EndPoint in UGSR register

Numbers (1) to (12) do not indicate the order in which USB Controller must perform processing. Instead, these numbers correspond to those in the following explanation.

(1) USB Controller is in the status where it waits to receive data (USB Packets) from the USB.

(2) USB Controller receives data (USB Packets) from the USB. As it is receiving the data, USB Controller performs NRZI decoding, CRC check, and Bit Stuffing Error check.

(3) USB Controller stores the received data into the FIFO.

(4) USB Controller checks whether there is buffer remaining in system memory area or not (checks if USB Controller should fetch new buffer descriptor or not).

(5) If the buffer is not remaining in system memory area, USB Controller starts to fetch new buffer descriptor.

(6) USB Controller checks whether the fetched buffer descriptor is a link pointer or not.

(7) If the fetched buffer descriptor is a link pointer, USB Controller updates the Pool Information Registers and restarts to fetch a new buffer descriptor.

(8) USB Controller then DMA-transfers data from the FIFO to system memory.

(9) USB Controller checks whether the DMA-transferred data is less than Max Packet Size.

(10) USB Controller checks whether buffer area becomes full or not.

(11) USB Controller updates the Size field and Last bit of Buffer Descriptor and writes the Rx indication into the prepared Mailbox.

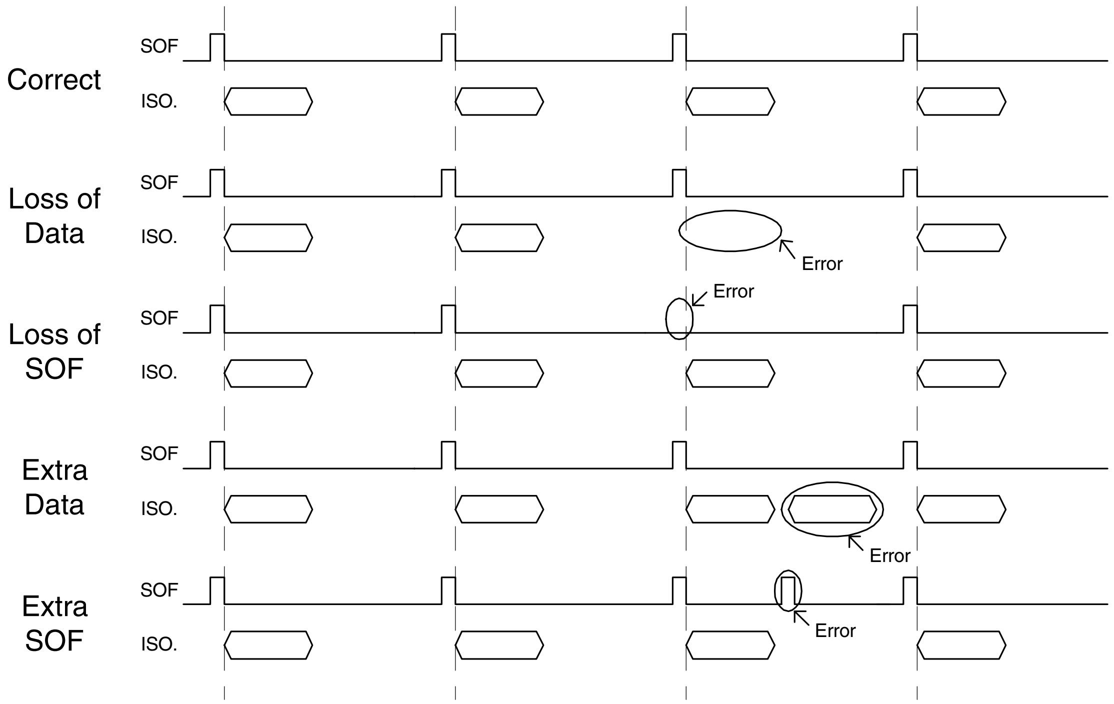
(12) USB Controller updates the write pointer of the MailBox (Rx MailBox Write Address Register Address: 1000_108CH). Also, it sets the receive completion bit of the USB General Status Register 1 and issues an interrupt to the VR4120A if it is not masked.

6.6.7 Detection of errors on USB

USB Controller has some functions which detect some errors on the USB.

Errors shown in figure below are related to Isochronous EndPoint and SOF packet.

Figure 6-24. USB Timing Errors



(1) If “Loss of Data” error has occurred, EP2ND bit (Bit 5) in USB General Status Register 2 will be set. The other action of USB Controller for this error is explained in next section (Section **6.6.8**).

(2) If “Loss of SOF” error has occurred, SL bit (Bit 0) in USB General Status Register 2 will be set.
In this case, USB Controller only reflect the error to USB General Status Register.

(3) If “Extra Data” error has occurred, EP2ED bit (Bit 6) in USB General Status Register 2 will be set.
In this case, USB Controller only reflect the error to USB General Status Register.

(4) If “Extra SOF” error has occurred, ES bit (Bit 1) in USB General Status Register 2 will be set.
In this case, USB Controller only reflect the error to USB General Status Register.

USB Controller can detect the other Error listed below.

- Isochronous data oversize error: If received data packet size is over Max Packet Size of EndPoint2, USB Controller will set EP2OS bit (Bit 7) in USB General Status Register 2.

- Incorrect EndPoint Number: If received IN/OUT TOKEN packet includes the EndPoint Number which is not enabled by V_R4120A or which is over 7, USB Controller will set IEA bit (Bit 19) in USB General Status Register 2.

- No data in EndPoint1 Tx FIFO: If IN TOKEN packet for EndPoint2 comes when Tx FIFO for EndPoint2 is not ready, USB Controller will not transmit any data to USB and will set EP1ND bit (Bit 2) in USB General Status Register 2.

- Extra Token on EndPoint1: If IN TOKEN packet for EndPoint2 comes which between two SOFs, USB Controller will set EP1ET bit (Bit 3) in USB General Status Register 2. In this case, USB Controller will transmit data only once.

- No Token on EndPoint1: If IN TOKEN packet for EndPoint2 does not come between two SOFs, USB Controller will set EP1NT bit (Bit 4) in USB General Status Register 2.

6.6.8 Rx data corruption on Isochronous EndPoint

On Isochronous Rx EndPoint (EP2), one data packet comes per one frame.

If any Isochronous data packet doesn't come between two SOF packet, it is assumed that Isochronous data is corrupted.

In the case of corruption, action of USB Controller varies according to Rx Mode.

(a) Rx normal mode

USB Controller sets EP2ND (EndPoint2 No Data) bit (Bit 6) in USB General Status Register 2.

USB Controller doesn't write any Rx indications.

USB Controller doesn't write any data to Data Buffer on system memory.

(b) Rx assemble mode

USB Controller sets EP2ND (EndPoint2 No Data) bit (Bit 6) in USB General Status Register 2.

USB Controller writes dummy data to Data Buffer (In fact, USB Controller only increment pointer which addresses Data Buffer by Max Packet Size. No DMA transfer occurs).

USB Controller writes Rx indications which indicates that received data is corrupted on Isochronous EndPoint.

(c) Rx separate mode

USB Controller sets EP2ND (EndPoint2 No Data) bit (Bit 6) in USB General Status Register 2.

USB Controller writes dummy data to Data Buffer (In fact, USB Controller only increment pointer which addresses Data Buffer by Max Packet Size. No DMA transfer occurs).

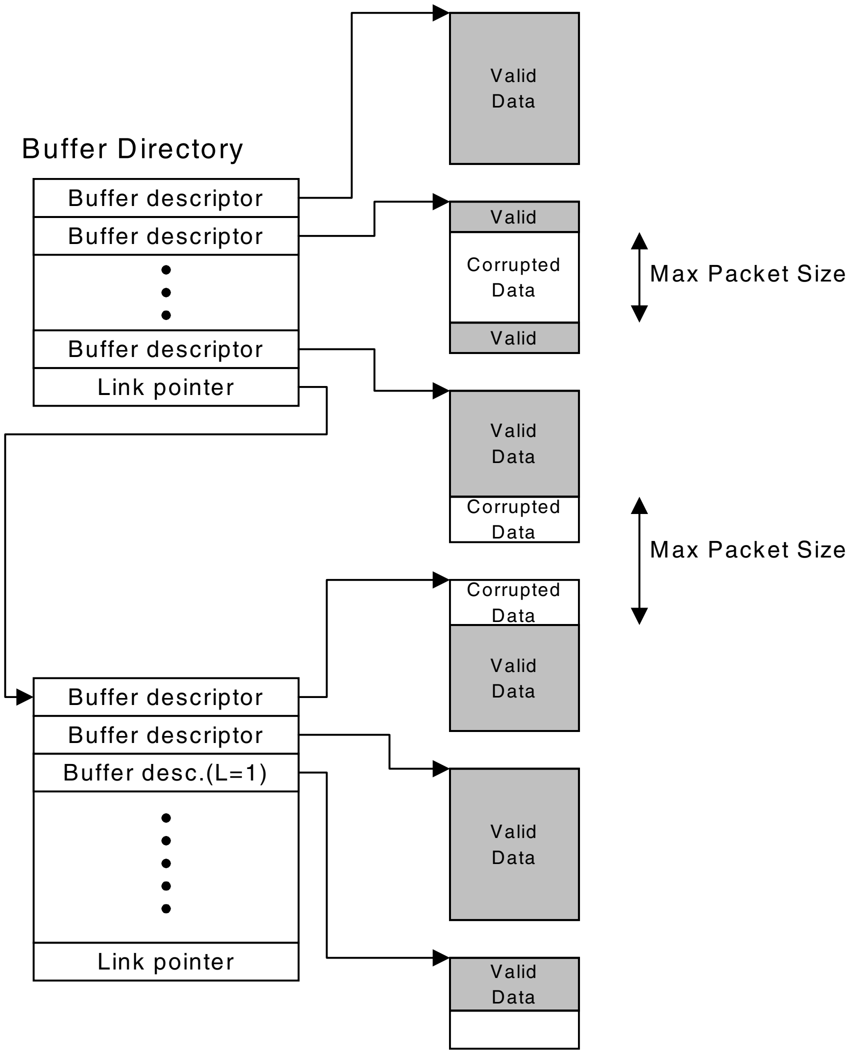
USB Controller writes Rx indications which indicates that received data is corrupted on Isochronous EndPoint.

Example

When USB Controller receives USB Packet in Rx Assemble Mode or Rx Separate Mode, if data corruption occurs on Isochronous Rx EndPoint (EndPoint4), Data Buffers becomes as **Figure 6-25**.

Shaded area in following figure is filled by valid data. If USB Controller detects that data is corrupted, next data must be stored in Data Buffer which keeps area for corrupted data. Corrupted data area is equal to Max Packet Size of Isochronous Rx EndPoint (EndPoint2).

Figure 6-25. Example of Buffers Including Corrupted Data



6.6.9 Rx FIFO overrun

On Isochronous Rx EndPoint (EP2), if data reading from Rx FIFO is delayed by some problem, Rx FIFO Overrun will occur.

In the case of corruption, action of USB Controller varies according to Rx Mode.

(a) Rx normal mode

USB Controller sets EP2FO (EndPoint2 No Data) bit (Bit 9) in USB General Status Register 2.

USB Controller writes the Rx indications which indicates that EP2 FIFO Overrun has occurred.

USB Controller doesn’t write any dummy data to Data Buffer on system memory.

(b) Rx assemble mode

USB Controller sets EP2FO (EndPoint2 No Data) bit (Bit 9) in USB General Status Register 2.

USB Controller writes dummy data to Data Buffer (In fact, USB Controller only increment pointer which addresses Data Buffer by Max Packet Size. No DMA transfer occurs) so that the sum of received data and dummy data becomes equal to Max Packet Size.

After USB Controller receives “USB short packet”, USB Controller writes Rx indications which indicates that EP2 FIFO Overrun has occurred.

(c) Rx separate mode

USB Controller sets EP2FO (EndPoint2 No Data) bit (Bit 9) in USB General Status Register 2.

USB Controller writes dummy data to Data Buffer (In fact, USB Controller only increment pointer which addresses Data Buffer by Max Packet Size. No DMA transfer occurs) so that the sum of received data and dummy data becomes equal to Max Packet Size.

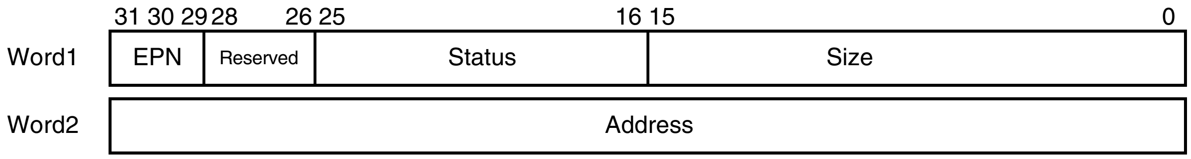
USB Controller writes Rx indications which indicates that EP2 FIFO Overrun has occurred.

6.6.10 Rx indication

For every data segment that it receives, USB Controller writes a receive indication into the receive MailBox. After writing a receive indication, USB Controller sets the receive completion bit of the USB General Status Register to a ‘1’ and, issues an interrupt if it is not masked.

The format of the receive indication is as shown below.

Figure 6-26. Receive Indication Format



Remark Bit28 to Bit26 are reserved.

EPN: Field that indicates the EndPoint number.

001: EndPoint0
011: EndPoint2
101: EndPoint4
111: EndPoint6

Status: Field that indicates the status upon data receiving.

Bit25: When set to a ‘0’, indicates that a data corruption did not occur on EndPoint2.
When set to a ‘1’, indicates that a data corruption occurred on EndPoint2.

Bit24: When set to a ‘0’, indicates that an internal bus error has not occurred.
When set to a ‘1’, indicates that processing terminated abnormally due to an internal bus error.
When this bit is set to 1, the Address field has no meaning.

Bit23: When set to a ‘0’, indicates that the received data is other than a USB Setup packet.
When set to a ‘1’, indicates that the received data is a USB Setup packet.
This bit is set only when receiving the data from the Control EndPoint (EndPoint0). If data is received from any other EndPoint, this bit is not set.

Bit22: When set to a ‘0’, indicates that a buffer overrun did not occur.

When set to a ‘1’, indicates that a buffer overrun occurred.
This bit is set only when receiving the data from the EndPoint1.

Bit21: Reserved.

Bit20: When set to a ‘0’, indicates that a CRC error has not occurred.
When set to a ‘1’, indicates that a CRC error has occurred.
When this bit is set to 1 when receiving data from the Isochronous EndPoint (EndPoint2), it indicates that the data stored in system memory includes a CRC error.
This bit is set only when receiving the data from the EndPoint2.
In the assemble mode, this bit is set only the following case; the USB packet which is received at last has error.

Bit19: When set to a ‘0’, indicates that a Bit Stuffing Error has not occurred.
When set to a ‘1’, indicates that a Bit Stuffing Error has occurred.
When this bit is set to 1 when receiving data from the Isochronous EndPoint (EndPoint2), it indicates that the data stored in system memory contains a Bit Stuffing Error.
This bit is set only when receiving the data from the EndPoint2.
In the assemble mode, this bit is set only the following case; the USB packet which is received at last has error.

Bit18: When set to a ‘0’, indicates that the size of the received data is up to 65535 bytes.
When set to a ‘1’, indicates that the size of the received data is greater than 65535 bytes.

Bit17-16: When set to 00 or 01, indicates that data is received in Normal Mode.
When set to 10, indicates that data is received in Assemble Mode.
When set to 11, indicates that data is received in Separate Mode.
When using EndPoint0 and EndPoint6, this field should be set to 00.

Size: Indicates the size of the received data.
When the size of the received data exceeds 65535 (FFFFH) bytes, Bit18 is set to 1, and this field will contain 65535 (FFFFH).

Address: Indicates the start address of the buffer into which the received data is stored.

6.7 Power Management

USB Controller has a built in feature that allows it to use interrupts to inform the VR4120A of its having received Suspend or Resume signaling from a Host PC. When the VR4120A receives a Suspend or a Resume, it must perform the appropriate processing.

Also, for those instances when the port to which the μPD98502 is connected is in the Suspend status (the μPD98502 is in the Suspend status), USB Controller has a function for issuing Remote Wake Up signaling to switch the Suspend status to Resume.

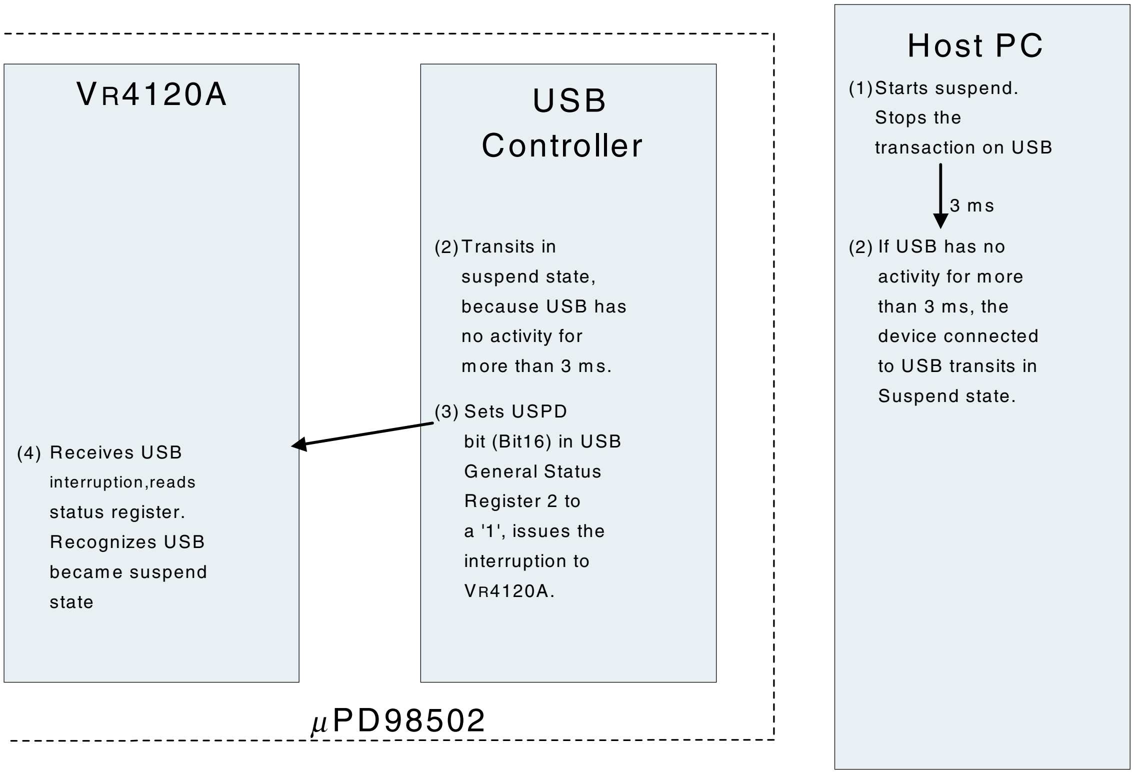
As a result, even if the μPD98502 is in the Suspend status, data that arrives from the line (ADSL) is not discarded but can be passed to a Host PC.

The following sections explain each of the sequences.

6.7.1 Suspend

The Suspend sequence is as shown below.

Figure 6-27. Suspend Sequence


VR4120A
(4) Receives USB interruption,reads status register. Recognizes USB became suspend state
USB Controller
(2) Transits in suspend state, because USB has no activity for more than 3 ms.
(3) Sets USPD bit (Bit16) in USB General Status Register 2 to a '1', issues the interruption to VR4120A.
μPD98502
Host PC
(1) Starts suspend. Stops the transaction on USB
3 ms
(2) If USB has no activity for more than 3 ms, the device connected to USB transits in Suspend state.

(1) The host places the USB in the Suspend status. Traffic stops flowing through the USB.

(2) After 3 ms there is no traffic through the USB. Therefore, all of the devices connected to the USB shift to the Suspend status. In the same way, USB Controller also enters the Suspend status. During DMA transfer is being performed, however, USB Controller does not enter the Suspend status until after the completion of DMA transfer.

(3) USB Controller sets the USPD bit (Bit16) of the USB General Status Register 2 (Address: 1000_1018H) to a ‘1’, then issues an interrupt to the VR4120A if interrupt is not masked.

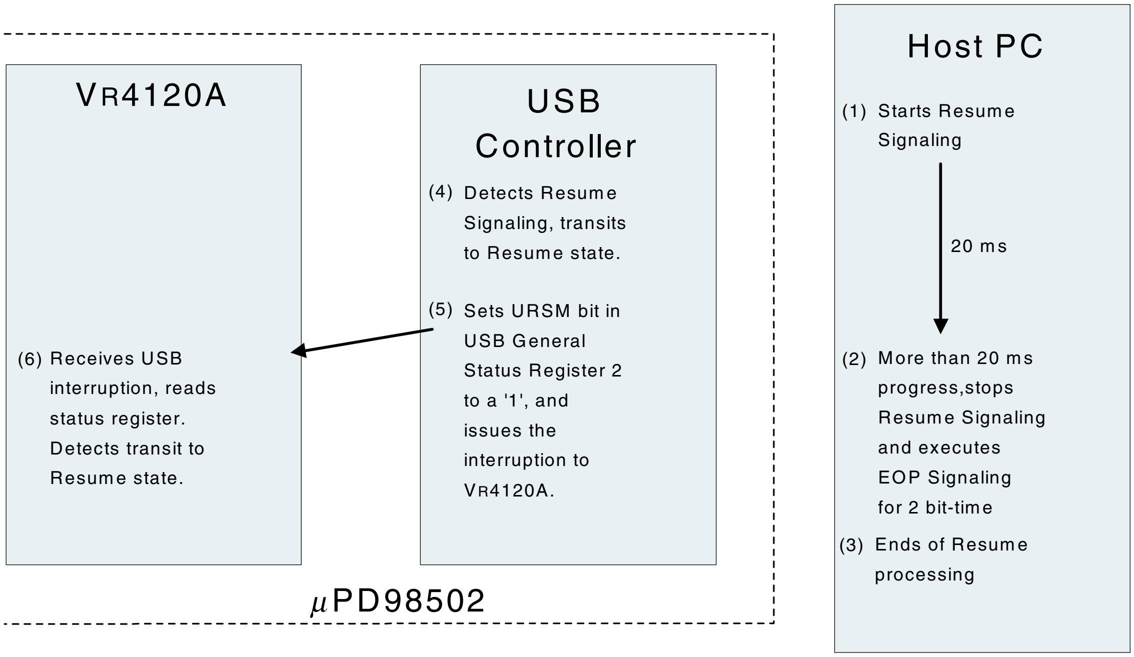
(4) The VR4120A receives the interrupt from USB Controller, reads the USB General Status Register 2 and, as a result, determines that the USB is in the Suspend status.

The VR4120A is not permitted to write to other than USB Controller's USB General Mode Register and USB Interrupt Mask Register 2 while USB Controller is in the Suspend status. Otherwise, after USB Controller enters the Resume status, its operation will be unpredictable.

6.7.2 Resume

The Resume sequence is shown below.

Figure 6-28. Resume Sequence


VR4120A
(6) Receives USB interruption, reads status register. Detects transit to Resume state.
USB Controller
(4) Detects Resume Signaling, transits to Resume state.
(5) Sets URSM bit in USB General Status Register 2 to a '1', and issues the interruption to VR4120A.
μPD98502
Host PC
(1) Starts Resume Signaling
20 ms
(2) More than 20 ms progress,stops Resume Signaling and executes EOP Signaling for 2 bit-time
(3) Ends of Resume processing

(1) The Host PC starts Resume Signaling. The Resume Signaling is passed to every device connected to the USB.

(2) After at least 20 ms have elapsed, the Host PC stops the Resume Signaling then performs EOP Signaling for a 2bit-time duration.

(3) This causes the Host PC to terminate its Resume processing.

(4) USB Controller receives the Resume Signaling.

(5) USB Controller sets the URSM bit (Bit 18) of the USB General Status Register 2 (Address: 1000_1018H) to a ‘1’, then issues an interruption to the VR4120A. If clock supplement is stopped, VR4120A has to check “usbwakeup_p” signal instead of “URSM” bit.

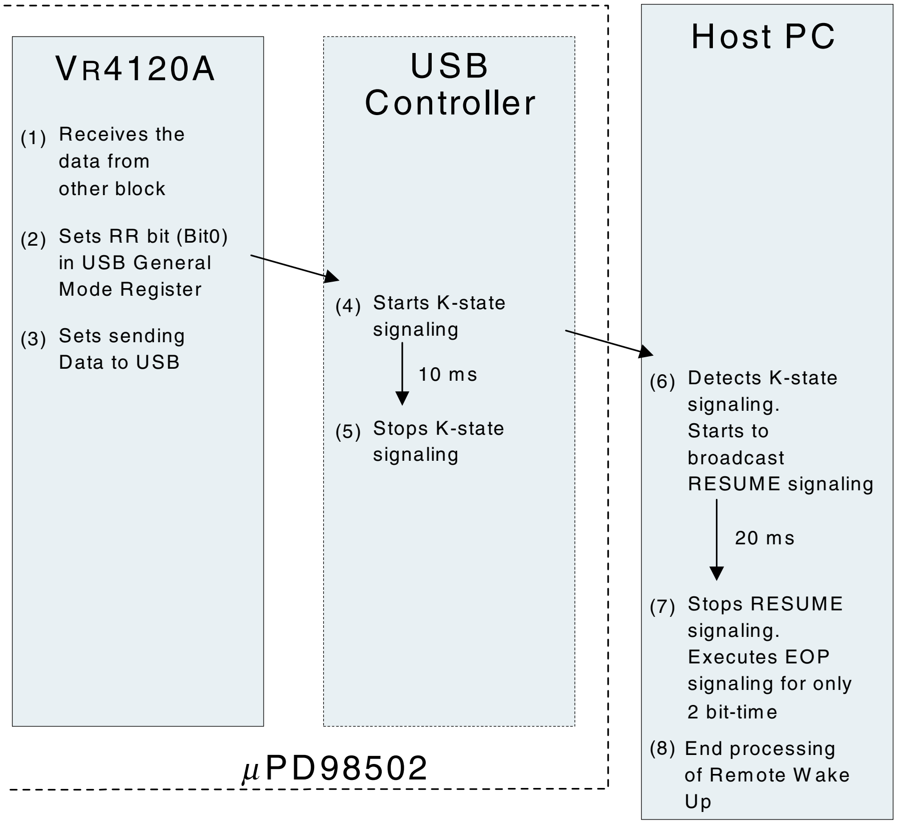
(6) The VR4120A receives the interruption from USB Controller and, as a result, determines that the USB has entered the Resume status.

After USB Controller enters the Resume status, it continues with the transmit/receive processing it was performing immediately before it entered the Suspend status.

6.7.3 Remote wake up

The Remote Wake Up sequence is shown below.

Figure 6-29. Remote Wake Up Sequence


VR4120A
(1) Receives the data from other block
(2) Sets RR bit (Bit0) in USB General Mode Register
(3) Sets sending Data to USB
USB Controller
(4) Starts K-state signaling
10 ms
(5) Stops K-state signaling
Host PC
(6) Detects K-state signaling. Starts to broadcast RESUME signaling
20 ms
(7) Stops RESUME signaling. Executes EOP signaling for only 2 bit-time
(8) End processing of Remote Wake Up
μPD98502

(1) Here, it is assumed that the USB is in the Suspend status. Data is received from other block.

(2) The VR4120A sets the RR bit (Bit 0) of the USB General Mode Register in order to switch the USB in the Suspend status to the Resume status.

(3) Once the RR bit of the USB General Mode Register has been set, USB Controller starts K-state Signaling for the USB.

(4) The VR4120A can continue to set transmit data for the USB. Specifically, the VR4120A prepares transmit data in system memory, then writes data into the USB Command Register (Address: 1000_1040H) and the USB Command Extension Register (Address: 1000_1044H).

(5) USB Controller continues K-state Signaling for 5 ms, then terminates the signaling.

(6) The Host PC, upon receiving the K-state Signaling, broadcasts RESUME signaling. This RESUME signaling continues for a minimum of 20 ms.

(7) Once at least 20 ms have elapsed, the Host PC terminates RESUME Signaling, then issues EOP Signaling for a 2 bit-time duration.

(8) As a result of this sequence, Remote Wake Up is terminated, and the transaction being performed by the USB is restarted.

6.8 Receiving SOF Packet

USB Controller can receive SOF Packets, and check if Frame Number is incremented correctly.

In addition, USB Controller can detect the timing skew of SOF Packet.

6.8.1 Receiving SOF Packet and updating the Frame Number

After USB Controller receives a SOF Packet, FN field in USB Frame Number/Version Register (Address: 1000_1004H) is updated. After FN field is updated, FW bit (Bit 21) in USB General Status Register 2 (Address: 1000_1018H) is set to a ‘1’.

6.8.2 Updating Frame Number automatically

If received SOF Packet has incorrect Frame Number, USB Controller can execute one of two processes shown below.

- USB Controller reflects the incorrect Frame Number to FN field directly.
- USB Controller increments Frame Number automatically and write the incremented value to FN field.

The policy of updating FN field is shown in following table:

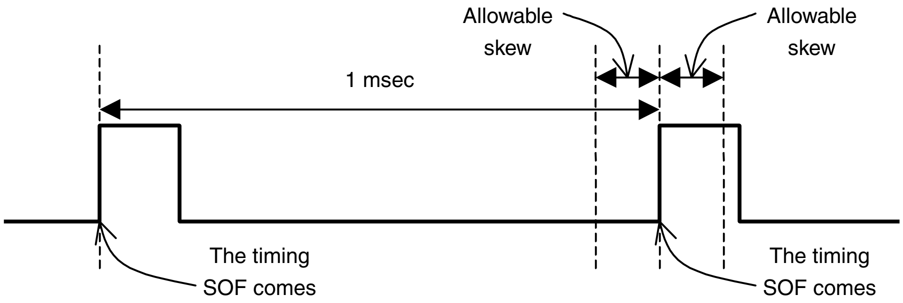
Case	AU bit [Note]	FN field	USB General Status Register 2
Correct SOF	0	Just load the received Frame Number	FW bit is set to 1
	1	Just load the received Frame Number	FW bit is set to 1
Loss of SOF	0	Not updated	SL bit is set to 1
	1	Increment the current FN	FW bit and SL bit are set to 1
Extra SOF	0	Not updated	ES bit is set to 1
	1	Not updated	ES bit is set to 1
Bit Stuff Error	0	Just load the received Frame Number	FW bit is set to 1
	1	Increment the current FN	FW bit is set to 1
CRC Error	0	Just load the received Frame Number	FW bit is set to 1
	1	Increment the current FN	FW bit is set to 1
Incorrect Frame Number	0	Just load the received Frame Number	FW bit is set to 1
	1	Increment the current FN	FW bit is set to 1

Note AU bit is in the USB General Mode Register (Address: 1000_1000H).

6.8.3 Checking if the skew of SOF arrival time is allowable of not

The allowable SOF skew can be defined by SOFINTVL field in the USB General Mode Register (Address: 1000_1000H).

Figure 6-30. Allowable Skew for SOF



SOFINTVL field is set to 18H (24 clocks) at default. This is 0.05 % of 48000 clocks (1 msec).

The value of SOFINTVL should be set before the first SOF packet comes.

6.9 Loopback Mode

USB Controller features a built-in loopback function for test purposes.

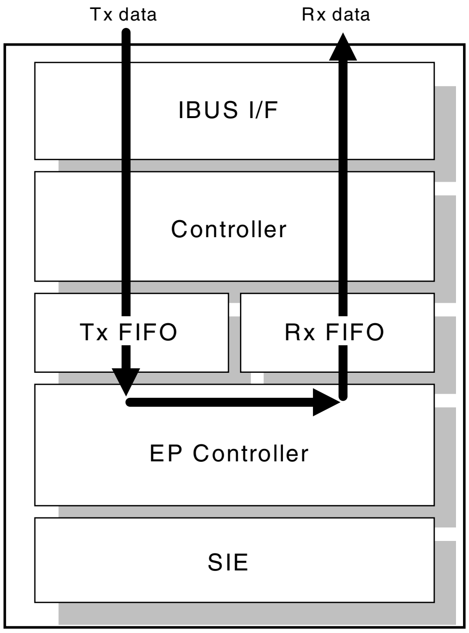
To enable the loopback function, set the LE bit (Bit 1) of the USB General Mode Register to 1.

Once the loopback function has been activated, USB Controller gets the data from system memory and places it into the Tx FIFO. The data is returned by the EndPoint Controller. The returned data is written into the Rx FIFO, after it is returned to system memory.

Transmitting and receiving should be performed using the normal settings. The Tx and Rx indications are issued as normal.

The internal data flow is as shown below.

Figure 6-31. Data Flow in Loopback Mode



As shown in the figure, in loopback mode data reception from the USB and data transmission to the USB are not performed. All data is returned by the EndPoint Controller.

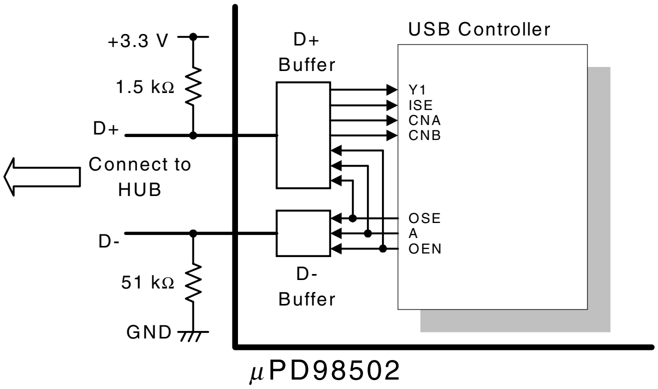
Following table indicates the Endpoint which is used to transmit data and the Endpoint at which the data will be received.

Tx EndPoint	Rx EndPoint
EndPoint0 (Control)	EndPoint0 (Control)
EndPoint1 (Isochronous)	EndPoint2 (Isochronous)
EndPoint3 (Bulk)	EndPoint4 (Bulk)
EndPoint5 (Interrupt)	EndPoint6 (Interrupt)

6.10 Example of Connection

USB Controller is connected to the μPD98502 internal USB I/O buffer as shown in the following **Figure 6-32**.

Figure 6-32. Example of Connection



When designing a PCB, it is necessary to connect a 1.5 kΩ pull-up resistor between the D+ pin and the 3.3 V power supply to indicate the presence of a full-speed device.

To avoid current floating on the integrated USB Buffer it is recommended to place a 51 kΩ pull-down resistor between the D- pin and the GND.

The circuit must be designed such that the μPD98502 power supply is turned on and off together with the external 3.3 V power supply. If the μPD98502 power supply is off, but the external 3.3 V power supply is on, the USB HUB connected to the μPD98502 will assume that a new device has been connected but, because the μPD98502 power is off, no response can be returned.

For details of the electrical specifications of the USB, refer to USB Specification 1.1.

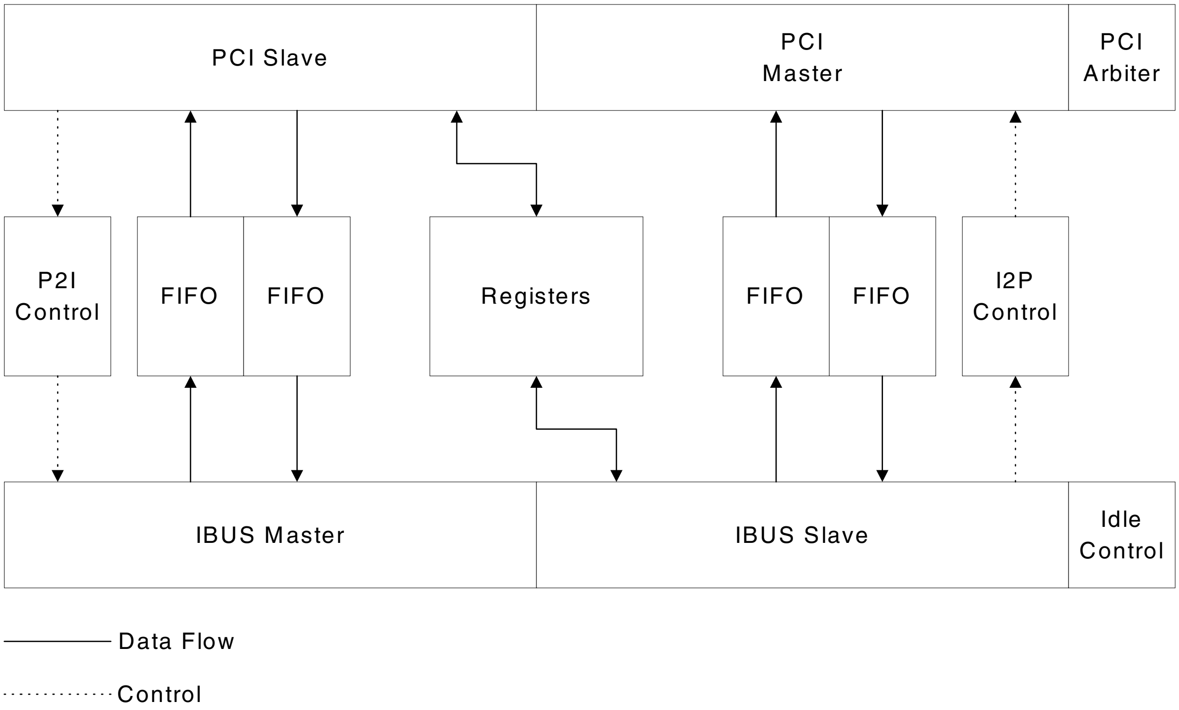
CHAPTER 7 PCI CONTROLLER

7.1 Overview

The PCI Controller supports both NIC mode and Host mode. With the NIC mode, the PCI Controller does not issue configuration cycle and the arbitration function is not enabled. With the Host mode, the PCI Controller can issue configuration cycle and the arbitration function is enabled. Futures of the PCI Controller are as follows.

- 32-bit PCI Interface
- 33-MHz PCI frequency capable
- Compliant to PCI Local Bus Specification Rev.2.2
- Compliant to PCI bus Power Management Interface Rev.1.1
- Supports up to 16-word burst access
- Supports posted write function
- Supports Delayed/Non Delayed read/write cycle
- Implements PCI bus arbiter that supports up to 4 external PCI-master devices at Host-mode

Figure 7-1. The PCI Controller Block Diagram



7.2 Bus Bridge Functions

7.2.1 Internal bus to PCI transaction

7.2.1.1 Window size

The PCI Controller can have a 2-MB length access window in internal memory space. The VR4120A can access external PCI devices through the access window. The access window can be positioned in the memory range from 1020_0000H to 103F_FFFFH. The base address of the PCI address space is defined by setting of P_PLBA Register.

7.2.1.2 PCI master

The PCI Controller can issue memory commands only. I/O commands, interrupt-acknowledge command, and special-cycle command are not supported. The PCI Controller can generates Configuration-Cycle in Host-mode.

In the case that Cache-Line-Size register is valid (which means Cache-Line-Size is 4 or 8 or 16 or 32), the PCI Controller uses 3 kinds of PCI-memory commands for read transactions in accordance with the recommendation in PCI Specification.

Memory Read: The case of reading a single 32 bit-word.
Memory Read Line: The case of reading more than a 32 bit-word up to the next cache-line boundary.
Memory Read Multiple: The case of reading a block that crosses a cache-line boundary of data.

When Cache-Line-Size register is not valid, the PCI Controller always issues Memory Read Command.

The PCI Controller uses Memory-Write-and-Invalidate commands for write transaction, when all the conditions as below are satisfied.

- The number of transfer words is just the multiple as the size of cache line.
- “Memory-Write-and-Invalidate” bit in configuration register is set to ‘1’.
- The start address of the write transaction is at a cache boundary.

7.2.1.3 Write issue from internal bus to PCI

(1) Posted write transaction

If IPWRD bit in P_BCNT register is ‘0’, the PCI Controller uses “Posted Write Transaction” rule for write transactions from the internal bus-side to PCI-side. The rule is as follows;

<1> An internal bus block[Note] connecting to the internal bus issues the write transaction to PCI target device.

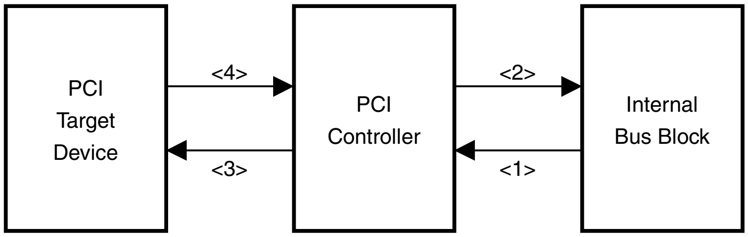
<2> The PCI Controller accepts this access and puts the data to be written into the internal FIFO. The transaction on internal bus is completed at this moment. Then, the PCI Controller will issues “retry” to all of later write access on internal bus to external PCI devices until the write transaction on PCI bus has been completed.

<3> The PCI Controller issues the write transaction to PCI target device.

<4> After the completion of the transaction, the PCI Controller can accepts the new write transaction on internal bus again.

Note Internal bus block is a block connecting to the internal bus like USB controller. All of other internal bus devices than the PCI Controller can issue access cycle to external PCI devices. The VR4120A can also issue access cycles to the PCI devices through the system controller.

Figure 7-2. Posted Write Transaction from Internal Bus to PCI



The maximum burst size is 16 words. When more than 16 words write burst is issued on Internal bus, the PCI Controller issues “disconnect” at 16th word. In the case of a burst access across the address boundary, the PCI Controller issues “disconnect”, too.

When the PCI Controller receives target abort/master abort on PCI bus after it accepts posted-write from Internal bus-side, it sets WRTAT/WRMAT bit of P_IGSR register and RTABT/RMABT bit of P_PGSR register, and issues an interrupt to PCI-Host and the VR4120A (if not masked). The data in the internal FIFO will be discarded. Then, the PCI Controller can accept the new write transaction on the internal bus.

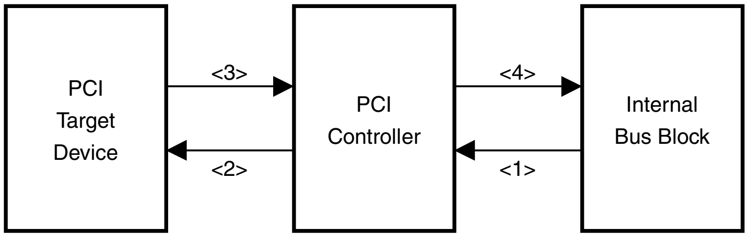
(2) Non posted write transaction

If IPWRD bit in P_BCNT register is ‘1’, the PCI Controller uses “Non Posted Write Transaction” rule for write transactions from Internal bus-side to PCI-side. In this mode, burst transfers are disconnected at every single word. The rule is as follows;

<1> An internal bus block[Note] connecting to the internal bus issues the write transaction to an external PCI target device. The PCI Controller latches the first word of the burst data. A wait time is inserted until the latched data is written to the PCI target device.
<2> The PCI Controller issues the write transaction to the external PCI target device.
<3> The PCI target device accepts the access.
<4> After the PCI transaction is completed, the PCI Controller issues “disconnect” to the internal bus block that try to continue the write access as the burst transfer. The internal bus block should terminate the transaction as soon as possible.

Note Internal bus feature is not described in this document. Internal bus has a similar feature to PCI bus. Then, internal bus supports “disconnect” function.

Figure 7-3. Non Posted Write Transaction from Internal Bus to PCI



When the internal bus block wants to transfer more words, it should issue an additional write transaction.

If the PCI Controller receives target abort/master abort on PCI bus after it accepts non-posted-write from Internal bus-side, it sets WRTAT/WRMAT bit of P_IGSR register and RTABT/RMABT bit of P_PGSR register, and issues interrupts to an external PCI-Host device and the VR4120A (if not masked). The PCI Controller will discard the data to be written. Then, the PCI Controller can accept the new write transaction from Internal bus, again.

7.2.1.4 Read issue from internal bus to PCI

(1) Delayed read transaction

When IDRTD bit in P_BCNT register is '0', the PCI Controller uses "Delayed Read Transaction" rule for read transactions from internal bus-side to PCI-side. The rule is as follows;

<1> An internal bus block connecting to the internal bus issues the read transaction to an external PCI target device.

<2> The PCI Controller responds to this access and issues "retry" to internal bus block. However, the PCI Controller latches the address and the command, and stores the access issued. Then, the PCI Controller issues "retry" to all the access until the transaction corresponding to the latched command on PCI bus has been completed.

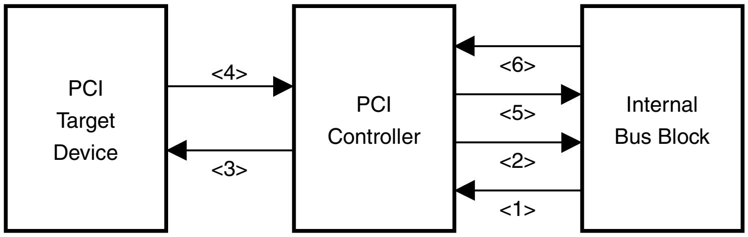
<3> The PCI Controller issues the read transaction to PCI target device.

<4> The PCI target device accepts the access. The read data from the PCI target device is stored in the internal FIFO.

<5> The PCI Controller waits that the same access with that is issued comes on the internal bus. The PCI Controller issues "retry" to other accesses.

<6> When the same access comes, which means the access with the same address and the same command, the PCI Controller accepts this access and returns the data from the internal FIFO.

Figure 7-4. Delayed Read Transaction from Internal Bus to PCI



The maximum burst size is 16 words so that when more than 16 words read burst is issued on Internal bus, the PCI Controller issues "disconnect" at 16th word. In the case of a burst access across the address boundary, it issues "disconnect", too.

When the same read access that has been issued does not come within 2^{15} clocks, the PCI Controller discards the data in the internal FIFO, sets PFDSC bit of P_PGSR register and reports to PCI-Host by interrupt (if not masked). Then, the PCI Controller can accept the new read transaction from Internal bus, again.

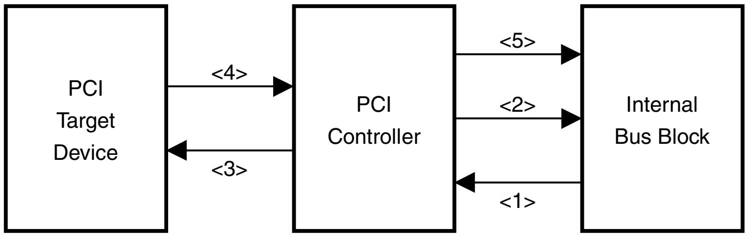
When the PCI Controller receives target abort/master abort on PCI bus after it has accepted delayed-read from PCI-side, it sets RTABT/RMABT bit of P_IGSR register and RDTAT/RDMAT bit of P_PGSR register, and reports by interrupts to an external PCI-Host device and the VR4120A (if not masked). Then, the PCI Controller returns all "0" words to internal bus block.

(2) Non delayed read transaction

When IDRTD bit in P_BCNT register is ‘1’, the PCI Controller uses “Non Delayed Read Transaction” rule for read transactions from Internal bus-side to PCI-side. In this mode, burst transfers are disconnected at every single word. The rule is as follows;

<1> An internal bus block connecting to the internal bus issues the read transaction to an external PCI target device.
<2> The PCI Controller inserts waits before the first data transfer on the internal bus.
<3> The PCI Controller issues the read transaction to the external PCI target device.
<4> The PCI target device accepts this access.
<5> The PCI Controller returns the read 1-word data to internal bus block. At the same time, the PCI Controller issues “disconnect” to internal bus block when the block tries to continue the burst read transaction. The internal bus block must terminate the transaction as soon as possible.

Figure 7-5. Non Delayed Read Transaction from Internal Bus to PCI



When the PCI Controller receives target abort/master abort on PCI bus after it has accepted non-delayed-read from an external PCI device, it sets RTABT/RMABT bit of P_IGSR register and RDTAT/RDMAT bit of P_PGSR register, and issues interrupts to an external PCI-Host device and the VR4120A (if not masked). Then, the PCI Controller returns all “0” word to internal bus block and issues “disconnect” when the internal bus block issues the burst transfer.

7.2.2 PCI to internal bus transaction

7.2.2.1 Window size

The PCI Controller supports a 2-MB address space as the access window from PCI-side to Internal bus-side in PCI memory space. The base address for the window is written to Window Memory Base Address register in configuration space by an external PCI-Host device in NIC mode. In Host-mode, the VR4120A has to write the base address to this register.

The internal registers of the PCI Controller cannot be read/written through this window even when it includes the address area of the PCI Controller. The base address for registers, which is written in Register Memory Base Address register in configuration space, has to be used in order to read/write to the internal registers of the PCI Controller.

When the issued burst transfer goes over the boundary of the address space, the PCI Controller issues “disconnect” at the boundary.

7.2.2.2 Access type

(1) PCI target

The acceptable PCI commands for the PCI Controller are as follows.

C/BE#[3:0]	PCI commands	As PCI Target
0000	Interrupt Acknowledge	Ignored
0001	Special Cycle	Ignored
0010	I/O Read	Ignored
0011	I/O Write	Ignored
0100	Reserved	-
0101	Reserved	-
0110	Memory Read	Accepted
0111	Memory Write	Accepted
1000	Reserved	-
1001	Reserved	-
1010	Configuration Read	Accepted
1011	Configuration Write	Accepted
1100	Memory Read Multiple	Accepted
1101	Dual Address Cycle	Ignored
1110	Memory Read Line	Accepted
1111	Memory Write and Invalidate	Accepted

(2) Internal bus

When ICMDS bit in P_BCNT register is ‘0’, the PCI Controller uses I/O command on Internal bus. In the case that ICMDS bit is ‘1’, the PCI Controller uses memory command on Internal bus.

7.2.2.3 Write issue from PCI to Internal bus

(1) Posted write transaction

If PPWRD bit in P_BCNT register is ‘0’, the PCI Controller uses “Posted Write Transaction” rule for write transactions from Internal bus-side to PCI-side. The rule is as follows;

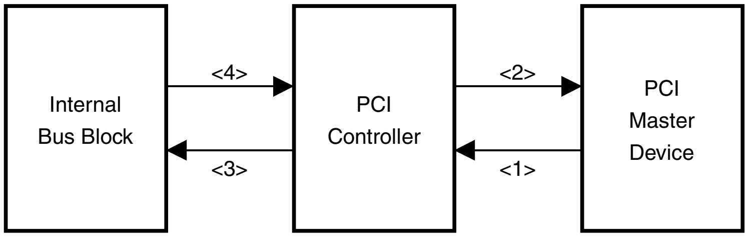
<1> A PCI master device issues the write transaction to an internal bus target block.

<2> The PCI Controller accepts this access and stores the write data into the internal FIFO. The transaction on PCI bus is completed at this moment. Then, the PCI Controller issues “retry” for all of write access to all PCI write transaction to the PCI Controller until the transaction on internal bus has been completed.

<3> The PCI Controller issues the write transaction to the internal bus target block

<4> The internal bus target block accepts this access and the PCI Controller completes the write transaction to it. After the completion of the transaction, the PCI Controller can accepts the new write access on PCI bus again.

Figure 7-6. Posted Write Transaction from PCI to Internal bus



The maximum burst size is 16 words, and when more than 16 words write burst is issued on PCI bus, the PCI Controller accepts it and issues “disconnect” at 16th word. When the PCI Controller encounters the address boundary, it issues “disconnect”, too.

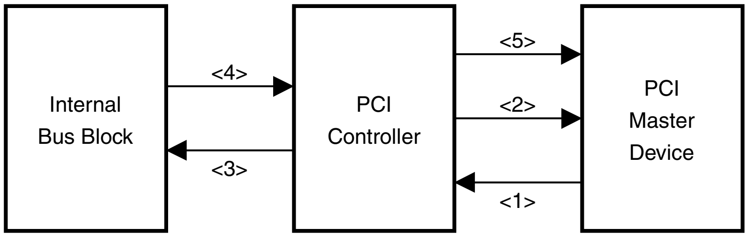
When the PCI Controller receives Bus Error on Internal bus after it has accepted posted-write from PCI-side, the PCI Controller sets IWBER bit of P_IGSR register and PWBER bit of P_PGSR register, and reports by interrupts to PCI-Host and the VR4120A (if not masked). The data in the internal FIFO will be discarded.

(2) Non posted write transaction

When PPWRD bit in P_BCNT register is ‘1’, the PCI Controller uses “Non Posted Write Transaction” rule for write transactions from Internal bus-side to PCI-side. In this mode, burst transfers are disconnected at every single word. The rule is as follows;

<1> PCI master device issues the write transaction to an internal bus target block.
<2> The PCI Controller responds to this access by asserting DEVSEL_B and latches the first word of the burst data. However, the PCI Controller does not assert TRDY_B at this moment.
<3> The PCI Controller issues the write transaction to the internal bus target block.
<4> The internal bus target block accepts this access and the PCI Controller writes the first word which is latched to it.
<5> The PCI Controller asserts TRDY_B in order to indicate that first data phase is completed. Then, the PCI Controller issues “disconnect” to PCI master device when it issues the burst transfer. PCI master device should terminates the transaction as soon as possible.

Figure 7-7. Non Posted Write Transaction from PCI to Internal bus



When the PCI Controller receives Bus Error on Internal bus, it completes the first data phase by asserting TRDY_B and issues “disconnect” to PCI master device if it issues the burst transfer. Then, the PCI Controller sets IWBER bit of P_IGSR register and PWBER bit of P_PGSR register and issues interrupts to an external PCI-Host device and the VR4120A (if not masked). The data in the internal FIFO will be discarded.

7.2.2.4 Read issue from PCI to internal bus

(1) Delayed read transaction

When PDRTD bit in P_BCNT register is ‘0’, the PCI Controller uses “Delayed Read Transaction” rule for read transactions from Internal bus-side to PCI-side. The rule is as follows;

<1> A PCI master device issues the read transaction to an internal bus target block.

<2> The PCI Controller responds to this access and issues “retry” to PCI master device. However, the PCI Controller latches the address and the command, and remembers the access issued. Then, the PCI Controller issues “retry” to all read access until the transaction on internal bus has been completed.

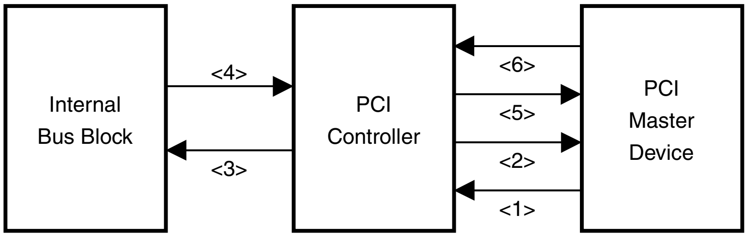
<3> The PCI Controller issues the read transaction to internal bus target block

<4> The internal bus target block accepts this access and the PCI Controller reads data into the internal FIFO from the internal bus target block.

<5> The PCI Controller waits that the same access that is issued comes on PCI bus. To other accesses, the PCI Controller issues “retry”.

<6> When the same access comes, which means the access with the same address and the same command, the PCI Controller accepts this access and returns the data in the internal FIFO.

Figure 7-8. Delayed Read Transaction from PCI to Internal bus



As the issued burst size cannot be known until the transaction is completed, the PCI Controller decides the prefetched word size based on the issued PCI command when Cache-Line-Size is valid. When Memory-Read command is used, the prefetched word size is 1 word. When Memory-Read-Line command, the size is same as Cache-Line-Size, when Memory-Read-Multiple command, the prefetched size is 16 words. When Cache-Line-Size is not valid, the prefetched size is always 16 words. When the PCI master device issues more words, the PCI Controller terminates the transaction by “disconnect” after it returns the prefetched data to PCI master device. When the PCI Controller encounters the address boundary, it issues “disconnect”, too.

If the same access with that has been issued does not come within 2^{15} clocks, the PCI Controller discards the data in the internal FIFO, sets IFDSC bit of P_IGSR register and issues an interrupt to the VR4120A by (if not masked).

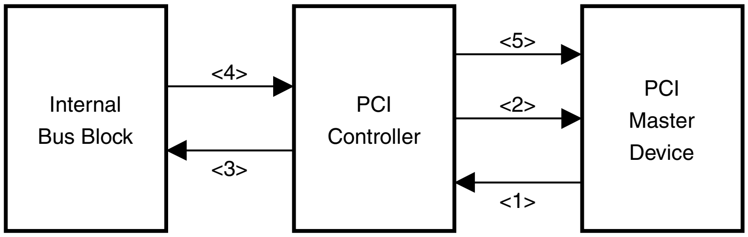
When the PCI Controller receives Bus Error on internal bus after it accepts delayed-read from PCI-side, it sets IRBER bit of P_IGSR register and PRBER bit of P_PGSR register, and issues interrupts to an external PCI-Host device and the VR4120A (if not masked). Then, the PCI Controller issues “target abort” responding to the access from PCI Master Device.

(2) Non delayed read transaction

When PDRTD bit in P_BCNT register is ‘1’, the PCI Controller uses “Non Delayed Read Transaction” rule for read transactions from Internal bus-side to PCI-side. In this mode, burst transfers are disconnected at every single word. The rule is as follows;

<1> PCI master device issues the read transaction to internal bus target block.
<2> The PCI Controller responds to this access by asserting DEVSEL_B, but the PCI Controller does not assert TRDY_B at this moment.
<3> The PCI Controller issues the read transaction to internal bus target block
<4> The internal bus target block accepts this access and the PCI Controller read a word from it.
<5> The PCI Controller asserts TRDY_B and return the 1-word data to PCI master device. Then, the PCI Controller issues “disconnect” to PCI master device when it issues the burst transfer. The PCI master device must terminates the transaction as soon as possible.

Figure 7-9. Non Delayed Read Transaction from PCI to Internal bus



When the PCI Controller receives Bus Error on internal bus after it accepts delayed-read from PCI-side, it sets IRBER bit of P_IGSR register and PRBER bit of P_PGSR register, and reports by interrupts to an external PCI-Host device and the VR4120A (if not masked). Then, the PCI Controller returns all “0” word to PCI master device and issues “disconnect” when PCI master device issues the burst transfer.

7.2.3 Abnormal Termination

7.2.3.1 On PCI bus

(1) Detecting parity error

When the access to the PCI Controller is issued on PCI bus and the PCI Controller detects the address parity error as a target, the PCI Controller issues a target abort to terminate the access. At the same time, the PCI Controller sets “Detected Parity Error” bit in configuration register, PPERR bit in P_IGSR register and DPERR bit in P_PGSR register, and issues interrupts to an external PCI-Host device and the VR4120A (if not masked). When “Parity Error Response” bit and “SERR# Enable” bit are set to “1”, the PCI Controller asserts SERR_B signal and set “signaled SERR_B” bit in configuration register (please note that in this case this wrong Address data is passed through the IBUS).

When the access to the PCI Controller is issued on PCI bus and the PCI Controller detects the data parity error as target, the PCI Controller sets following bits; “Detected Parity Error” bit in configuration register, PPERR bit in P_IGSR register and DPERR bit in P_PGSR register. In addition, the PCI Controller issues interrupts to an external PCI-Host device and the VR4120A (if not masked) too. However, the PCI Controller does not terminate the current transfer, and continues it.

When the PCI Controller issues the access on PCI bus and the PCI Controller detects the data parity error as master, the PCI Controller sets following bits; “Detected Parity Error” bit in configuration register, PPERR bit in P_IGSR register and DPERR bit in P_PGSR register. In addition, the PCI Controller issues interrupts to an external PCI-Host device and the VR4120A (if not masked). When “Parity Error Response” bit in configuration register is set, the PCI Controller asserts PERR# signal and set “Master Data Parity Error” bit in configuration register. However, the PCI Controller does not terminate the current transfer, and continues it.

(2) Received master abort as PCI-master

When the PCI Controller receives master abort on PCI bus as master, the PCI Controller sets “Received Master Abort” bit in configuration register, RMABT bit in P_PGSR register. In addition, the PCI Controller sets RDMAT bit in P_IGSR register when read transaction, or sets WRMAT bit in P_IGSR register when write transaction. Then the PCI Controller issues interrupts to an external PCI-Host device and the VR4120A (if not masked). The PCI Controller stops the access, and returns to the state in which the PCI Controller can accept a new access.

(3) Received target abort as PCI-master

When the PCI Controller receives target abort on PCI bus as master, the PCI Controller sets “Received Target Abort” bit in configuration register and RTABT bit in P_PGSR register. In addition, the PCI Controller sets RDTAT bit in P_IGSR register when read transaction, or sets WRTAT bit in P_IGSR register when write transaction. Then the PCI Controller issues interrupts to an external PCI-Host device and the VR4120A (if not masked). The PCI Controller stops the access, and returns to the state in which the PCI Controller can accept a new access.

(4) Received target disconnect as PCI-master

When the PCI Controller receives target disconnect on PCI bus as master, the PCI Controller terminates the current access and issues the access to the same target again in order to transfer remains of data.

(5) Received target retry as PCI-master

When the PCI Controller receives target retry on PCI bus as master, the PCI Controller terminates the current access and issues the access to the same target again in order to transfer data.

In the case that the value except for ‘0’ is set to P_RTMR register, the PCI Controller abandons the access when the number of target retry which the PCI Controller is received for the same access goes over the value in P_RTMR register. This function is called as “Retry-Timer”. Setting ‘0’ to P_RTMR register disables this function.

7.2.3.2 On Internal bus

(1) Bus Error

When the PCI Controller receives Bus Error on internal bus as master, the PCI Controller sets IRBER bit in P_IGSR register and PRBER bit in P_PGSR register in read transaction, or sets IWBER bit in P_IGSR register and PWBER bit in P_PGSR register in write transaction. Then, the PCI Controller issues interrupts to an external PCI-Host device and the VR4120A (if not masked). The PCI Controller stops the access, and returns to the state in which the PCI Controller can accept a new access.

7.2.4 Warning for Deadlocks

The PCI Controller can use Non-Delayed Read rule and Non-Posted Write rule for each direction. In these rules, the PCI Controller does not release the bus until it completes the transaction on the other bus. Therefore, if the PCI Controller is set to use Non-Delayed Read rule and Non-Posted Write rule on each buses and transactions are issued from both buses at the same time, deadlock will occur. It is recommended that either Non-Delayed Read rule or Non-Posted Write rule is used at one side at least.

The PCI Controller can accept 1 transaction on each bus and for read and write respectively. Once the PCI Controller accepts a transaction, it issues “retry” for the same kind of the transactions. If the master that has the highest priority issues the access to the PCI Controller continuously on each bus, the PCI Controller can not completes the transactions, and deadlock will occur.

7.3 PCI Power Management Interface

The PCI Controller has the mechanism for power management compliant to PCI Power Management Interface (PPMI) Rev.1.1 as a PCI-device. The PCI Controller does not control the power state of the chip, but issues signals of power transition from the VR4120A to an external PCI-Host device, or from the PCI-Host device to the VR4120A. The PCI-Host device and the VR4120A are responsible for the management of the power state.

The PCI Controller does not have the PCI bus control function for power management as a PCI-Host.

7.3.1 Power state

The PCI Controller supports D0, D1, D3hot and D3cold as PPMI states. Power Management Events (PME) would be generated from D0, D1 and D3hot.

In PPMI Rev.1.1, D0 is defined as the maximum power state, D1 as the optional power management state, D3hot as the power management state in which clock is suspended, D3cold as the power management state in which clock is suspended and power is removed.

7.3.2 Power management event

The PCI Controller supports Power Management Events (PME) from D0, D1 and D3hot. PME shows the event that issues the transition of the power state from device.

The PME is reported to an external PCI-Host device by asserting PME_B. The PCI Controller can assert PME_B. When a ‘1’ is written to PMERQ bit in P_PPCR register, the PCI Controller asserts PME_B signal.

7.3.3 Power supply

The PCI Controller does not need the auxiliary power supply (Vaux), because the PCI Controller does not support PME from D3cold.

The PCI Controller is designed on the assumption that there is no separated power supply, so that the transition to D3cold state, in which the power is removed, means the power for all parts of the chip removed.

The reset signals may be separated for each bus, but in case of the wake-up from D3cold state, which is defined to require the assertion of PCI-reset in PPMI Rev.1.1, all parts of the chip should be reset.

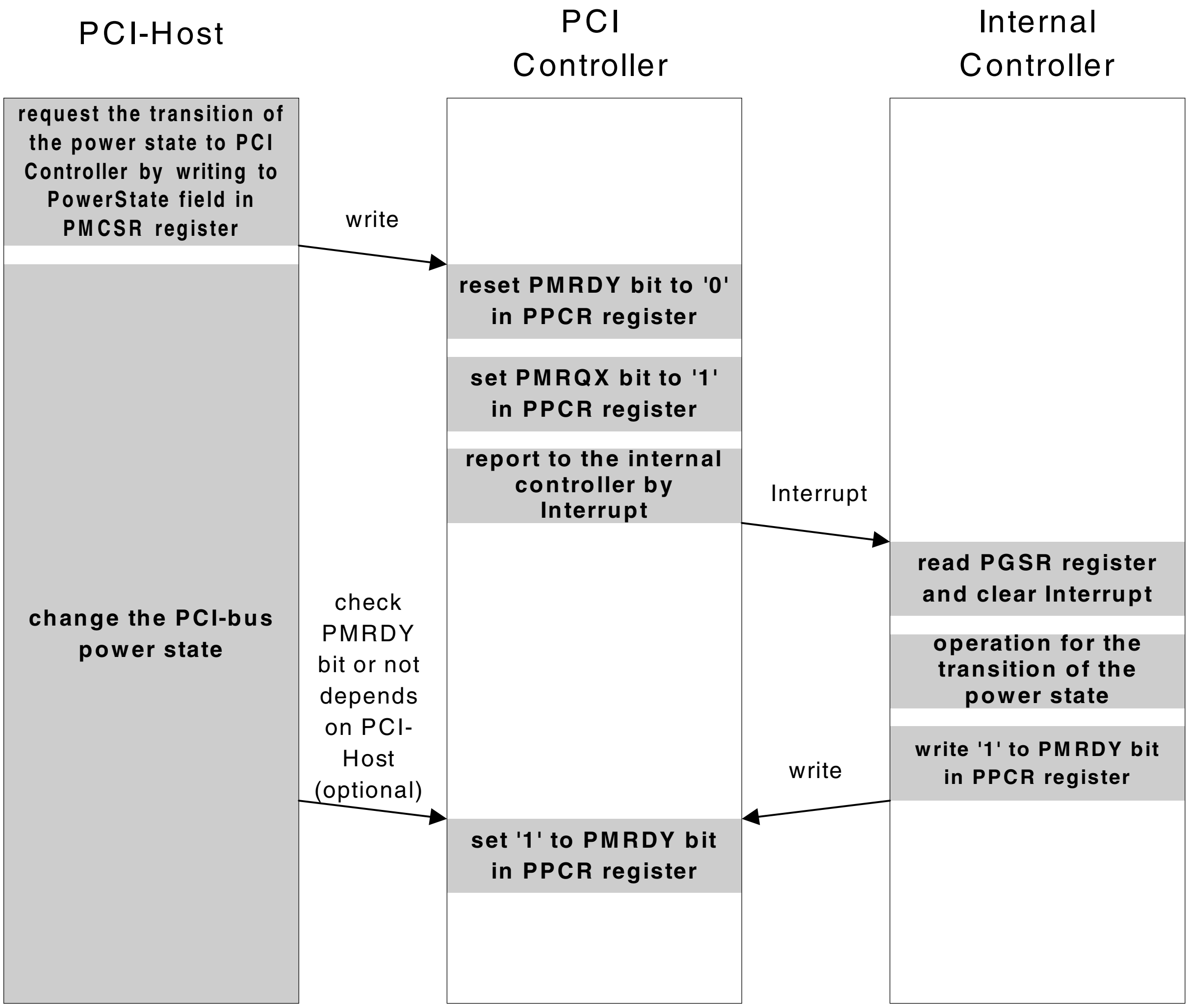
7.3.4 Power state transition

7.3.4.1 Transition by issue from PCI-Host

An example of the transition sequence is as follows:

1. When PCI-Host wants to change the power state of the chip, it writes the state code to Power State field in PMCSR register.
2. The PCI Controller resets PMRDY bit in P_PPCR register to a ‘0’.
3. The PCI Controller sets PMRQX bit (PMRQ0, PMRQ1, and PMRQ3) in P_PPCR register, and issues an interrupt to the VR4120A (if not masked).
4. The VR4120A clears the interrupt by reading PGSR register and knows the transition of power state is issued.
5. Then, the VR4120A knows which power state is issued by reading P_PPCR register.
6. The VR4120A executes the operations for the system that the chip is used in, if needed.
7. When the VR4120A is ready for the transition, the VR4120A writes a ‘1’ to PMRDY bit in P_PPCR register.
8. An external PCI-Host device is able to know that the chip is ready by reading PMRDY bit. The PCI-Host device does not need to wait the completion of the preparation of the chip. Therefore, power and clock may be removed suddenly.

Figure 7-10. The Sequence of the Transition by Issues from PCI-Host


PCI-Host
PCI Controller
Internal Controller
request the transition of the power state to PCI Controller by writing to PowerState field in PMCSR register
write
reset PMRDY bit to '0' in PPCR register
set PMRQX bit to '1' in PPCR register
report to the internal controller by Interrupt
Interrupt
read PGSR register and clear Interrupt
operation for the transition of the power state
change the PCI-bus power state
check PMRDY bit or not depends on PCI-Host (optional)
write '1' to PMRDY bit in PPCR register
write
set '1' to PMRDY bit in PPCR register

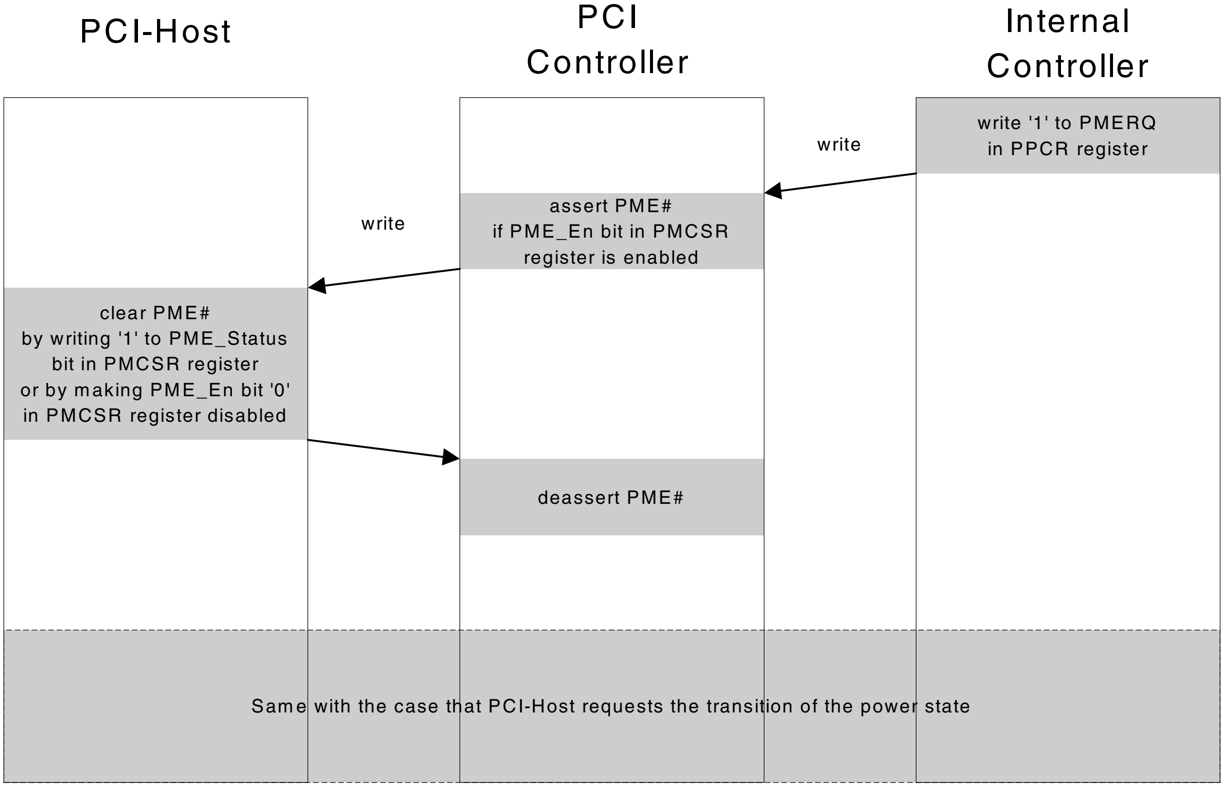
7.3.4.2 Transition by power management event

The sequence is as follows:

1. When Power Management Event occurs, the VR4120A writes a ‘1’ to PMERQ bit in P_PPCR register.
2. The PCI Controller asserts PME_B if PME_En bit in PMCSR register is enabled.
3. An external PCI-Host device writes a ‘1’ to PME_Status bit in PMCSR register or writes a ‘0’ to PME_En bit in PMCSR register in order to clear PME_B.
4. The PCI Controller deasserts PME_B.

Hereafter, as same case of the transition is issued by PCI-Host.

Figure 7-11. The Sequence of the Transition by PME


PCI-Host
PCI Controller
Internal Controller
write '1' to PMERQ in PPCR register
write
assert PME# if PME_En bit in PMCSR register is enabled
write
clear PME# by writing '1' to PME_Status bit in PMCSR register or by making PME_En bit '0' in PMCSR register disabled
deassert PME#
Same with the case that PCI-Host requests the transition of the power state

7.4 Functions in Host-mode

The functions described in this section are available when PMODE is set to low.

7.4.1 Generating configuration cycle

7.4.1.1 How to generate Configuration Cycle

The PCI Controller can generates Configuration Cycle on PCI bus by accessing of the following two registers;

PCI Configuration Address Register (P_PCAR)
PCI Configuration Data Register (P_PCDR)

At first, the information like address to be accessed for Configuration Cycle has to be set to P_PCAR register. Then, access to P_PCDR register generates Configuration Cycle on PCI bus.

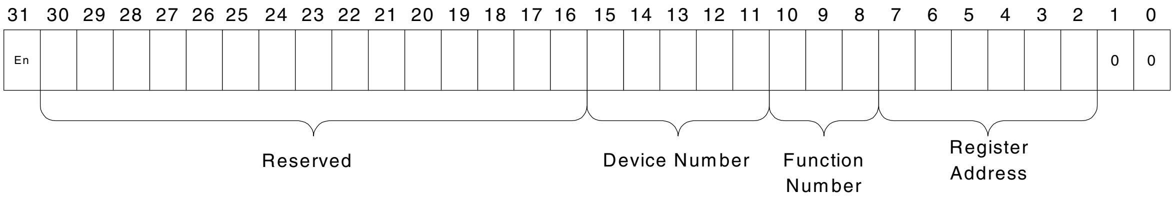
7.4.1.2 PCI Configuration Address Register (P_PCAR)

Setting a ‘1’ to bit 31(Configuration Cycle Enable bit) enables generating Configuration Cycle. When this bit is set to a ‘0’, access to P_PCDR register does not generate Configuration Cycle. The data will be ignored when a write transaction is issued, and all “0” data will be returned when a read transaction is issued.

There are two types of Configuration Cycle, Type0 for PCI devices except for PCI bridges and Type1 for PCI bridges, in PCI specification. The PCI Controller can generate both types of Configuration Cycle.

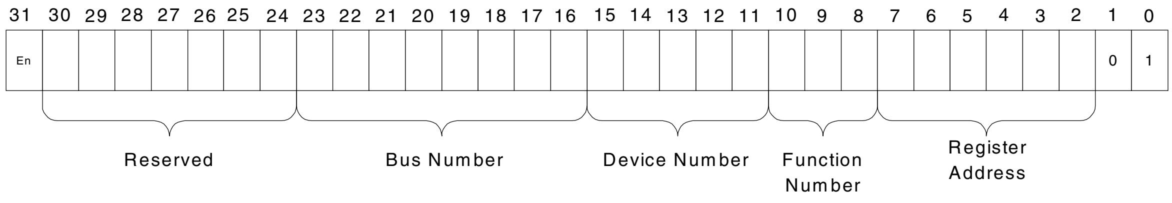
When Type1 Configuration Cycle is generated, the content to be set to P_PCAR register is as below;

Figure 7-12. The Content of P_PCAR Register for Type0 Configuration Cycle



The PCI Controller sends this information on AD [31:0] at the address phase of the cycle. The content to be set to P_PCAR register for Type 0 configuration cycle is as below;

Figure 7-13. The Content of P_PCAR Register for Type1 Configuration Cycle



As the PCI Controller does not have separated IDSEL output signals, it set one bit of AD [31:16] to ‘1’ and all other bits to ‘0’ in address phase of Type0 transaction, so that the bit set to ‘1’ can be used as the substitution for IDSEL. The details are described the following section.

7.4.1.3 PCI Configuration Data Register (P_PCDR)

When bit31 in the PCAR register is set to ‘1’, access to PCDR register generates Configuration Cycle.

Read access to P_PCDR register generates Configuration Read Cycle on PCI bus. Write access to P_PCDR register, also, generates Configuration Write Cycle on PCI bus.

7.4.1.4 IDSEL signals

As mentioned above, in address phase of Type0 transaction, the PCI Controller set one bit in AD [31:16] to ‘1’ and all other bit to ‘0’. The PCI Controller decodes the device number field in P_PCAR register and selects the bit to be set as below;

Table 7-1. Device Number Decode Table

Device Number	AD [31:16]
00000	0000 0000 0000 0001
00001	0000 0000 0000 0010
00010	0000 0000 0000 0100
00011	0000 0000 0000 1000
00100	0000 0000 0001 0000
00101	0000 0000 0010 0000
00110	0000 0000 0100 0000
00111	0000 0000 1000 0000
01000	0000 0001 0000 0000
01001	0000 0010 0000 0000
01010	0000 0100 0000 0000
01011	0000 1000 0000 0000
01100	0001 0000 0000 0000
01101	0010 0000 0000 0000
01110	0100 0000 0000 0000
01111	1000 0000 0000 0000
1xxxx	0000 0000 0000 0000

The bits of AD [31:16] can be used as IDSEL signal. However, it maybe causes the violation for PCI electrical specifications if AD [31:16] signal lines are connected to IDSEL ports of each PCI devices directly on boards. Therefore, it will be desirable to insert resistors in the connections that these signal lines are connected to each PCI device’s IDSEL port.

In this case, the delay may be caused before the time IDSEL input to PCI device becomes valid. The PCI Controller always uses 2clock-address-continuous-stepping, taking into consideration the influence of this delay.

Figure 7-14. An Example How to Connect AD [31:16] Signal Line to IDSEL Port

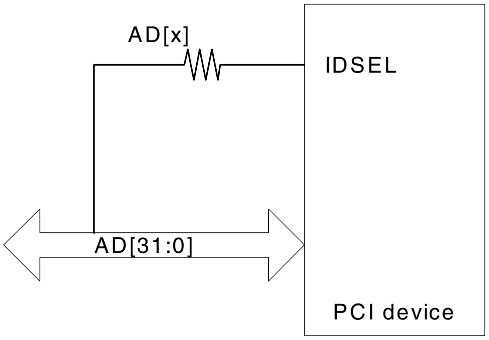
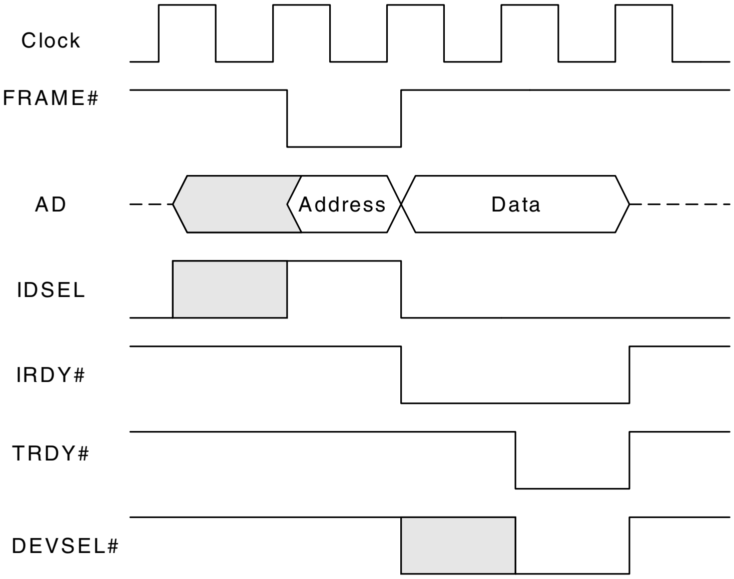


Figure 7-15. Address Stepping for IDSEL



7.4.2 PCI bus arbiter

The PCI Controller has an arbiter that supports 4 external PCI master devices. This arbiter is enabled only when in Host mode and PARBEN is set to high. When this internal arbiter is disabled, the PCI Controller asserts REQ_B output signal to external arbiter in order to acquire PCI bus both in NIC and in Host mode.

When there are less than 4 PCI master devices on PCI bus and this arbiter is used, REQ_B pins that are not used should be pull-up.

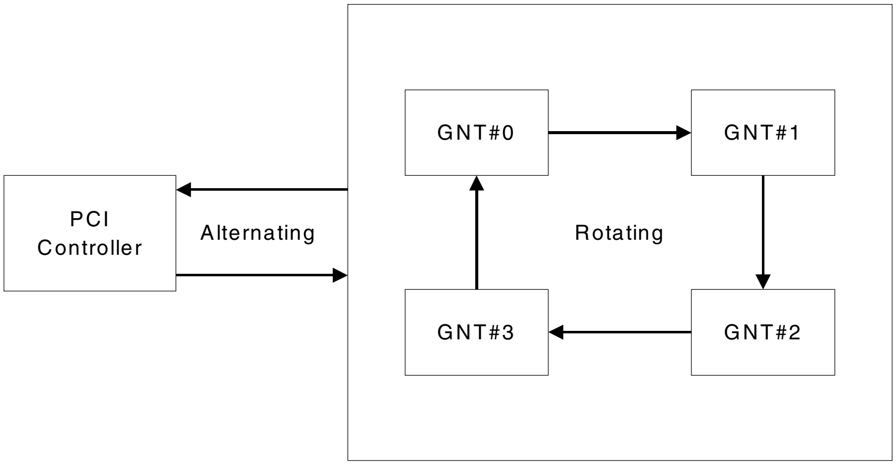
This internal arbiter has 2 modes for arbitration algorithm. These modes can be selected by PARBM bit in P_HMCR register.

7.4.2.1 Alternating mode

PCI master devices except the PCI Controller are arbitrated as one group in this mode. Priority alternates The PCI Controller with the group of PCI master devices on the transaction by transaction. In the group of PCI master devices, priority rotates among them.

When all REQ_B input signals to this arbiter go up to high, which means no device issues the acquisition of PCI bus, this arbiter gives the right of use of PCI bus to the PCI Controller in order to make the PCI Controller drive AD lines and CBE_B lines as arbitration parking.

Figure 7-16. Arbitration in Alternating Mode

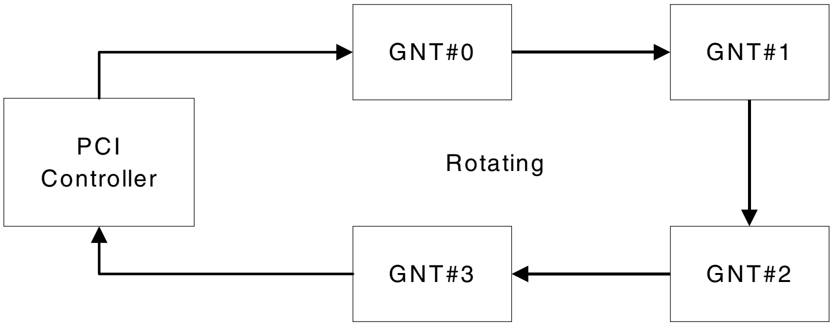


7.4.2.2 Rotating mode

Priority rotates among all PCI master devices including the PCI Controller in this mode.

When all REQ_B input signals to this arbiter go up to high, which means no device issues the acquisition of PCI bus, this arbiter gives the right of use of PCI bus to the device that had acquired PCI bus last as arbitration parking.

Figure 7-17. Arbitration in Rotating Mode



7.4.3 Reset output

In Host mode, the PCI Controller asserts reset signal for PCI bus, when PRSTO bit in P_HMCR register is set to a "1". In order to deassert the signal, the VR4120A should reset PRSTO bit to a "0".

7.4.4 Interrupt input

The PCI Controller has an interrupt input port and the SERR_B input port so as to receive interrupts and SERR_B from PCI-devices.

If any of PCI interrupts is asserted, PINTR bit in P_IGSR register is set and the PCI Controller issues an interrupt to the VR4120A (if not masked). The PCI Controller handles the interrupt input port in "level sensitive" way.

If SERR_B comes, PSERI bit in P_IGSR register is set and the PCI Controller issues an interrupt to the VR4120A (if not masked). The PCI Controller handles SERR_B input in "edge sensitive" way.

The PCI Controller has only 1 port for interrupt input and SERR_B input each, so the interrupt or SERR_B from PCI devices should be combined, in wire-ORed way for example, on the board.

7.5 Registers

7.5.1 Register map

Offset Address	Register Name	R/W		Access	Description
		Internal bus	PCI		
1000_4000H	P_PLBA	R/W	R/W	W/H/B	PCI Lower Base Address Register
1000_4004H	N/A	-	-	-	Reserved
1000_4008H	P_IBBA	R/W	R/W	W/H/B	Internal Bus Base Address Register
1000_400CH	N/A	-	-	-	Reserved
1000_4010H	P_VERR	R	R	W/H/B	Version Register
1000_4014H	P_PCAR	R/W	R	W/H/B	PCI Configuration Address Register
1000_4018H	P_PCDR	R/W	R	W/H/B	PCI Configuration Data Register
1000_401CH	P_IGSR	RC	R/W	W	Internal Bus General Status Register
1000_4020H	P_IIMR	R/W	R/W	W/H/B	Internal Bus Interrupt Mask Register
1000_4024H	P_PGSR	R/W	RC	W	PCI General Status Register
1000_4028H	P_PIMR	R/W	R/W	W/H/B	PCI Interrupt Mask Register
1000_402CH	N/A	-	-	-	Reserved
1000_4030H	P_HMCR	R/W	R/W	W/H/B	Host Mode Control Register
1000_4034H: 1000_403CH	N/A	-	-	-	Reserved
1000_4040H	P_PWCD	R/W	R	W/H/B	Power Consumption Data Register
1000_4044H	P_PWDD	R/W	R	W/H/B	Power Dissipation Data Register
1000_4048H: 1000_404CH	N/A	-	-	-	Reserved
1000_4050H	P_BCNT	R/W	R/W	W/H/B	Bridge Control Register
1000_4054H	P_PPCR	R/W	R	W/H/B	Power Control Register
1000_4058H	P_SWRR	-	W	W	Software Reset Register
1000_405CH	P_RTMR	R/W	R/W	W/H/B	Retry Timer register
1000_4060H: 1000_40FCH	N/A	-	-	-	Reserved
1000_4100H: 1000_41FCH	P_CONFIG	See 7.5.19	See 7.5.19	W/H/B	PCI Configuration Registers

Remarks 1. In the “R/W” field,
“W” means “writeable”,
“R” means “readable”,
“RC” means “read-cleared”,
“- “ means “not accessible”.

2. All internal registers are 32-bit word-aligned registers.

3. The burst access to the internal register is prohibited.
If such burst access has been occurred, IRERR bit in NSR is set and NMI will assert to CPU.

4. Read access to the reserved area will set the CBERR bit in the NSR register and the dummy read response data with the data-error bit set on SysCMD [0] is returned.

5. Write access to the reserved area will set the CBERR bit in the NSR register, and the write data is lost.

6. In the “Access” filed,
“W” means that word access is valid,
“H” means that half word access is valid,
“B” means that byte access is valid.

7. Write access to the read-only register cause no error, but the write data is lost.

8. The CPU can access all internal registers, but IBUS master device cannot access them.

7.5.2 P_PLBA (PCI Lower Base Address Register)

When the PCI Controller issues 32-bit PCI address, this register contains PCI base address. When the access from Internal bus-side to PCI-side comes, the PCI Controller replaces the upper 10 bits of the address on internal bus with the upper 10 bits of this register, and issues as the address on PCI bus.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
31:0	PLBA	R/W (upper 10 bits) R (lower 22 bits)	R/W (upper 10 bits) R (lower 22 bits)	0	PCI lower base address. Upper 10 bits are readable/writeable. Lower 22 bits are read-only.

7.5.3 P_IBBA (Internal Bus Base Address Register)

This register contains internal bus base address. When the access from PCI-side to internal bus-side comes, the PCI Controller replaces the upper 10-bits of the address on PCI with the upper 10-bits of this register, and issues as the address on internal bus.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
31:0	IBBA	R/W (upper 10 bits) R (lower 22 bits)	R/W (upper 10 bits) R (lower 22 bits)	0	Internal bus base address. Upper 10 bits are readable/writeable. Lower 22 bits are read-only.

7.5.4 P_VERR (Version Register)

This register contains the version number of the PCI Controller. The upper 16 bits indicate the major version, and the lower 16 bits indicate the minor version.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
31:0	VERR	R/W (upper 10 bits) R (lower 22 bits)	R/W (upper 10 bits) R (lower 22 bits)	0001_ 0000H	Hardwired to “0001_0000H”, which indicates the current version of The PCI Controller.

7.5.5 P_PCAR (PCI Configuration Address Register)

PCAR register is used to set the information for Configuration Cycle. How to generate Configuration Cycle is described in **7.4.1 Generating configuration cycle**.

The PCI Controller can executes Configuration Cycle only in Host-mode.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
31	COCEN	R/W	R	0	Configuration Cycle Generation Enable. When the bit is set to a ‘1’, reading from/writing to PCDR generates Configuration Cycle. When this bit is set to a ‘0’, reading from/writing to PCDR does not generate Configuration Cycle.
30:24	Reserved	-	-	0	Hardwired to ‘0’s
23:16	BUSNM	R/W	R	0	Bus Number.
15:11	DEVNM	R/W	R	0	Device Number.
10:8	FNCNM	R/W	R	0	Function Number.
7:2	REGAC	R/W	R	0	Offset Address for each device.
1:0	CYCTP	R/W	R	00	Select Configuration Cycle Type. ‘00’: Configuration Type 0 ‘01’: Configuration Type 1

7.5.6 P_PCDR (PCI Configuration Data Register)

PCDR register is used to read/write configuration data during Configuration Cycle. How to generate Configuration Cycle is described in **7.4.1 Generating configuration cycle**.

The PCI Controller can executes Configuration Cycle only in Host-mode.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
31:0	PCDR	R/W	R	0	The data during Configuration Cycle.

7.5.7 P_IGSR (Internal Bus-side General Status Register)

IGSR register shows the interrupt status of the PCI Controller to the VR4120A. When an event that triggers interruption occurs, the PCI Controller sets a bit in this register corresponds to the event. When the corresponding bit in IIMR is set, the PCI Controller asserts an internal interrupt signal to the VR4120A. Reading this register clears all of bits in the register.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
31:16	IUINT	R	R/W	0	Interrupts that can be defined by the system the chip is used. When ‘1’ is written to a bit in this field from PCI bus side, interrupt to the VR4120A is asserted.
15:12	Reserved	-	-	0	Hardwired to ‘0’s.
11	PINTR	R	R	0	Used only in Host-mode. Interrupts from PCI-device occurred. ‘1’ indicates that a PCI-interruption occur.
10	PSERI	R	R	0	Used only in Host-mode. SERR_B from external PCI-devices is asserted. ‘1’ indicates that SERR_B is asserted.
9	PPERR	R	R	0	PCI Parity Error. ‘1’ indicates that the PCI Controller has detected a parity error on PCI bus.
8	PPREQ	R	R	0	The transition of PPMI Power state issued. ‘1’ indicates that PCI-Host issues the transition of power state of The PCI Controller. When this bit is set, the VR4120A should check PPCR register to know which state the PCI Controller moves to.
7	SRREQ	R	R	0	Software Reset is issued. ‘1’ indicates that PCI-Host issues Software Reset, when PCI-Host writes to SWRS register. the VR4120A should be set SWRDN bit in PGSR register to a ‘1’ after the completion of software reset.
6	IRBER	R	R	0	Internal bus Error in read transaction. ‘1’ indicates that the PCI Controller has received Bus Error on internal bus during read transaction as master.
5	IWBER	R	R	0	Internal bus Error in write transaction. ‘1’ indicates that the PCI Controller has received Bus Error on internal bus during write transaction as master.
4	IFDSC	R	R	0	Internal bus FIFO data discarded. ‘1’ indicates that the PCI Controller has discarded the data for read-delayed-transaction in FIFO, because the same issue has not been repeated within 2^{15} clock.
3	RDTAT	R	R	0	PCI Target Abort in read transaction. ‘1’ indicates that the PCI Controller has received Target Abort on PCI bus during read transaction as master.
2	WRTAT	R	R	0	PCI Target Abort in write transaction. ‘1’ indicates that the PCI Controller has received Target Abort on PCI bus during write transaction as master.
1	RDMAT	R	R	0	PCI Master Abort in read transaction. ‘1’ indicates that the PCI Controller has received Master Abort on PCI bus during read transaction as master.
0	WRMAT	R	R	0	PCI Master Abort in write transaction. ‘1’ indicates that the PCI Controller has received Master Abort on PCI bus during write transaction as master.

7.5.8 P_IIMR (Internal Bus Interrupt Mask Register)

IIMR register masks the interruption for each corresponding event. A mask bit, which locates in the same bit position to a corresponding bit in IGSR, controls interruption triggered by the event. When a bit of this register is reset to ‘0’, the corresponding bit of the IGSR is masked. If it is set to ‘1’, the corresponding bit is unmasked. When the mask bit is reset and the bit in IGSR is set, the PCI Controller sets the interrupt signal to the VR4120A.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
31:16	IUINT	R/W	R/W	0	Mask bits for the Interrupts defined by the system the chip is used. ‘0’ means masked. ‘1’ means unmasked.
15:12	Reserved	-	-	0	Hardwired to ‘0H’.
11	PINTR	R/W	R/W	0	Mask bit for the PCI interruption ‘0’ means masked. ‘1’ means unmasked.
10	PSERI	R/W	R/W	0	Mask bit for the PCI SERR# ‘0’ means masked. ‘1’ means unmasked.
9	PPERR	R/W	R/W	0	Mask bit for PCI Parity Error ‘0’ means masked. ‘1’ means unmasked.
8	PPREQ	R/W	R/W	0	Mask bit for the state transition for PPMI. ‘0’ means masked. ‘1’ means unmasked.
7	SRREQ	R/W	R/W	0	Mask bit for the Software Reset issue. ‘0’ means masked. ‘1’ means unmasked.
6	IRBER	R/W	R/W	0	Mask bit for internal bus Error in read transaction. ‘0’ means masked. ‘1’ means unmasked.
5	IWBER	R/W	R/W	0	Mask bit for internal bus Error in write transaction. ‘0’ means masked. ‘1’ means unmasked.
4	IFDSC	R/W	R/W	0	Mask bit for the discard of data in internal bus FIFO. ‘0’ means masked. ‘1’ means unmasked.
3	RDTAT	R/W	R/W	0	Mask bit for PCI Target Abort in read transaction. ‘0’ means masked. ‘1’ means unmasked.
2	WRTAT	R/W	R/W	0	Mask bit for PCI Target Abort in write transaction. ‘0’ means masked. ‘1’ means unmasked.
1	RDMAT	R/W	R/W	0	Mask bit for PCI Master Abort in read transaction. ‘0’ means masked. ‘1’ means unmasked.
0	WRMAT	R/W	R/W	0	Mask bit for PCI Master Abort in write transaction. ‘0’ means masked. ‘1’ means unmasked.

7.5.9 P_PGSR (PCI-side General Status Register)

PGSR register shows the interrupt status of the PCI Controller to PCI-side (which means PCI-Host). When an event that triggers interruption occurs, the PCI Controller sets a bit in PGSR corresponds to the type of incident. If the interruption is not masked, the PCI Controller interrupts to PCI-Host using the interrupt signal.

Reading this register from PCI-side clears all of bits in the register.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
31:16	IUINT	R	R/W	0	Interrupts that can be defined by the system the chip is used. When ‘1’ is written to a bit in this field from PCI bus side, interrupt to the VR4120A is asserted.
15:12	Reserved	-	-	0	Hardwired to ‘0’s.
11	PINTR	R	R	0	Used only in Host-mode. Interrupts from PCI-device occurred. ‘1’ indicates that a PCI-interruption occur.
10	PSERI	R	R	0	Used only in Host-mode. SERR_B from external PCI-devices is asserted. ‘1’ indicates that SERR_B is asserted.
9	PPERR	R	R	0	PCI Parity Error. ‘1’ indicates that the PCI Controller has detected a parity error on PCI bus.
8	PPREQ	R	R	0	The transition of PPMI Power state issued. ‘1’ indicates that PCI-Host issues the transition of power state of The PCI Controller. When this bit is set, the VR4120A should check PPCR register to know which state the PCI Controller moves to.
7	SRREQ	R	R	0	Software Reset is issued. ‘1’ indicates that PCI-Host issues Software Reset, when PCI-Host writes to SWRS register. the VR4120A should be set SWRDN bit in PGSR register to a ‘1’ after the completion of software reset.
6	IRBER	R	R	0	Internal bus Error in read transaction. ‘1’ indicates that the PCI Controller has received Bus Error on internal bus during read transaction as master.
5	IWBER	R	R	0	Internal bus Error in write transaction. ‘1’ indicates that the PCI Controller has received Bus Error on internal bus during write transaction as master.
4	IFDSC	R	R	0	Internal bus FIFO data discarded. ‘1’ indicates that the PCI Controller has discarded the data for read-delayed-transaction in FIFO, because the same issue has not been repeated within 2^{15} clock.
3	RDTAT	R	R	0	PCI Target Abort in read transaction. ‘1’ indicates that the PCI Controller has received Target Abort on PCI bus during read transaction as master.
2	WRTAT	R	R	0	PCI Target Abort in write transaction. ‘1’ indicates that the PCI Controller has received Target Abort on PCI bus during write transaction as master.
1	RDMAT	R	R	0	PCI Master Abort in read transaction. ‘1’ indicates that the PCI Controller has received Master Abort on PCI bus during read transaction as master.
0	WRMAT	R	R	0	PCI Master Abort in write transaction. ‘1’ indicates that the PCI Controller has received Master Abort on PCI bus during write transaction as master.

7.5.10 P_IIMR (Internal Bus Interrupt Mask Register)

IIMR register masks the interruption for each corresponding event. A mask bit, which locates in the same bit position to a corresponding bit in IGSR, controls interruption triggered by the event. When a bit of this register is reset to ‘0’, the corresponding bit of the IGSR is masked. If it is set to ‘1’, the corresponding bit is unmasked. When the mask bit is reset and the bit in IGSR is set, the PCI Controller sets the interrupt signal to the VR4120A.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
31:16	PUINT	R	R/W	0000H	Interrupts which can be defined by the system the chip is used. If ‘1’ is written to this field from internal bus side, an interrupt to PCI-Host is asserted.
15:11	Reserved	-	-	0H	Hardwired to ‘0H’
10	IPREQ	R	R	0	The transition of PPMI Power state issued. ‘1’ indicates that the VR4120A issues the transition of power state of The PCI Controller.
9	SWRDN	R/W	R	0	Software Reset done. ‘1’ indicates that the software reset has been done.
8	DPERR	R	R	0	Detected PCI Parity Error. ‘1’ indicates that the PCI Controller has detected a parity error on PCI bus.
7	SSERR	R	R	0	Signaled SERR#. ‘1’ indicates that the PCI Controller has asserted SERR_B.
6	RMABT	R	R	0	Received Master Abort. ‘1’ indicates that the PCI Controller has received Master Abort as master.
5	RTABT	R	R	0	Received Target Abort. ‘1’ indicates that the PCI Controller has received Target Abort as master.
4	STABT	R	R	0	Signaled Target Abort. ‘1’ indicates that the PCI Controller has executed Target Abort as target.
3	PFDSC	R	R	0	PCI FIFO discarded. ‘1’ indicates that the PCI Controller has discarded the data for read-delayed-transaction in FIFO, because the same issue has not been repeated within 2^{15} clock.
2	RTYTE	R	R	0	Retry Timer Expired. ‘1’ indicates that Retry Timer has expired and the PCI Controller abandons the retry-access. See **7.2.3.1 (5)**.
1	PRBER	R	R	0	Internal bus Error in read transaction ‘1’ indicates that the PCI Controller has received Bus Error on internal bus during read transaction as master.
0	PWBER	R	R	0	Internal bus Error in write transaction ‘1’ indicates that the PCI Controller has received Bus Error on internal bus during write transaction as master.

7.5.11 P_PIMR (PCI Interrupt Mask Register)

PIMR register masks interruptions. A mask bit, which locates in the same bit position to a corresponding bit in PGSR, can mask the interruption. When a bit of this register is reset to ‘0’, the corresponding bit of the PGSR is masked. When it is set to ‘1’, the corresponding bit is unmasked. When the mask bit is reset and the bit in PGSR is set, the PCI Controller sets the interrupt signal to PCI-Host.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
31:16	PUINT	R/W	R/W	0	Mask bits for the Interrupts defined by the system the chip is used. ‘0’ means masked. ‘1’ means unmasked.
15:11	Reserved	-	-	0	Hardwired to ‘0’s.
10	IPREQ	R/W	R/W	0	Mask bit for the state transition for PPMI. ‘0’ means masked. ‘1’ means unmasked.
9	SWRDN	R/W	R/W	0	Mask bit for Software Reset done. ‘0’ means masked. ‘1’ means unmasked.
8	DPERR	R/W	R/W	0	Mask bit for Detected Parity Error. ‘0’ means masked. ‘1’ means unmasked.
7	SSERR	R/W	R/W	0	Mask bit for Signaled SERR_B. ‘0’ means masked. ‘1’ means unmasked.
6	RMABT	R/W	R/W	0	Mask bit for Received Master Abort. ‘0’ means masked. ‘1’ means unmasked.
5	RTABT	R/W	R/W	0	Mask bit for Received Target Abort. ‘0’ means masked. ‘1’ means unmasked.
4	STABT	R/W	R/W	0	Mask bit for Signaled Target Abort. ‘0’ means masked. ‘1’ means unmasked.
3	PFDSC	R/W	R/W	0	Mask bit for the discard of data in PCI FIFO. ‘0’ means masked. ‘1’ means unmasked.
2	RTYTE	R/W	R/W	0	Mask bit for Retry Timer Expired. ‘0’ means masked. ‘1’ means unmasked.
1	PRBER	R/W	R/W	0	Mask bit for internal bus Error in read transaction. ‘0’ means masked. ‘1’ means unmasked.
0	PWBER	R/W	R/W	0	Mask bit for internal bus Error in write transaction. ‘0’ means masked. ‘1’ means unmasked.

7.5.12 P_HMCR (Host Mode Control Register)

This register is used to control the PCI-Host functions.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
31	PRSTO	R/W	R	0	Reset Out. PCI reset output as Host. The PCI Controller asserts PRSTO during this bit is ‘1’.
30:1	Reserved	-	-	0	Hardwired to ‘0’s.
0	PARBM	R/W	R/W	0	PCI arbiter mode. The bit defines arbiter mode. 0: Alternating mode 1: Rotating mode

7.5.13 P_PCDR (Power Consumption Data Register)

This register is used to indicate the power consumption data for each power state.

The VR4120A should be set the value to this register as initialization. The unit of data is “0.01Watts”.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
31:24	D3CSP	R/W	R	00H	D3 Power Consumption Data.
23:16	Reserved	-	-	00H	Hardwired to ‘00H’.
15:8	D1CSP	R/W	R	00H	D1 Power Consumption Data.
7:0	D0CSP	R/W	R	00H	D0 Power Consumption Data.

7.5.14 P_PDDR (Power Dissipation Data Register)

This register is used to indicate the power dissipation data for each power state.

The VR4120A should be set the values to this register as initialization. The unit of data is “0.01Watts”.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
31:24	D3DSP	R/W	R	00H	D3 Power Dissipation Data.
23:16	Reserved	-	-	00H	Hardwired to ‘00H’.
15:8	D1DSP	R/W	R	00H	D1 Power Dissipation Data.
7:0	D0DSP	R/W	R	00H	D0 Power Dissipation Data.

7.5.15 P_BCNT (Bridge Control Register)

This register is used to control the PCI-internal bus bridge function.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
31	INITD	R/W	R	0	Initialize done. The VR4120A should set this bit to '1' after the initialization of the chip. The PCI Controller always issues "retry" against the access on PCI to the PCI Controller bus when this bit is set to a '0'.
30:6	Reserved	-	-	0	Hardwired to '0's.
5	ICMDS	R/W	R/W	0	Internal bus Command Select. When this bit is set to '1', the PCI Controller uses Memory command on Internal bus. When this bit is set to '0', the PCI Controller uses I/O command on Internal bus.
4	DACEN	R/W	R/W	0	Dual Address Cycle Enable. Reserved for the future use. The bit has to be set to 0.
3	PDRTD	R/W	R/W	0	PCI Delayed Read Transaction Disable. '1' disables the PCI Controller to generate Delayed Read Transaction on PCI bus. PCI Read transactions are executed as None Delayed Read Transaction.
2	PPWRD	R/W	R/W	0	PCI Posted Write Transaction Disable. '1' disables the PCI Controller to generate Posted Write Transaction on PCI bus. PCI Write transactions are executed as None Posted Write Transaction.
1	IDRTD	R/W	R/W	0	Internal bus Delayed Read Transaction Disable. '1' disables the PCI Controller to generate Delayed Read Transaction on Internal bus. PCI Read transactions are executed as None Delayed Read Transaction.
0	IPWRD	R/W	R/W	0	Internal bus Posted Write Transaction Disable. '1' disables the PCI Controller to generate Posted Write Transaction on Internal bus. PCI Write transactions are executed as None Posted Write Transaction.

7.5.16 P_PPCR (PCI Power Control Register)

This register is used to control the power state for PPMI.

See **7.6 Information for Software** for further details.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
31	PMRDY	R/W	R	0	Power Management Ready. '1' indicates that the transition of power state has been done. When PCI-Host writes to PowerState field of PMCSR register in Configuration Space, this bit is reset to '0'. the VR4120A should be set '1' to this bit after the transition of power state has been done.
30	PMERQ	W	-	0	PME# issue. This bit is only writeable from Internal bus-side. The PCI Controller asserts PME# signal when '1' is written to this bit. If PCI-Host want to clear PME# asserting, it should write '1' to PME_Status bit in PMCSR register, or disable PME_En bit in PMCSR register. See **7.6** for further details.
29:4	Reserved	-	-	0	Hardwired to '0's.
3	PMRQ0	R/W	R	0	The transition to D0 power state issued. '1' indicates that PCI-Host issues the transition to D0 state. The VR4120A should write '1' to this bit in order to clear after the recognition that this bit is set.
2	PMRQ1	R/W	R	0	The transition to D1 power state issued. '1' indicates that PCI-Host issues the transition to D1 state. The VR4120A should write '1' to this bit in order to clear after the recognition that this bit is set.
1	Reserved	-	-	0	Hardwired to a '0'.
0	PMRQ3	R/W	R	0	The transition to D3 power state issued. '1' indicates that PCI-Host issues the transition to D3 state. The VR4120A should write '1' to this bit in order to clear after the recognition that this bit is set.

7.5.17 P_SWRR (Software Reset Register)

This register is used for Software Reset and can be written only from PCI-side. Writing any value to this register causes the Software Reset. When this register is written, the SRREQ bit in P_IGSR register will be set and an internal interrupt will be asserted to the VR4120A. The VR4120A is responsible to execute reset-operations (for example, asserting warm-reset to all blocks).

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
31:0	SWRS	-	W	-	This register is write-only.

7.5.18 P_RTMR (Retry Timer Register)

This register is used to set the limitation of the number of retry repetition. '0' disables this function. See **7.2.3.1 (5) Received target retry as PCI-master** for further details.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
31:0	RTMR	R/W	R/W	0000_ 0000H	Sets the number of retry repetition. '0000_0000H' disables this function.

7.5.19 P_CONFIG (PCI Configuration Registers)

7.5.19.1 PCI configuration register map

Offset Address[Note]	31 24	23 16	15 8	7 0
1000_4100H	Device ID		Vendor ID	
1000_4104H	Status		Command	
1000_4108H	Class Code		Revision ID	
1000_410CH	Reserved	Header Type	Latency Timer	Cache Line Size
1000_4110H	Window Memory Base Address Register			
1000_4114H	Register Memory Base Address Register			
1000_4118H	Reserved			
1000_411CH	Reserved			
1000_4120H	Reserved			
1000_4124H	Reserved			
1000_4128H	Reserved			
1000_412CH	Subsystem ID		Subsystem Vendor ID	
1000_4130H	Reserved			
1000_4134H	Reserved			Cap_Ptr
1000_4138H	Reserved			
1000_413CH	Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line
1000_4140H	PMC		Next_Item_Ptr	Cap_ID
1000_4144H	PMData	Reserved	PMCSR	
1000_4148H: 1000_41FCH	Reserved			

Note The view from PCI side address is assigned by "Register Memory Base Address Register".

Offset Address	Register Name	Size (byte)	Internal bus	PCI	Description
1000_4100H	Vendor ID	2	R	R	Vendor ID for NEC = 1033H
1000_4102H	Device ID	2	R	R	Device Specific ID
1000_4104H	Command	2	R/W	R/W	PCI Command
1000_4106H	Status	2	R/W	R/W	PCI Status
1000_4108H	Revision ID	1	R	R	Revision ID
1000_4109H	Class Code	3	R	R	Class Code
1000_410CH	Cache Line Size	1	R/W	R/W	Cache Line Size
1000_410DH	Latency Timer	1	R/W	R/W	Maximum Latency Timer
1000_410EH	Header Type	1	R	R	Header Type = 00H
1000_410FH	Reserved	1	-	-	Reserved
1000_4110H	Window Memory Base Address	4	R/W	R/W	Memory Base Address for access window
1000_4114H	Register Memory Base Address	4	R/W	R/W	Memory Base Address for the registers of The PCI Controller
1000_4118H	Reserved	4	-	-	Reserved
1000_411CH	Reserved	4	-	-	Reserved
1000_4120H	Reserved	4	-	-	Reserved
1000_4124H	Reserved	4	-	-	Reserved
1000_4128H	Reserved	4	-	-	Reserved
1000_412CH	Subsystem Vendor ID	2	R/W	R	Subsystem Vendor ID
1000_412EH	Subsystem ID	2	R/W	R	Subsystem ID
1000_4130H	Reserved	4	-	-	Reserved
1000_4134H	Cap_Ptr	1	R	R	Capabilities Pointer = 40H
1000_4135H	Reserved	3	-	-	Reserved
1000_4138H	Reserved	4	-	-	Reserved
1000_413CH	Interrupt Line	1	R/W	R/W	Interrupt Line
1000_413DH	Interrupt Pin	1	R	R	Interrupt Pin = 01H
1000_413EH	Min_Gnt	1	R/W	R	Minimum GNT
1000_413FH	Max_Lat	1	R/W	R	Maximum Latency
1000_4140H	Cap_ID	1	R	R	New Capability ID = 01H
1000_4141H	Next_Item_Ptr	1	R	R	Next Item Pointer = 00H
1000_4142H	PMC	2	R/W	R	Power Management Capabilities
1000_4144H	PMCSR	2	R/W	R/W	Power Management Control/Status
1000_4146H	Reserved	1	-	-	Reserved
1000_4147H	PMData	1	R	R	Power Management Data
1000_4148H: 1000_41FFH	Reserved	8	-	-	Reserved

Configuration registers can be read/written from the VR4120A. Configuration registers starts from 1000_4100H of offset address in the internal registers.

7.5.19.2 Vendor ID register

This register identifies the manufacturer of the device. The identifier for NEC is ‘1033H”.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
15:0	Vendor ID	R	R	1033H	Hardwired to ‘1033H’, which means the Vendor ID of NEC

7.5.19.3 Device ID register

This register identifies the particular device. The manufacturer of the device allocates this identifier.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
15:0	Device ID	R	R	00A7H	Hardwired to ‘00A7H’.

7.5.19.4 Command register

This register provides coarse control over a device's ability to generate and respond to PCI cycles. This register is valid in Host-mode. The VR4120A should set the register.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
15:10	Reserved	R	R	0	Hardwired to '0's.
9	Fast Back-to-Back Enable	R	R	0	Hardwired to a '0', because the PCI Controller cannot generate fast back-to-back transaction.
8	System Error Enable	R/W	R/W	0	This bit is an enable bit for the SERR_B driver. '1' enables the SERR_B driver.
7	Wait Cycle Enable	R	R	0	Hardwired to a '0', because the PCI Controller does not use address/data stepping as PCI-device.
6	Parity Error Response	R/W	R/W	0	This bit controls the PCI Controller's response to parity errors. When the bit is set to a '0', the PCI Controller must take its normal action if a parity error is detected. When the bit is set to a '1', the PCI Controller sets its Detected Parity Error status bit (bit15 in the Status register) if an error is detected, but does not assert PERR_B and continues normal operation.
5	VGA Palette Snoop Enable	R	R	0	Hardwired to a '0', because the PCI Controller does not have VGA function.
4	Memory Write and Invalidate Enable	R/W	R/W	0	This is an enable bit for using the Memory Write and Invalidate command. When this bit is set to a '1', the PCI Controller can generate the Memory Write and Invalidate command. When it is a '0', Memory Write is used and Memory Write and Invalidate command is not issued.
3	Special Cycle Recognition	R	R	0	Hardwired to a '0', because the PCI Controller dose not respond to Special Cycle operations.
2	Bus Master Enable	R/W	R/W	0	Controls The PCI Controller's ability to act as a master on the PCI bus. '0' disables the PCI Controller from generating PCI accesses. '1' allows the PCI Controller to behave as a bus master.
1	Memory Access Enable	R/W	R/W	0	Controls a device's response to Memory Space accesses. '0' disables The PCI Controller's response. '1' allows the PCI Controller to respond to Memory space accesses.
0	I/O Access Enable	R	R	0	Hardwired to '0', because the PCI Controller does not issue the I/O space.

7.5.19.5 Status register

This register is used to show PCI bus related events status. These bits are set when events related to the status on PCI bus and reset to '0' by writing '1'.

In Host-mode, any bit in this register is not set even if corresponding events occur.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
15	Detected Parity Error	R/W	R/W	0	This bit is set to a '1' when the PCI Controller detects a parity error.
14	Signaled System Error	R/W	R/W	0	This bit is set to a '1' when the PCI Controller asserts SERR_B.
13	Received Master Abort	R/W	R/W	0	This bit is set to a '1' when the PCI Controller terminates its transaction with Master-Abort as master device.
12	Received Target Abort	R/W	R/W	0	This bit is set to a '1' when the PCI Controller receives Target-Abort and its transaction is terminated as master device.
11	Signaled Target Abort	R/W	R/W	0	This bit is set to a '1' when the PCI Controller terminates a transaction with Target-Abort as target device.
10:9	DEVSEL_B timing	R	R	01	Hardwired to '01', because the PCI Controller asserts DEVSEL_B at medium speed.
8	Master Data Parity Error	R/W	R/W	0	This bit is set when three conditions as the follows are met: (1) The bus agent asserted PERR_B itself (on a read) or observed PERR_B asserted (on a write) (2) The agent setting the bit acted as the bus master for the operation in which the error occurred (3) The Parity Error Response bit is set to a '1'
7	Fast Back-to-Back Capable	R	R	0	Hardwired to a '0', because the PCI Controller does not accept fast back-to-back transactions when the transactions to the different agents are required.
6	Reserved	R	R	0	Hardwired to a '0'.
5	66 MHz Enable	R	R	0	Hardwired to a '0', because the PCI Controller can be used at 33 MHz only.
4	Capabilities List	R	R	1	Hardwired to a '1', because the PCI Controller has the PPMI function as New Capability.
3:0	Reserved	R	R	0	Hardwired to '0's.

7.5.19.6 Revision ID register

This register specifies a device specific revision identifier.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
7:0	Revision ID	R	R	01H	Hardwired to '01H' that shows the revision number of the chip.

7.5.19.7 Class code register

This register is used to identify the generic function of the device.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
23:0	Class Code	R	R	000302H **Note**	Hardwired to '000302H' that shows the class code of the chip.

Note When class code register in the configuration space is read, the wrong value "000302H" (related VGA) is returned although "020300H" (ATM controller) should be returned as the class code of the μPD98502. Please change code on the Host driver side.

7.5.19.8 Cache line size register

This register specifies the system cache-line size in units of words (32-bit length). The value in this register is also used to determine whether to use Memory Read, Memory Read Line, Memory Read Multiple, Memory Write and Invalidate commands for accessing memory.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
7:0	Cache Line Size	R/W	R/W	0	The system cache-line size in units of words.

7.5.19.9 Latency timer register

This register specifies the value of the Latency Timer in units of PCI bus clocks. The bottom three bits are hardwired to "0"s, so that a timer granularity is eight-clocks.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
7:0	Latency Timer	R/W	R/W	0	The value of the Latency Timer in units of PCI bus clocks.

7.5.19.10 Header type register

This register identifies the layout of the second part of the predefined header and also whether or not the device contains multiple functions.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
7:0	Header Type	R	R	0	Hardwired to ‘00H’, because the PCI Controller is a single function device and not a PCI-PCI bridge.

7.5.19.11 Window memory base address register

This register specifies the base address of the PCI Memory space for the access window.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
31:0	Window Memory Base Address	R/W (upper 10 bits) R (lower 22 bits)	R/W (upper 10 bits) R (lower 22 bits)	0	Memory Base Address for the access window. Not all bits in this register are writeable. The lower 22 bits are hardwired to ‘0’ in order to indicate that the area with 2 MB is required in the 32-bit Memory Space.

7.5.19.12 Register memory base address register

This register specifies the base address of the PCI Memory space for the registers.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
31:0	Register Memory Base Address	R/W (upper 20 bits) R (lower 12 bits)	R/W (upper 20 bits) R (lower 12 bits)	0	Memory Base Address for the I/O registers. Not all bits in this register are writeable. The lower 12 bits are hardwired to ‘0’ in order to indicate that the area which size is 4 KB is required in the 32-bit Memory Space.

7.5.19.13 Subsystem vendor ID register

This register is used to uniquely identify the manufacturer of the expansion board or subsystem where the PCI device resides.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
15:0	Subsystem Vendor ID	R/W	R	0	The VR4120A should set the identifier to this register.

7.5.19.14 Subsystem ID register

This register is used to uniquely identify the expansion board or subsystem where the PCI device resides.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
15:0	Subsystem ID	R/W	R	0	The VR4120A should set the identifier to this register.

7.5.19.15 Cap_Ptr register

This register is used to show a linked list of new capabilities implemented by The PCI Controller. The PCI Controller has PPMI function as a new capability. Its data structure starts from ‘40H’ in the configuration register.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
7:0	Capabilities Pointer	R	R	40H	Hardwired to ‘40H’, where is the first term of the new capabilities list.

7.5.19.16 Interrupt line register

The Interrupt Line register is used to communicate interrupt line routing information.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
7:0	Interrupt Line	R/W	R/W	0	The value in this register shows the input of the system interrupt controller that the interrupt pin is connected.

7.5.19.17 Interrupt pin register

This register tells which interrupt pin the PCI Controller uses.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
7:0	Interrupt Pin	R	R	01H	Hardwired to ‘01H’, which means the PCI Controller uses INTA_B pin.

7.5.19.18 Min_Gnt register

This register specifies how long a burst period the PCI Controller needs assuming a clock rate of 33MHz.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
7:0	Min_Gnt	R/W	R	0	The user of the PCI Controller must prepare the value to this register. The value should be set by the VR4120A.

7.5.19.19 Max_Lat register

This register specifies how often the device needs to get the PCI bus usage.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
7:0	Max_Lat	R/W	R	0	The value should be set by the VR4120A.

7.5.19.20 Cap_ID register

This register indicates what kind of data structure of the capability is pointed to. The value '01H' means that the data structure is for the PCI Power Management.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
7:0	Cap_ID	R	R	01H	Hardwired to '01H', that identifies the linked list item as being the PCI Power Management registers.

7.5.19.21 Next_Item_Ptr register

This register points the next item in the capabilities list. NULL means that this item is the last one in the list.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
7:0	Next_Item_Ptr	R	R	0	Hardwired to '00H', because the PPMI function is the last item in the capabilities list for The PCI Controller.

7.5.19.22 PMC register

The Power Management Capabilities register provides information on the capabilities of the function related to power management.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
15:11	PME_Support	R/W	R	0	The VR4120A should set the bits of the power state in which the chip supports PME_B assertion.
10	D2_Support	R	R	0	Hardwired to a '0', because the PCI Controller does not support D2 state.
9	D1_Support	R/W	R	0	If the chip supports D1 state, this bit should be set by the VR4120A.
8:6	Aux_Current	R	R	000	Hardwired to '000', because the PCI Controller does not support PME# from D3cold state.
5	DSI	R	R	0	Hardwired to a '0', because the PCI Controller does not require the special initialization.
4	Reserved	R	R	0	Hardwired to a '0'.
3	PME Clock	R/W	R	0	Indicating Whether PCI clock is required or not, in order to generate PME_B. PME_B is generated using an 66MHz internal clock. The VR4120A must set a '1' to this bit.
2:0	Version	R	R	010	Hardwired to '010', which indicates that the PCI Controller complies with Revision 1.1 of the PPMI.

7.5.19.23 PMCSR register

This register is used to manage the PCI function’s power management state as well as to enable/monitor PME.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
15:10	PME_Staus	R	R/W	0	This bit is set when the PCI Controller asserts the PME_B signal independent of the PME_En bit. Writing a ‘1’ to this bit will clear it and cause the PCI Controller to stop asserting a PME_B (if enabled). Writing a ‘0’ has no effect.
14:13	Data_Scale	R	R	10	Hardwired to ‘10’, because the PCI Controller uses ‘0.01x’ as the unit of Data register.
12:9	Data_Select	R	R/W	0	This field is used to select which data is to be reported through the Data register and Data_Scale field. The definition of the field values is given below. ‘0H’ = D0 Power Consumed ‘1H’ = D1 Power Consumed ‘3H’ = D3 Power Consumed ‘4H’ = D0 Power Dissipated ‘5H’ = D1 Power Dissipated ‘7H’ = D3 Power Dissipated Other values = all ‘0’
8	PME_En	R	R/W	0	Set a ‘1’ enables the function to assert PME_B. When ‘0’, PME# assertion is disabled.
7:2	Reserved	R	R	0	Hardwired to ‘0’.
1:0	PowerState	R/W	R/W	0	This field is used both to determine the current power state and to set the PCI Controller into a new power state. The definition of the field values is given below. ’00’ = D0 ’01’ = D1 ‘11’ = D3 When other values are written to this field, the PCI Controller remains in the current power state.

7.5.19.24 PMData register

The PMData register is used to report the power consumed and the heat dissipation for each power state. The Data to be reported is selected by Data_Select field.

Bits	Field	R/W		Default	Description
		Internal bus	PCI		
7:0	PMData	R	R	00H	This register is used to report the state dependent data selected by the Data_Select field. The value of this register is scaled by the value reported in the Data_Scale field.

7.6 Information for Software

7.6.1 NIC mode

7.6.1.1 Initialization

(1) Initialization by the VR4120A

The PCI Controller issues "retry" to all accesses from PCI-side until INITD bit in P_BCNT register is set to '1'. Therefore, Initialization of the chip should be done before INITD bit is set to '1'.

The following sequence shows an example of initialization procedures required for the VR4120A.

- Sets Subsystem Vendor ID register, Subsystem ID register, Min_Gnt register and Max_Lat register in configuration space, if needed
- Sets '1' to "PME Clock" bit in PMC register, if PCI-clock does not be required to generate PME
- Sets base addresses to P_PLBA register and P_IBBA register
- Enables mask bits in P_IIMR register, if needed
- Sets data about power to P_PWCD register and P_PWDD register
- Sets commands the PCI Controller uses on Internal bus, I/O or memory, by ICMDS bit in P_BCNT register
- Selects modes for data transfers by PDRTD bit, PPWRD bit, IDRTD bit, IPWRD bit in P_BCNT register.
- Sets '00' to PowerState field in PMCSR register in order to indicate that chip can be run.
- Sets a '1' to PMRDY bit in P_PPCR register to indicate that the issue for the transition of power state is acceptable.
- Sets a '1' to INITD bit in P_BCNT register in order to indicate that the Initialization of the PCI Controller has been completed.

(2) Initialization by PCI-Host

After the time INITD bit is set, the PCI Controller can accepts the access from PCI-side. An external PCI-Host device is responsible to configure the configuration register of the PCI Controller so that the PCI Controller can run as PCI-device.

The following sequence shows an example of initialization procedures required for external PCI-Host device.

- Sets a '1' to "Memory Access Enable" bit in command register
- Sets a '1' to "Bus Master Enable" bit in command register, if the chip executes transaction as PCI-master
- Sets a '1' to "Memory Write and Invalidate Enable" bit in command register, if needed
- Sets a '1' to "Parity Error Response" bit in command register, if needed
- Sets a '1' to "System Error Response " bit in command register, if needed
- Sets the cache line size of system to "Cache Line Size" register
- Sets "Latency Timer" register, if needed
- Sets base addresses to "Window Base Memory Address" register and "Register Base Memory Address" register

- Sets a ‘1’ to PME_En bit in PMCSR register, if needed

Then, the PCI-Host device initializes internal registers.

- Sets the value of base address in P_IBBA register, if needed
- Enables mask bits in P_PIMR register, if needed
- Sets Retry Timer register, if needed

(3) Error

In the case that Error described in **7.2.3 Abnormal Termination** occurs, the PCI Controller sets bits in Status register of configuration space, P_IGSR register and P_PGSR register, and issues interrupts to the VR4120A and an external PCI-Host device (if not masked). The VR4120A and the external PCI-Host device are responsible for how to handle these error statuses. The PCI Controller would stop the current transaction, and returns to the state in which the PCI Controller can accept new accesses.

(4) Software Reset

When PCI-Host writes to P_SWRR register for Software Reset, the PCI Controller sets SRREQ bit in P_IGSR register and reports to the VR4120A by interrupt (if not masked). But the PCI Controller does not reset itself by this sequence.

The VR4120A should assert warm-reset signals or cold-reset signals for each blocks inside chip, if it wants to reset the chip.

(5) Transition of Power State

When PCI-Host writes to PowerState field in PMCSR register to change the power state of the chip, the PCI Controller resets PMRDY bit in P_PPCR register, sets PPREQ bit in P_IGSR register and reports to the VR4120A by interrupt (if not masked). What transition is required is indicated by PMRQ0 bit, PMRQ1 bit and PMRQ3 bit in PMCSR register.

However, the PCI Controller does not change the power state of the chip. The VR4120A should be responsible for the transition of the power state.

If the VR4120A wants the transition of power state, it can generate PME_B by writing ‘1’ to PMERQ bit in P_PPCR register.

7.6.2 Host mode

In Host mode, the host on PCI bus is the PCI Controller itself. This means that the VR4120A is responsible for initialization.

7.6.2.1 Initialization

In Host mode, the host on PCI bus is the PCI Controller itself. This means that the VR4120A is responsible for initialization.

The PCI Controller issues “retry” to all accesses from PCI-side until INITD bit in P_BCNT register is set to ‘1’.

The following sequence shows an example of configuration register initialization.

- Sets Subsystem Vendor ID register, Subsystem ID register, Min_Gnt register and Max_Lat register in configuration space
- Sets a ‘1’ to “Memory Access Enable” bit in command register

- Sets a ‘1’ to “Bus Master Enable” bit in command register, if the chip executes transaction as PCI-master
- Sets a ‘1’ to “Memory Write and Invalidate Enable” bit in command register, if needed
- Sets a ‘1’ to “Parity Error Response” bit in command register, if needed
- Sets a ‘1’ to “System Error Response ” bit in command register, if needed
- Sets a the cache line size of system to “Cache Line Size” register
- Sets a “Latency Timer” register, if needed
- Sets base addresses to “Window Base Memory Address” register and “Register Base Memory Address” register

Initialization for internal registers is as follows;

-Sets base addresses to P_PLBA register and P_IBBA register

-Enables mask bits in P_IIMR register, if needed

-Sets arbiter mode by PARBM bit in P_HMCR register, if needed

-Sets which command the PCI Controller uses on Internal bus, I/O or memory, by ICMDS bit in P_BCNT register

-Sets which the PCI Controller uses DAC on PCI bus or not, by DACEN bit in P_BCNT register

-Sets data transfer mode by PDRTD bit, PPWRD bit, IDRTD bit, IPWRD bit in P_BCNT register.

-Sets ‘00’ to PowerState field in PMCSR register in order to indicate that chip can be run.

-Sets a ‘1’ to INITD bit in P_BCNT register in order to indicate that the Initialization of the PCI Controller has been completed.

After the time INITD bit is set, the PCI Controller can accepts the access from PCI-side.

(1) Error

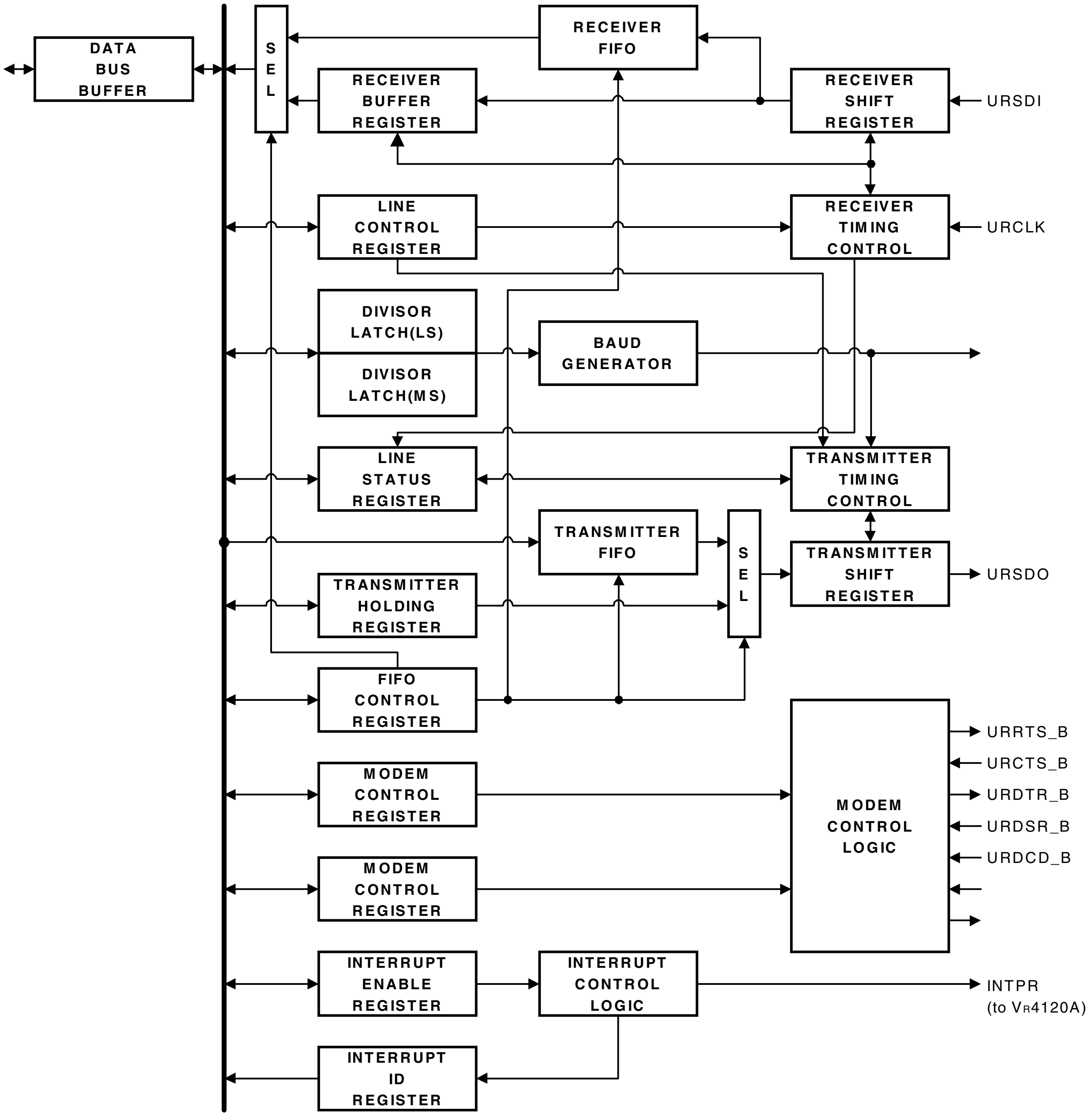
If Error described in **7.2.3 Abnormal Termination** occurs, the PCI Controller sets bits in Status register of configuration space, P_IGSR register and P_PGSR register, and issues interrupts to the VR4120A and an external PCI-Host device (if not masked). The VR4120A and PCI-Host are responsible for how to handle these error statuses. The PCI Controller would stop the current transaction, and returns to the state in which the PCI Controller can accept new accesses.

CHAPTER 8 UART

8.1 Overview

UART is a serial interface that conforms to the RS-232C communication standard and is equipped with two one-channel interfaces, one for transmission and one for reception. This unit is functionally compatible with the NS16550D.

8.2 UART Block Diagram



8.3 Registers

This controller uses the NEC NA16550L Mega-Function as its internal UART. This UART is functionally identical to the National Semiconductor NS16550D. Refer to the NEC "User's Manual. Mega FunctionNA16550L" for more information and programming details.

8.3.1 Register map

Offset Address	Register Name	R/W	Access	Description
1000_0080H	UARTRBR	R	W/H/B	UART, Receiver Buffer Register [DLAB=0,READ]
1000_0080H	UARTTHR	W	W/H/B	UART, Transmitter Holding Register [DLAB=0,WRITE]
1000_0080H	UARTDLL	R/W	W/H/B	UART, Divisor Latch LSB Register [DLAB=1]
1000_0084H	UARTIER	R/W	W/H/B	UART, Interrupt Enable Register [DLAB=0]
1000_0084H	UARTDLM	R/W	W/H/B	UART, Divisor Latch MSB Register [DLAB=1]
1000_0088H	UARTIIR	R	W/H/B	UART, Interrupt ID Register [READ]
1000_0088H	UARTFCR	W	W/H/B	UART, FIFO control Register [WRITE]
1000_008CH	UARTLCR	R/W	W/H/B	UART, Line control Register
1000_0090H	UARTMCR	R/W	W/H/B	UART, Modem Control Register
1000_0094H	UARTLSR	R/W	W/H/B	UART, Line status Register
1000_0098H	UARTMSR	R/W	W/H/B	UART, Modem Status Register
1000_009CH	UARTSCR	R/W	W/H/B	UART, Scratch Register

Remarks 1. In the "R/W" field,
"W" means "writeable",
"R" means "readable",
"RC" means "read-cleared",
"- " means "not accessible".

2. All internal registers are 32-bit word-aligned registers.

3. The burst access to the internal register is prohibited.
If such burst access has been occurred, IRERR bit in NSR is set and NMI will assert to CPU.

4. Read access to the reserved area will set the CBERR bit in the NSR register and the dummy read response data with the data-error bit set on SysCMD [0] is returned.

5. Write access to the reserved area will set the CBERR bit in the NSR register, and the write data is lost.

6. In the "Access" filed,
"W" means that word access is valid,
"H" means that half word access is valid,
"B" means that byte access is valid.

7. Write access to the read-only register cause no error, but the write data is lost.

8. The CPU can access all internal registers, but IBUS master device cannot access them.

8.3.2 UARTRBR (UART Receiver data Buffer Register)

This register holds receive data. It is only accessed when the Divisor Latch Access bit (DLAB) is cleared in the UARTLCR.

Bits	Field	R/W	Default	Description
31:8	Reserved	R	0	Hardwired to 0.
7:0	UDATA	R	-	UART receive data (read only) when DLAB = 0.

8.3.3 UARTTHR (UART Transmitter data Holding Register)

This register holds transmit data. It is only accessed when the Divisor Latch Access bit (DLAB) is cleared in the UARTLCR.

Bits	Field	R/W	Default	Description
31:8	Reserved	W	0	Hardwired to 0.
7:0	UDATA	W	-	UART transmit data (write only) when DLAB = 0.

8.3.4 UARTIER (UART Interrupt Enable Register)

This register is used to enable UART interrupts. It is only accessed when the Divisor Latch Access bit (DLAB) is set in the UARTLCR. The UARTIM (bit2 in Interrupt Mask Register “S_IMR”) is a global enable for interrupt sources enabled by this register.

Bits	Field	R/W	Default	Description
31:4	Reserved	R/W	0	Hardwired to 0.
3	ERBMI	R/W	0	UART Modem status Interrupts: 1 = Enable Modem status change interrupt 0 = Disable such interrupts Modem status changes are reported to UARTMSR
2	ERBLI	R/W	0	UART Line status Interrupts: 1 = Enable Line status error interrupt 0 = Disable such interrupts Line status error interrupt state reported to UARTLSR
1	ERBEI	R/W	0	UART Transmitter Buffer empty Interrupt: 1 = Enable Transmitter Buffer empty interrupt 0 = Disable such interrupts Transmitter Buffer empty state reported to UARTLSR

8.3.5 UARTDLL (UART Divisor Latch LSB Register)

This register is used to set the divisor (division rate) for the baud rate generator. The data in this register and the lower 8-bit data in UARTDLM register are together handled as 16-bit data.

Bits	Field	R/W	Default	Description
31:8	Reserved	R/W	0	Hardwired to 0.
7:0	DIVLSB	R/W	-	UART divisor latch (see **Table 8-1**): Only accessed when DLAB = 1 in UARTLCR

8.3.6 UARTDLM (UART Divisor Latch MSB Register)

This register is used to set the divisor (division rate) for the baud rate generator. The data in this register and the lower 8-bit data in UARTDLL register are together handled as 16-bit data.

Bits	Field	R/W	Default	Description
31:8	Reserved	R/W	0	Hardwired to 0.
7:0	DIVLSB	R/W	-	UART divisor latch (see **Table 8-1**): Only accessed when DLAB = 1 in UARTLCR

Table 8-1. Correspondence between Baud Rates and Divisors

Baud Rate	UART Source Clock Frequency					
[bps]	18.432 MHz (Use External Clock)		33.000 MHz (CPU Clock = 66 MHz)		50.000 MHz (CPU Clock = 100 MHz)	
	Divisor	Error	Divisor	Error	Divisor	Error
50	23040 (5A00H)	0	41250 (A122H)	>1 %	62500 (F424H)	0
75	15360 (3C00H)	0	27500 (6B6CH)	>1 %	41667 (A2C3H)	>1 %
110	10473 (28E9H)	0	18750 (493EH)	>1 %	28409 (6EF9H)	>1 %
134.5	8565 (2175H)	0	15335 (3BE7H)	>1 %	23234 (5AC2H)	>1 %
150	7680 (1E00H)	0	13750 (35B6H)	>1 %	20833 (5161H)	>1 %
300	3840 (F00H)	0	6875 (1ADBH)	>1 %	10417 (28B1H)	>1 %
600	1920 (780H)	0	3438 (D6EH)	>1 %	5208 (1458H)	>1 %
1200	920 (398H)	0	1719 (6B7H)	>1 %	2604 (A2CH)	>1 %
1800	640 (280H)	0	1146 (47AH)	>1 %	1736 (6C8H)	>1 %
2000	573 (23DH)	0	1031 (407H)	>1 %	1562 (61AH)	>1 %
2400	480 (1E0H)	0	859 (35BH)	>1 %	1302 (516H)	>1 %
3600	320 (140H)	0	573 (23DH)	>1 %	868 (364H)	>1 %
4800	240 (F0H)	0	430 (1AEH)	>1 %	651 (28BH)	>1 %
7200	160 (A0H)	0	286 (11EH)	>1 %	434 (1B2H)	>1 %
9600	120 (78H)	0	215 (D7H)	>1 %	326 (146H)	>1 %
19200	60 (3CH)	0	107 (6BH)	>1 %	163 (A3H)	>1 %
38400	30 (1EH)	0	54 (36H)	>1 %	81 (51H)	>1 %
56000	21 (15H)	2.04 %	37 (25H)	>1 %	56 (38H)	>1 %
128000	9 (9H)	0	16 (10H)	>1 %	24 (18H)	1.69 %
144000	8 (8H)	0	14 (EH)	2.25 %	22 (16H)	1.38 %
192000	6 (6H)	0	11 (BH)	2.41 %	16 (10H)	1.69 %
230400	5 (5H)	0	9 (9H)	>1 %	14 (EH)	3.22 %
288000	4 (4H)	0	7 (7H)	2.25 %	11 (BH)	1.38 %
384000	3 (3H)	0	5 (5H)	6.90 %	8 (8H)	1.69 %
576000	2 (2H)	0	4 (4H)	11.7 %	5 (5H)	7.83 %
1152000	1 (1H)	0	2 (2H)	11.7 %	3 (3H)	10.6 %

Remark If UCSEL bit in the S_GMR Register is set, The external UART clock “URCLK” is used as UART source clock.

If UCSEL bit is reset, 1/2 of CPU clock is used as UART source clock.

8.3.7 UARTIIR (UART Interrupt ID Register)

This register indicates priority levels for interrupts and existence of pending interrupt. From highest to lowest priority, these interrupts are receive line status, receive data ready, character timeout, transmit holding register empty, and modem status. The content of UARTIIR [3] bit is valid only in FIFO mode, and it is always 0 in 16550 mode. UARTIIR [2] bit becomes 1 when UARTIIR [3] bit is set to 1.

Bits	Field	R/W	Default	Description
31:8	Reserved	R	0	Hardwired to 0.
7:6	UFIFOEN	R	00	UART FIFO is enable (read only): Both bits set to 1 when the transmit/receive FIFO is enabled in the UFIFOEN0 bit is set in the UARTFCR.
5:4	Reserved	R	00	Hardwired to 0.
3:1	UIID	R	000	Indicates the priority level of pending interrupt: 011 = 1st Priority: Receiver Line status Overrun Error, Parity, Framing Error, or Break interrupt 010 = 2nd Priority: Received data available Receiver Data Available or Trigger Level Reached 110 = 3rd Priority: Character timeout indication No change in receiver FIFO during the last four character times and FIFO is not empty. 001 = 4th Priority: Transmitter holding Register Empty 000 = 5th Priority: Modem Status (CTS_L, DSR_L or DCD_L.)
0	INTPENDL	R	1	Pending interrupts 0 = UART Interrupt pending (read only) 1 = No UART interrupt pending

8.3.8 UARTFCR (UART FIFO Control Register)

This register is used to control the FIFOs: enable FIFO, clear FIFO, and set the receive FIFO trigger level.

Bits	Field	R/W	Default	Description
31:8	Reserved	W	0	Hardwired to 0.
7:6	URFTR	W	00	UART Receive FIFO Trigger level. When the trigger level is reached, a Receive-buffer-Full interrupt is generated, if enable by the ERBFI bit in the UARTIER. Number of bytes in Receive FIFO is following. 00 = 1 byte 01 = 4 bytes 10 = 8 bytes 11 = 14 bytes
5:4	Reserved	W	0	Hardwired to 0.
3	FIFOMD	W	0	Switch between 16550 mode and FIFO mode 1 = From 16550 mode to FIFO mode 0 = From FIFO mode to 16550 mode
2	UTFRST	W	0	UART Transmitter FIFO Reset. (write-only) 1 = clear transmit FIFO and reset counter. 0 = no clear.
1	URFRST	W	0	UART Receiver FIFO Reset. (write-only) 1 = clear receive FIFO and reset counter. 0 = no clear.
0	UFIFOEN0	W	0	UART Receiver FIFO Enable. (write-only): 1 = enable receive and transmit FIFOs. 0 = disable and clear receive and transmit FIFOs.

8.3.9 UARTLCR (UART Line Control Register)

This register is used to specify the format for asynchronous communication and exchange and to set the divisor latch access bit. Bit 6 is used to send the break status to the receive side's UART. When bit 6 = 1, the serial output (URSDO) is forcibly set to the spacing (0) state. The setting of bit 5 becomes valid according to settings in bits 4 and 3.

Bits	Field	R/W	Default	Description
31:8	Reserved	R/W	0	Hardwired to 0.
7	DLAB	R/W	0	Divisor Latch access bit. 1 = access baud-rate divisor at offset 84H 0 = access URSDO/URSDI and IE at offset 84H When this bit is set, UART accesses the UART Divisor Latch LSB Register (UARTDLM) at offset 84H. When cleared, the UART accesses the Receiver Data Buffer Register (UARTRBR) on reads at offset 80H, the UARTTHR on writes at offset 80H, and UARTIER on any accesses at offset 84H.
6	USB	R/W	0	Send Break 1 = force URSDO signal output Low 0 = normal operation
5	USP	R/W	0	Stick parity. 1 = force URSDO signal output Low 0 = normal operation
4	EPS	R/W	0	Parity select. 1 = even parity 0 = odd parity
3	PEN	R/W	0	Parity enable. 1 = generate parity on writes, check it on reads. 0 = no parity generation or checking. For the UART, even or odd parity can be generated or checked, as specified in Bit 4 (EPS).
2	STB	R/W	0	Number of stop bits. 1 = 2 bits, except 1.5 stop bits for 5bits / words (WLS = 00) 0 = 1 bit
1:0	WLS	R/W	00	Word length select. 11 = 8 bits 10 = 7 bits 01 = 6 bits 00 = 5 bits

8.3.10 UARTMCR (UART Modem Control Register)

This register controls the state of external URDTR_B and URRTS_B modem-control signals and of the loop-back test.

Bits	Field	R/W	Default	Description
31:5	Reserved	R/W	0	Hardwired to 0.
4	LOOP	R/W	0	Loop-Back Test. 1 = loop-back. 0 = normal operation. This is an NEC internal test function.
3	OUT2	R/W	0	Out 2 (internal signal). 1 = OUT2_B (internal) state active. 0 = OUT2_B (internal) state inactive (reset value). This is a user-defined bit that has no associated external signal. Software can write to the bit, but this has no effect.
2	OUT1	R/W	0	Out 1 (internal signal). 1 = OUT1_B (internal) state active. 0 = OUT1_B (internal) state inactive (reset value). This is a user-defined bit that has no associated external signal. Software can write to the bit, but this has no effect.
1	RTS	R/W	0	Request To Send. 1 = negate URRTS_B signal. 0 = assert URRTS_B signal.
0	DTR	R/W	0	Data Terminal Ready. 1 = negate URDTR_B signal. 0 = assert URDTR_B signal.

8.3.11 UARTLSR (UART Line Status Register)

This register reports the current state of the transmitter and receiver logic.

Bits	Field	R/W	Default	Description
31:8	Reserved	R/W	0	Hardwired to 0.
7	RFERR	R/W	0	Receiver FIFO Error. 1 = parity, framing, or break error in receiver buffer. 0 = no such error.
6	TEMT	R/W	1	Transmitter Empty. 1 = transmitter holding and shift registers empty. 0 = transmitter holding or shift register not empty.
5	THRE	R/W	1	Transmitter Holding Register Empty. 1 = transmitter holding register empty. 0 = transmitter holding register not empty. Transmit data is stored in the UART Transmitter Data Holding Register (UARTTHR)
4	BI	R/W	0	Break Interrupt. 1 = break received on URSDI signal. 0 = no break.
3	FE	R/W	0	Receive-Data Framing Error. 1 = framing error on receive data. 0 = no such error.
2	PE	R/W	0	Receive-Data Parity Error. 1 = parity error on receive data. 0 = no such error.
1	OE	R/W	0	Receive-Data Overrun Error. 1 = overrun error on receive data. 0 = no such error.
0	DR	R/W	0	Receive-Data Ready. 1 = receive data buffer full. 0 = receive data buffer not full. Receive data is stored in the UART Receiver Data Buffer Register (UARTRBR).

8.3.12 UARTMSR (UART Modem Status Register)

This register reports the current state of and changes in various control signals.

Bits	Field	R/W	Default	Description
31:8	Reserved	R/W	0	Hardwired to 0.
7	DCD	R/W	0	Data Carrier Detect. 1 =URDCD_B state active. 0 = URDCD_B state inactive. This bit is the complement of the URDCD_B input signal. If the LOOP bit in the UART Modem Control Register (UARTMCR), is set to 1, the DCD bit is equivalent to the OUT2 bit in the UARTMCR.
6	RI	R/W	0	Ring Indicator. 1 = not valid. 0 = always reads 0. This bit has no associated external signal.
5	DSR	R/W	0	Data Set Ready. 1 = URDSR_B state active. 0 = URDSR_B state inactive. This bit is the complement of the URDSR_B input signal. If the LOOP bit in the UART Modem Control Register (UARTMCR), is set to 1, the DSR bit is equivalent to the DTR bit in the UARTMCR.
4	CTS	R/W	0	Clear To Send. 1 =URCTS_B state active. 0 = URCTS_B state inactive. This bit is the complement of the URCTS_B input signal. If the LOOP bit in the UART Modem Control Register (UARTMCR), is set to 1, the CTS bit is equivalent to the RTS bit in the UARTMCR.
3	DDCD	R/W	0	Delta Data Carrier Detect. 1 = URDCD_B state changed since this register was last read. 0 = no such change.
2	TERI	R/W	0	Trailing Edge Ring Indicator. 1 = RI_B state changed since this register last read. 0 = no such change. RI_B is not implemented as an external signal, so this bit is never set by the controller.
1	DDSR	R/W	0	Delta Data Set Ready. 1 = URDSR_B input signal changed since this register was last read. 0 = no such change.
0	DCTS	R/W	0	Delta Clear To Send. 1 = URCTS_B state changed since this register was last read. 0 = no such change.

8.3.13 UARTSCR (UART Scratch Register)

This register contains a UART reset bit plus 8 bits of space for any software use.

Bits	Field	R/W	Default	Description
31:8	Reserved	R/W	0	Hardwired to 0.
7:0	USCR	R/W	-	UART Scratch Register. Available to software for any purpose.

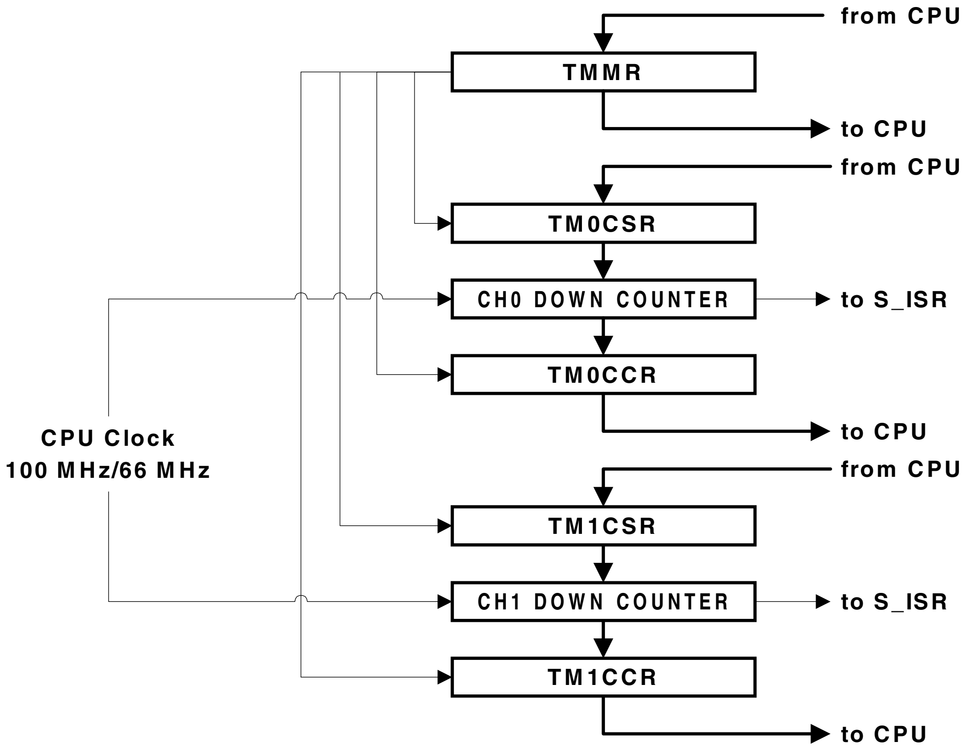
CHAPTER 9 TIMER

9.1 Overview

There are two Timers. The timers are clocked at the system clock rate. All two timers are read/writeable by the CPU. Timers can be read by the CPU while they are counting. They can be automatically reloaded with the "Timer Set Count Register" value and restarted. Two timers issues interrupt to the CPU upon reaching their maximum value, the interrupts can be enabled/disabled.

The TM0IS and TM1IS fields in the Interrupt Status Register "S_ISR" indicate the end of timer count, when set indicate there is a timer event that completed. All timers count down. The read-write registers "TM0CSR" or "TM1CSR" have different offset from the read register so write registers are not affected while a value is read from the read registers "TM0CCR"/"TM1CCR" which indicate a running count of the timer/counter at a given time. Once a value is loaded in the TM0CSR/TM1CSR, it stays there until Timer's interrupts are cleared in the Interrupt Status Register "S_ISR". The original value can be reloaded in the counter to restart it from that count if Timer CH0/CH1 reload enable bit is set in the Timer Mode Register "TMMR". Interrupts are automatically cleared upon CPU reading the Interrupt Status Register of System Controller "S_ISR".

9.2 Block Diagram



9.3 Registers

9.3.1 Register map

Offset Address	Register Name	R/W	Access	Description
1000_00B0H	TMMR	R/W	W/H/B	Timer Mode Register
1000_00B4H	TM0CSR	R/W	W/H/B	Timer CH0 Count Set Register
1000_00B8H	TM1CSR	R/W	W/H/B	Timer CH1 Count Set Register
1000_00BCH	TM0CCR	R	W/H/B	Timer CH0 Current Count Register
1000_00C0H	TM1CCR	R	W/H/B	Timer CH1 Current Count Register

Remarks 1. In the “R/W” field,
“W” means “writeable”,
“R” means “readable”,
“RC” means “read-cleared”,
“- “ means “not accessible”.

2. All internal registers are 32-bit word-aligned registers.

3. The burst access to the internal register is prohibited.
If such burst access has been occurred, IRERR bit in NSR is set and NMI will assert to CPU.

4. Read access to the reserved area will set the CBERR bit in the NSR register and the dummy read response data with the data-error bit set on SysCMD [0] is returned.

5. Write access to the reserved area will set the CBERR bit in the NSR register, and the write data is lost.

6. In the “Access” filed,
“W” means that word access is valid,
“H” means that half word access is valid,
“B” means that byte access is valid.

7. Write access to the read-only register cause no error, but the write data is lost.

8. The CPU can access all internal registers, but IBUS master device cannot access them.

9.3.2 TMMR (Timer Mode Register)

The Timer Mode Register “TMMR” is a read-write and 32-bit word-aligned register. TMMR is used to control the timer. TMMR is initialized to 0 at reset and contains the following fields:

Bits	Field	R/W	Default	Description
31:9	Reserved	R/W	0	Hardwired to 0.
8	TM1EN	R/W	0	Timer CH1 enable: 1 = Enable, starts the timer CH1 (Automatically reloads the original timer value and starts if timer had expired) 0 = Disable, stops the timer
7:1	Reserved	R/W	0	Hardwired to 0.
0	TM0EN	R/W	0	Timer CH0 enable: 1 = Enable, starts the timer CH0 (Automatically reloads the original timer value and starts if timer had expired) 0 = Disable, stops the timer

9.3.3 TM0CSR (Timer CH0 Count Set Register)

The Timer CH0 Count Set Register “TM0CSR” is a read-write and 32-bit word-aligned register. CPU (VR4120A) loads a value in it and the counter starts counting down from the (TM0CSR –1) value. When it reaches 0000_0000H, it generates an interrupt to the CPU via Interrupt Status Register “ISR” if the TM0IS in ISR is not masked by TM0IM in IMR. TM0CSR is initialized to 0 at reset and contains the following field:

Bits	Field	R/W	Default	Description
31:0	TM0SET	R/W	0	Initial and Reload Value for Timer CH0

9.3.4 TM1CSR (Timer CH1 Count Set Register)

The Timer CH1 Count Set Register “TM1CSR” is a read-write and 32-bit word-aligned register. CPU (VR4120A) loads a value in it and the counter starts counting down from the (TM1CSR –1) value. When it reaches 0000_0000H, it generates an interrupt to the CPU via Interrupt Status Register “ISR” if the TM1IS in ISR is not masked by TM1IM in IMR. TM1CSR is initialized to 0 at reset and contains the following field:

Bits	Field	R/W	Default	Description
31:0	TM1SET	R/W	0	Initial and Reload Value for Timer CH1

9.3.5 TM0CCR (Timer CH0 Current Count Register)

The Timer CH0 Current Count Register “TM0CCR” is read-only and 32-bit word-aligned register. CPU (VR4120A) can read its value to get timer CH0 current count. TM0CSR is initialized to FFFF_FFFFH at reset and contains the following field:

Bits	Field	R/W	Default	Description
31:0	TM0CNT	R	FFFF_ FFFFH	Timer CH0 Current Count Value

9.3.6 TM1CCR (Timer CH1 Current Count Register)

The Timer CH1 Current Count Register “TM1CCR” is read-only and word aligned 32bit register. CPU (VR4120A) can read its value to get timer CH1 current count. TM1CCR is initialized to FFFF_FFFFH at reset and contains the following fields:

Bits	Field	R/W	Default	Description
31:0	TM1CCR	R	FFFF_ FFFFH	Timer CH1 Current Count Value

CHAPTER 10 MICRO WIRE

10.1 Overview

This EEPROM interface is compatible with the Micro Wire serial interface. Connection to the “NM93C46” serial EEPROM, manufactured by National Semiconductor, is recommended.

Serial EEPROM memory area is accessed in-directly throghout Micro Wire-macro registers, that is ECCR and ERDR registers. To access the EEPROM, the VR4120A writes a command into the ECCR register of Micro Wire-macro. When Micro Wire-macro accepts the command, it executes the command via the EEPROM interface. To read EEPROM data, the VR4120A sets an address and READ command into the ECCR register. When the microwire-macro is reading data, the MSB bit of the ERDR register is set to 1. Once the microwire-macro finishes reading the data, it sets the MSB bit to 0 and stores the data in the EDAT field. After issuing the command, the VR4120A checks that the MSB bit of the ERDR register is set to 0, then obtains the data. To write data into or erase data from the EEPROM, the VR4120A must enable write and erase operations using the EWEN command in advance. When no EEPROM is connected, accessing these registers is meaningless.

This Micro Wire interface has also auto-load function. By this function, user can read 12-byte data of EEPROM (1H to 6H of half-word unit) throughout MACAR1 to MACAR3 registers without ECCR register’s control. This auto-load function works one-time after system boot. In addition, it is necessary for auto-loading to obey initial data format for EEPROM (refer to under table).

During both auto-loading or loading by ECCR register, MSB bit of ERDR register is asserted to ‘1’ for flag of busy state. At the end of EEPROM loading, MSB bit of ERDR register is de-asserted to ‘0’, and then USER can read MACAR1 to MACAR3 registers or [15:0] field of ERDR register.

10.2 Operations

10.2.1 Data read at the power up load

After reset release, power up load processes starts.

In case of the value from EEPROM address 00H is:

1. A5A5H

   System Controller sets the EEPROM data (address: 01H to 06H) in the internal registers (MACAR1, MACAR2, MACAR3).

2. NOT A5A5H

   System Controller sets the fix data “0000_0000H” in the internal registers (MACAR1, MACAR2, MACAR3).

Table 10-1. EEPROM Initial Data

EEPROM Address	Data	Stored Register
00H	A5A5H	-
01H	MAC1 Address data[15:0]	MACAR1[15:0]
02H	MAC1 Address data[31:16]	MACAR1[31:16]
03H	MAC1 Address data[47:32]	MACAR2[15:0]
04H	MAC2 Address data[15:0]	MACAR2[31:16]
05H	MAC2 Address data[31:16]	MACAR3[15:0]
06H	MAC2 Address data[47:32]	MACAR3[31:16]

10.2.2 Accessing to EEPROM

Access to EEPROM starts by writing to the ECCR (EEPROM Command Control Register).

Write command (3 bits) and address (6 bits) of EEPROM into lower 9 bits of ECCR.

There is a difference between write command and read command.

1. Write command

   Write data into upper 16 bits of ECCR.

2. Read command

   Data is loaded into lower 16bits of ERDR (EEPROM Read Data Register).

Table 10-2. EEPROM Command List

Command	bits 8:6	bits 5:0	Operation
READ	110	A5:A0	Read data from EEPROM assigned by address A5:A0
EWEN	100	11xxxx	Enable Erase/Write command. This command must be executed before other operating.
ERASE	111	A5:A0	Erase data of EEPROM assigned by address A5:A0
WRITE	101	A5:A0	Write data to EEPROM assigned by address A5:A0
ERAL	100	10xxxx	Erase all data of EEPROM.
WRAL	100	01xxxx	Write all data to EEPROM.
EWDS	100	00xxxx	Disable Erase/Write command.

10.3 Registers

10.3.1 Register map

Offset Address	Register Name	R/W	Access	Description
1000_00D0H	ECCR	W	W/H/B	EEPROM Command Control Register
1000_00D4H	ERDR	R	W/H/B	EEPROM Read Data Register
1000_00D8H	MACAR1	R	W/H/B	MAC Address Register 1
1000_00DCH	MACAR2	R	W/H/B	MAC Address Register 2
1000_00E0H	MACAR3	R	W/H/B	MAC Address Register 3

10.3.2 ECCR (EEPROM Command Control Register)

Bits	Field	R/W	Default	Description
31:16	DATA	W	0	Write data to Serial EEPROM. No meaning in case of data read.
15:9	Reserved	W	0	Reserved
8:6	CMD	W	0	Serial EEPROM command.
5:0	ADDRESS	W	0	Serial EEPROM address.

10.3.3 ERDR (EEPROM Read Data Register)

Bits	Field	R/W	Default	Description
31	B	R	1	Operation status of Micro Wire block. 1: busy 0: idle
30:16	Reserved	R	0	Reserved
15:0	READ DATA	R	0	Read data from Serial EEPROM.

10.3.4 MACAR1 (MAC Address Register 1)

Bits	Field	R/W	Default	Description
31:16	SERIAL EEPROM 02H ADDRESS	R	0	Stored Serial EEPROM data of address 01H, 02H.
15:0	SERIAL EEPROM 01H ADDRESS	R	0	

10.3.5 MACAR2 (MAC Address Register 2)

Bits	Field	R/W	Default	Description
31:16	SERIAL EEPROM 04H ADDRESS	R	0	Stored Serial EEPROM data of address 03H, 04H.
15:0	SERIAL EEPROM 03H ADDRESS	R	0	

10.3.6 MACAR3 (MAC Address Register 3)

Bits	Field	R/W	Default	Description
31:16	SERIAL EEPROM 06H ADDRESS	R	0	Stored Serial EEPROM data of address 05H, 06H.
15:0	SERIAL EEPROM 05H ADDRESS	R	0	

APPENDIX A MIPS III INSTRUCTION SET DETAILS

This chapter provides a detailed description of the operation of each instruction in both 32- and 64-bit modes. The instructions are listed in alphabetical order.

A.1 Instruction Notation Conventions

In this chapter, all variable subfields in an instruction format (such as *rs*, *rt*, *immediate*, etc.) are shown in lowercase names.

For the sake of clarity, we sometimes use an alias for a variable subfield in the formats of specific instructions. For example, we use *base* instead of *rs* in the format for load and store instructions. Such an alias is always lower case, since it refers to a variable subfield.

Figures with the actual bit encoding for all the mnemonics are located at the end of this chapter (**A.6 CPU Instruction Opcode Bit Encoding**), and the bit encoding also accompanies each instruction.

In the instruction descriptions that follow, the operation section describes the operation performed by each instruction using a high-level language notation. The VR4120A CPU can operate as either a 32- or 64-bit microprocessor and the operation for both modes is included with the instruction description.

Special symbols used in the notation are described in Table A-1.

Table A-1. CPU Instruction Operation Notations

Symbol	Description
←	Assignment
||	Bit string concatenation
x^y	Replication of bit value *x* into a *y*-bit string. *x* is always a single-bit value
xy:z	Selection of bits *y* through *z* of bit string *x*. Little-endian bit notation is always used. If *y* is less than *z*, this expression is an empty (zero length) bit string
+	2's complement or floating-point addition
-	2's complement or floating-point subtraction
*	2's complement or floating-point multiplication
div	2's complement integer division
mod	2's complement modulo
/	Floating-point division
<	2's complement less than comparison
and	Bit-wise logical AND
or	Bit-wise logical OR
xor	Bit-wise logical XOR
nor	Bit-wise logical NOR
GPR [*x*]	General-Register *x*. The content of GPR [0] is always zero. Attempts to alter the content of GPR [0] have no effect.
CPR [*z, x*]	Coprocessor unit *z*, general register *x*.
CCR [*z, x*]	Coprocessor unit *z*, control register *x*.
COC [*z*]	Coprocessor unit *z* condition signal.
BigEndianMem	Big-endian mode as configured at reset (0 → Little, 1 → Big). Specifies the endianness of the memory interface (see LoadMemory and StoreMemory), and the endianness of Kernel and Supervisor mode execution. However, this value is always 0 since the VR4120A CPU supports the little endian order only.
ReverseEndian	Signal to reverse the endianness of load and store instructions. This feature is available in User mode only, and is effected by setting the RE bit of the Status register. Thus, ReverseEndian may be computed as (SR25 and User mode).However, this value is always 0 since the VR4120A CPU supports the little endian order only.
BigEndianCPU	The endianness for load and store instructions (0 → Little, 1 → Big). In User mode, this endianness may be reversed by setting RE bit. Thus, BigEndianCPU may be computed as BigEndianMem XOR ReverseEndian.However, this value is always 0 since the VR4120A CPU supports the little endian order only.
T + *i*:	Indicates the time steps between operations. Each of the statements within a time step are defined to be executed in sequential order (as modified by conditional and loop constructs). Operations which are marked T + *i*: are executed at instruction cycle *i* relative to the start of execution of the instruction. Thus, an instruction which starts at time *j* executes operations marked T + *i:* at time *i + j*. The interpretation of the order of execution between two instructions or two operations that execute at the same time should be pessimistic; the order is not defined.

(1) Instruction notation examples

The following examples illustrate the application of some of the instruction notation conventions:

Example 1:

$$\text{GPR [rt]} \leftarrow \text{immediate} \parallel 0^{16}$$

Sixteen zero bits are concatenated with an immediate value (typically 16 bits), and the 32-bit string is assigned to general register *rt*.

Example 2:

$$(\text{immediate}_{15})^{16} \parallel \text{immediate}_{15...0}$$

Bit 15 (the sign bit) of an immediate value is extended for 16-bit positions, and the result is concatenated with bits 15 through 0 of the immediate value to form a 32-bit sign extended value.

A.2 Load and Store Instructions

In the VR4120A CPU implementation, the instruction immediately following a load may use the loaded contents of the register. In such cases, the hardware interlocks, requiring additional real cycles, so scheduling load delay slots is still desirable, although not required for functional code.

In the load and store descriptions, the functions listed in Table A-2 are used to summarize the handling of virtual addresses and physical memory.

Table A-2. Load and Store Common Functions

Function	Description
AddressTranslation	Uses the TLB to find the physical address given the virtual address. The function fails and an exception is taken if the required translation is not present in the TLB.
LoadMemory	Uses the cache and main memory to find the contents of the word containing the specified physical address. The low-order three bits of the address and the Access Type field indicate which of each of the four bytes within the data word need to be returned. If the cache is enabled for this access, the entire word is returned and loaded into the cache. If the specified data is short of word length, the data position to which the contents of the specified data is stored is determined considering the endian mode and reverse endian mode.
StoreMemory	Uses the cache, write buffer, and main memory to store the word or part of word specified as data in the word containing the specified physical address. The low-order three bits of the address and the Access Type field indicate which of each of the four bytes within the data word should be stored. If the specified data is short of word length, the data position to which the contents of the specified data is stored is determined considering the endian mode and reverse endian mode.

As shown in Table A-3, the Access Type field indicates the size of the data item to be loaded or stored. Regardless of access type or byte-numbering order (endian), the address specifies the byte that has the smallest byte address in the addressed field. This is the rightmost byte in the VR4120A CPU since it supports the little-endian order only.

Table A-3. Access Type Specifications for Loads/Stores

Access Type	Description
DOUBLEWORD	8 bytes (64 bits)
SEPTIBYTE	7 bytes (56 bits)
SEXTIBYTE	6 bytes (48 bits)
QUINTIBYTE	5 bytes (40 bits)
WORD	4 bytes (32 bits)
TRIPLEBYTE	3 bytes (24 bits)
HALFWORD	2 bytes (16 bits)
BYTE	1 byte (8 bits)

The bytes within the addressed doubleword that are used can be determined directly from the access type and the three low-order bits of the address.

A.3 Jump and Branch Instructions

All jump and branch instructions have an architectural delay of exactly one instruction. That is, the instruction immediately following a jump or branch (that is, occupying the delay slot) is always executed while the target instruction is being fetched from storage. A delay slot may not itself be occupied by a jump or branch instruction; however, this error is not detected and the results of such an operation are undefined.

If an exception or interrupt prevents the completion of a legal instruction during a delay slot, the hardware sets the EPC register to point at the jump or branch instruction that precedes it. When the code is restarted, both the jump or branch instructions and the instruction in the delay slot are reexecuted.

Because jump and branch instructions may be restarted after exceptions or interrupts, they must be restartable. Therefore, when a jump or branch instruction stores a return link value, register *r31* (the register in which the link is stored) may not be used as a source register.

Since instructions must be word-aligned, a Jump Register or Jump and Link Register instruction must use a register which contains an address whose two low-order bits (low-order one bit in the 16-bit mode) are zero. If these low-order bits are not zero, an address exception will occur when the jump target instruction is subsequently fetched.

A.4 System Control Coprocessor (CP0) Instructions

There are some special limitations imposed on operations involving CP0 that is incorporated within the CPU. Although load and store instructions to transfer data to/from coprocessors and to move control to/from coprocessor instructions are generally permitted by the MIPS architecture, CP0 is given a somewhat protected status since it has responsibility for exception handling and memory management. Therefore, the move to/from coprocessor instructions are the only valid mechanism for writing to and reading from the CP0 registers.

Several CP0 instructions are defined to directly read, write, and probe TLB entries and to modify the operating modes in preparation for returning to User mode or interrupt-enabled states.

A.5 CPU Instruction

This section describes the functions of CPU instructions in detail for both 32-bit address mode and 64-bit address mode.

The exception that may occur by executing each instruction is shown in the last of each instruction's description. For details of exceptions and their processes, see **Section 2.5 Exception Processing**.

ADD Add ADD

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	rd	0 0 0 0 0 0	ADD 1 0 0 0 0 0
6	5	5	5	5	6

Format:

ADD rd, rs, rt

Description:

The contents of general register *rs* and the contents of general register *rt* are added to form the result. The result is placed into general register *rd*. In 64-bit mode, the operands must be valid sign-extended, 32-bit values.

An overflow exception occurs if the carries out of bits 30 and 31 differ (2's complement overflow). The destination register *rd* is not modified when an integer overflow exception occurs.

Operation:

```
32    T:    GPR [rd] ← GPR [rs] + GPR [rt]

64    T:    temp ← GPR [rs] + GPR [rt]
            GPR [rd] ← $(temp_{31})^{32}$ || $temp_{31...0}$
```

Exceptions:

Integer overflow exception

ADDI Add Immediate ADDI

31 26	25 21	20 16	15 0
ADDI 0 0 1 0 0 0	rs	rt	immediate
6	5	5	16

Format:

ADDI rt, rs, immediate

Description:

The 16-bit *immediate* is sign-extended and added to the contents of general register *rs* to form the result. The result is placed into general register *rt.* In 64-bit mode, the operand must be valid sign-extended, 32-bit values.

An overflow exception occurs if carries out of bits 30 and 31 differ (2's complement overflow). The destination register *rt* is not modified when an integer overflow exception occurs.

Operation:

```
32   T:   $GPR\ [rt] \leftarrow GPR\ [rs] + (immediate_{15})^{16} \,||\, immediate_{15...0}$

64   T:   $temp \leftarrow GPR\ [rs] + (immediate_{15})^{48} \,||\, immediate_{15...0}$
          $GPR\ [rt] \leftarrow (temp_{31})^{32} \,||\, temp_{31...0}$
```

Exceptions:

Integer overflow exception

ADDIU Add Immediate Unsigned ADDIU

31 26	25 21	20 16	15 0
ADDIU 0 0 1 0 0 1	rs	rt	immediate
6	5	5	16

Format:

ADDIU rt, rs, immediate

Description:

The 16-bit *immediate* is sign-extended and added to the contents of general register *rs* to form the result. The result is placed into general register *rt.* No integer overflow exception occurs under any circumstances. In 64-bit mode, the operand must be valid sign-extended, 32-bit values.

The only difference between this instruction and the ADDI instruction is that ADDIU never causes an integer overflow exception.

Operation:

```
32    T:    GPR [rt] ← GPR [rs] + $(\text{immediate}_{15})^{16}$ || $\text{immediate}_{15...0}$

64    T:    temp ←GPR [ rs] + $(\text{immediate}_{15})^{48}$ || $\text{immediate}_{15...0}$
            GPR [rt] ← $(\text{temp}_{31})^{32}$ || $\text{temp}_{31...0}$
```

Exceptions:

None

ADDU Add Unsigned ADDU

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	rd	0 0 0 0 0 0	ADDU 1 0 0 0 0 1
6	5	5	5	5	6

Format:

ADDU rd, rs, rt

Description:

The contents of general register *rs* and the contents of general register *rt* are added to form the result. The result is placed into general register *rd*. No integer overflow exception occurs under any circumstances. In 64-bit mode, the operands must be valid sign-extended, 32-bit values.

The only difference between this instruction and the ADD instruction is that ADDU never causes an integer overflow exception.

Operation:

```
32   T:   GPR [rd] ← GPR [rs] + GPR [rt]

64   T:   temp ← GPR [rs] + GPR [rt]
          GPR [rd] ← $(temp_{31})^{32}$ || $temp_{31...0}$
```

Exceptions:

None

AND And AND

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	rd	0 0 0 0 0 0	AND 1 0 0 1 0 0
6	5	5	5	5	6

Format:

AND rd, rs, rt

Description:

The contents of general register *rs* are combined with the contents of general register *rt* in a bit-wise logical AND operation. The result is placed into general register *rd*.

Operation:

```
32   T:   GPR [rd] ←GPR [rs] and GPR [rt]

64   T:   GPR [rd] ← GPR [rs] and GPR [rt]
```

Exceptions:

None

ANDI And Immediate ANDI

31 26	25 21	20 16	15 0
ANDI 0 0 1 1 0 0	rs	rt	immediate
6	5	5	16

Format:

ANDI rt, rs, immediate

Description:

The 16-bit *immediate* is zero-extended and combined with the contents of general register *rs* in a bit-wise logical AND operation. The result is placed into general register *rt*.

Operation:

```
32   T:   GPR [rt] ← 0^16 || (immediate and GPR [rs]15...0)

64   T:   GPR [rt] ← 0^48 || (immediate and GPR [rs]15...0)
```

Exceptions:

None

BC0F Branch On Coprocessor 0 False BC0F

31 26	25 21	20 16	15 0
COPz 0 1 0 0 X X Note	BC 0 1 0 0 0	BCF 0 0 0 0 0	offset
6	5	5	16

Format:

BC0F offset

Description:

A branch target address is computed from the sum of the address of the instruction in the delay slot and the 16-bit *offset*, shifted left two bits and sign-extended. If contents of CP0's condition signal (CpCond), as sampled during the previous instruction, is false, then the program branches to the target address with a delay of one instruction.

Because the condition line is sampled during the previous instruction, there must be at least one instruction between this instruction and a coprocessor instruction that changes the condition line.

Operation:

```
32   T-1:  condition ← not $SR_{18}$
     T:    target ← $(offset_{15})^{14}$ || offset || $0^2$
     T+1:  if condition then
                PC ← PC + target
           endif

64   T-1:  condition ← not $SR_{18}$
     T:    target ← $(offset_{15})^{46}$ || offset || $0^2$
     T+1:  if condition then
                PC ← PC + target
           endif
```

Exceptions:

Coprocessor unusable exception

Note See the opcode table below, or **A.6 CPU Instruction Opcode Bit Encoding**.

Opcode Table:

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	0
BC0F	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
	Opcode				Coprocessor number		BC sub-opcode					Branch condition					

BC0FL Branch On Coprocessor 0 False Likely (1/2) BC0FL

31 26	25 21	20 16	15 0
COPz 0 1 0 0 X X [Note]	BC 0 1 0 0 0	BCFL 0 0 0 1 0	offset
6	5	5	16

Format:

BC0FL offset

Description:

A branch target address is computed from the sum of the address of the instruction in the delay slot and the 16-bit *offset*, shifted left two bits and sign-extended. If the contents of CP0's condition (CpCond) line, as sampled during the previous instruction, is false, the target address is branched to with a delay of one instruction.

If the conditional branch is not taken, the instruction in the branch delay slot is nullified.

Because the condition line is sampled during the previous instruction, there must be at least one instruction between this instruction and a coprocessor instruction that changes the condition line.

Operation:

```
32    T-1:   condition ← not $SR_{18}$
      T:     target ← $(\text{offset}_{15})^{14}$ || offset || $0^2$
      T+1:   if condition then
                  PC ← PC + target
             else
                  NullifyCurrentInstruction
             endif

64    T-1:   condition ← not SR
      T:     target ← $(\text{offset}_{15})^{46}$ || offset || $0^2$
      T+1:   if condition then
                  PC ← PC + target
             else
                  NullifyCurrentInstruction
             endif
```

Exceptions:

Coprocessor unusable exception

Note See the opcode table below, or **A.6 CPU Instruction Opcode Bit Encoding**.

BC0FL Branch On Coprocessor 0 False Likely (2/2) BC0FL

Opcode Table:

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	0
BC0FL	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	
	Opcode				Coprocessor number		BC sub-opcode					Branch condition					

BC0T Branch On Coprocessor 0 True BC0T

31 26	25 21	20 16	15 0
COPz 0 1 0 0 X X Note	BC 0 1 0 0 0	BCT 0 0 0 0 1	offset
6	5	5	16

Format:

BC0T offset

Description:

A branch target address is computed from the sum of the address of the instruction in the delay slot and the 16-bit *offset*, shifted left two bits and sign-extended. If the contents of CP0's condition signal (CpCond) is true, then the program branches to the target address, with a delay of one instruction.

Because the condition line is sampled during the previous instruction, there must be at least one instruction between this instruction and a coprocessor instruction that changes the condition line.

Operation:

```
32   T-1:  condition ← $SR_{18}$
     T:    target ← $(offset_{15})^{14}$ || offset || $0^2$
     T+1:  if condition then
                PC ← PC + target
           endif

64   T-1:  condition ← $SR_{18}$
     T:    target ← $(offset_{15})^{46}$ || offset || $0^2$
     T+1:  if condition then
                PC ← PC + target
           endif
```

Exceptions:

Coprocessor unusable exception

Note See the opcode table below, or **A.6 CPU Instruction Opcode Bit Encoding**.

Opcode Table:

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	0
BC0T	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	
	Opcode				Coprocessor number		BC sub-opcode					Branch condition					

BC0TL Branch On Coprocessor 0 True Likely (1/2) BC0TL

31 26	25 21	20 16	15 0
COPz 0 1 0 0 X X Note	BC 0 1 0 0 0	BCTL 0 0 0 1 1	offset
6	5	5	16

Format:

BC0TL offset

Description:

A branch target address is computed from the sum of the address of the instruction in the delay slot and the 16-bit *offset*, shifted left two bits and sign-extended. If the contents of CP0's condition (CpCond) line, as sampled during the previous instruction, is true, the target address is branched to with a delay of one instruction.

If the conditional branch is not taken, the instruction in the branch delay slot is nullified.

Because the condition line is sampled during the previous instruction, there must be at least one instruction between this instruction and a coprocessor instruction that changes the condition line.

Operation:

```
32   T-1:  condition ← $SR_{18}$
     T:    target ← $(offset_{15})^{14}$ || offset || $0^2$
     T+1:  if condition then
                 PC ← PC + target
           else
                 NullifyCurrentInstruction
           endif

64   T-1:  condition ← $SR_{18}$
     T:    target ← $(offset_{15})^{46}$ || offset || $0^2$
     T+1:  if condition then
                 PC ← PC + target
           else
                 NullifyCurrentInstruction
           endif
```

Exceptions:

Coprocessor unusable exception

Note See the opcode table below, or **A.6 CPU Instruction Opcode Bit Encoding**.

BC0TL Branch On Coprocessor 0 True Likely (2/2) BC0TL

Opcode Table:

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	0
BC0TL	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	1	
	Opcode				Coprocessor number		BC sub-opcode					Branch condition					

BEQ Branch On Equal BEQ

31 26	25 21	20 16	15 0
BEQ 0 0 0 1 0 0	rs	rt	offset
6	5	5	16

Format:

BEQ rs, rt, offset

Description:

A branch target address is computed from the sum of the address of the instruction in the delay slot and the 16-bit *offset*, shifted left two bits and sign-extended. The contents of general register *rs* and the contents of general register *rt* are compared. If the two registers are equal, then the program branches to the target address, with a delay of one instruction.

Operation:

```
32    T:      target ← $(\text{offset}_{15})^{14}$ || offset || $0^2$
              condition ← (GPR [rs] = GPR [rt])
      T+1:    if condition then
                  PC ← PC + target
              endif

64    T:      target ← $(\text{offset}_{15})^{46}$ || offset || $0^2$
              condition ← (GPR [rs] = GPR [rt])
      T+1:    if condition then
                  PC ← PC + target
              endif
```

Exceptions:

None

BEQL Branch On Equal Likely BEQL

31 26	25 21	20 16	15 0
BEQL 0 1 0 1 0 0	rs	rt	offset
6	5	5	16

Format:

BEQL rs, rt, offset

Description:

A branch target address is computed from the sum of the address of the instruction in the delay slot and the 16-bit offset, shifted left two bits and sign-extended. The contents of general register *rs* and the contents of general register *rt* are compared. If the two registers are equal, the target address is branched to, with a delay of one instruction. If the conditional branch is not taken, the instruction in the branch delay slot is nullified.

Operation:

```
32  T:    target ← $(\text{offset}_{15})^{14}$ || offset || $0^2$
          condition ← (GPR [rs] = GPR [rt])
    T+1:  if condition then
                PC ← PC + target
          else
                NullifyCurrentInstruction
          endif

64  T:    target ← $(\text{offset}_{15})^{46}$ || offset || $0^2$
          condition ← (GPR [rs] = GPR [rt])
    T+1:  if condition then
                PC ← PC + target
          else
                NullifyCurrentInstruction
          endif
```

Exceptions:

None

BGEZ Branch On Greater Than Or Equal To Zero BGEZ

31 26	25 21	20 16	15 0
REGIMM 0 0 0 0 0 1	rs	BGEZ 0 0 0 0 1	offset
6	5	5	16

Format:

BGEZ rs, offset

Description:

A branch target address is computed from the sum of the address of the instruction in the delay slot and the 16-bit *offset*, shifted left two bits and sign-extended. If the contents of general register *rs* are zero or greater when compared to zero, then the program branches to the target address, with a delay of one instruction.

Operation:

```
32   T:     target ← $(\text{offset}_{15})^{14}$ || offset || $0^2$
            condition ← (GPR $[\text{rs}]_{31}$ = 0)
     T+1:   if condition then
                 PC ← PC + target
            endif

64   T:     target ← $(\text{offset}_{15})^{46}$ || offset || $0^2$
            condition ← (GPR $[\text{rs}]_{63}$ = 0)
     T+1:   if condition then
                 PC ← PC + target
            endif
```

Exceptions:

None

BGEZAL Branch On Greater Than Or Equal To Zero And Link BGEZAL

31 26	25 21	20 16	15 0
REGIMM 0 0 0 0 0 1	rs	BGEZAL 1 0 0 0 1	offset
6	5	5	16

Format:

BGEZAL rs, offset

Description:

A branch target address is computed from the sum of the address of the instruction in the delay slot and the 16-bit *offset,* shifted left two bits and sign-extended. Unconditionally, the address of the instruction after the delay slot is placed in the link register, *r31*. If the contents of general register *rs* are zero or greater when compared to zero, then the program branches to the target address, with a delay of one instruction.

General register *rs* may not be general register *r31*, because such an instruction is not restartable. An attempt to execute this instruction is not trapped, however.

Operation:

```
32   T:    target ← $(\text{offset}_{15})^{14}$ || offset || $0^2$
           condition ← $(\text{GPR [rs]}_{31} = 0)$
           GPR [31] ← PC + 8
     T+1:  if condition then
                PC ← PC + target
           endif

64   T:    target ← $(\text{offset}_{15})^{46}$ || offset || $0^2$
           condition ← $(\text{GPR [rs]}_{63} = 0)$
           GPR [31] ← PC + 8
     T+1:  if condition then
                PC ← PC + target
           endif
```

Exceptions:

None

BGEZALL Branch On Greater Than Or Equal To Zero And Link Likely BGEZALL

31 26	25 21	20 16	15 0
REGIMM 0 0 0 0 0 1	rs	BGEZALL 1 0 0 1 1	offset
6	5	5	16

Format:

BGEZALL rs, offset

Description:

A branch target address is computed from the sum of the address of the instruction in the delay slot and the 16-bit *offset,* shifted left two bits and sign-extended. Unconditionally, the address of the instruction after the delay slot is placed in the link register, *r31*. If the contents of general register *rs* are zero or greater when compared to zero, then the program branches to the target address, with a delay of one instruction. General register *r31* should not be specified as general register *rs*. If register *r31* is specified, restarting may be impossible due to the destruction of *rs* contents caused by storing a link address. Even such instructions are executed, an exception does not result.

Operation:

```
32   T:     target ← $(\text{offset}_{15})^{14}$ || offset || $0^2$
            condition ← (GPR [rs]$_{31}$ = 0)
            GPR [31] ← PC + 8
     T+1:   if condition then
                PC ← PC + target
            else
                NullifyCurrentInstruction
            endif

64   T:     target ← $(\text{offset}_{15})^{46}$ || offset || $0^2$
            condition ← (GPR [rs]$_{63}$ = 0)
            GPR [31] ← PC + 8
     T+1:   if condition then
                PC ← PC + target
            else
                NullifyCurrentInstruction
            endif
```

Exceptions:

None

BGEZL Branch On Greater Than Or Equal To Zero Likely BGEZL

31 26	25 21	20 16	15 0
REGIMM 0 0 0 0 0 1	rs	BGEZL 0 0 0 1 1	offset
6	5	5	16

Format:

BGEZL rs, offset

Description:

A branch target address is computed from the sum of the address of the instruction in the delay slot and the 16-bit *offset*, shifted left two bits and sign-extended. If the contents of general register *rs* are zero or greater when compared to zero, then the program branches to the target address, with a delay of one instruction. If the conditional branch is not taken, the instruction in the branch delay slot is nullified.

Operation:

```
32   T:     target ← $(\text{offset}_{15})^{14}$ || offset || $0^2$
            condition ← $(\text{GPR [rs]}_{31} = 0)$
     T+1:   if condition then
                PC ← PC + target
            else
                NullifyCurrentInstruction
            endif

64   T:     target ← $(\text{offset}_{15})^{46}$ || offset || $0^2$
            condition ← $(\text{GPR [rs]}_{63} = 0)$
     T+1:   if condition then
                PC ← PC + target
            else
                NullifyCurrentInstruction
            endif
```

Exceptions:

None

BGTZ Branch On Greater Than Zero BGTZ

31 26	25 21	20 16	15 0
BGTZ 0 0 0 1 1 1	rs	0 0 0 0 0 0	offset
6	5	5	16

Format:

BGTZ rs, offset

Description:

A branch target address is computed from the sum of the address of the instruction in the delay slot and the 16-bit *offset*, shifted left two bits and sign-extended. If the contents of general register *rs* are zero or greater when compared to zero, then the program branches to the target address, with a delay of one instruction.

Operation:

```
32   T:     target ← $(\text{offset}_{15})^{14}$ || offset || $0^2$
            condition ← $(\text{GPR [rs]}_{31} = 0)$ and $(\text{GPR [rs]} \neq 0^{32})$
     T+1:   if condition then
                PC ← PC + target
            endif

64   T:     target ← $(\text{offset}_{15})^{46}$ || offset || $0^2$
            condition ← $(\text{GPR [rs]}_{63} = 0)$ and $(\text{GPR [rs]} \neq 0^{64})$
     T+1:   if condition then
                PC ← PC + target
            endif
```

Exceptions:

None

BGTZL Branch On Greater Than Zero Likely BGTZL

31 26	25 21	20 16	15 0
BGTZL 0 1 0 1 1 1	rs	0 0 0 0 0 0	offset
6	5	5	16

Format:

BGTZL rs, offset

Description:

A branch target address is computed from the sum of the address of the instruction in the delay slot and the 16-bit *offset*, shifted left two bits and sign-extended. The contents of general register *rs* are compared to zero. If the contents of general register *rs* are greater than zero, then the program branches to the target address, with a delay of one instruction. If the conditional branch is not taken, the instruction in the branch delay slot is nullified.

Operation:

```
32   T:     target ← $(\text{offset}_{15})^{14}$ || offset || $0^2$
            condition ← $(\text{GPR [rs]}_{31} = 0)$ and $(\text{GPR [rs]} \neq 0^{32})$
     T+1:   if condition then
                PC ← PC + target
            else
                NullifyCurrentInstruction
            endif

64   T:     target ← $(\text{offset}_{15})^{46}$ || offset || $0^2$
            condition ← $(\text{GPR [rs]}_{63} = 0)$ and $(\text{GPR [rs]} \neq 0^{64})$
     T+1:   if condition then
                PC ← PC + target
            else
                NullifyCurrentInstruction
            endif
```

Exceptions:

None

BLEZ Branch On Less Than Or Equal To Zero BLEZ

31 26	25 21	20 16	15 0
BLEZ 0 0 0 1 1 0	rs	0 0 0 0 0 0	offset
6	5	5	16

Format:

BLEZ rs, offset

Description:

A branch target address is computed from the sum of the address of the instruction in the delay slot and the 16-bit *offset*, shifted left two bits and sign-extended. The contents of general register *rs* are compared to zero. If the contents of general register *rs* are zero or smaller than zero, then the program branches to the target address, with a delay of one instruction.

Operation:

```
32   T:     target ← $(\text{offset}_{15})^{14}$ || offset || $0^2$
            condition ← (GPR [rs]$_{31}$ = 1) or (GPR [rs] = $0^{32}$)
     T+1:   if condition then
                PC ← PC + target
            endif

64   T:     target ← $(\text{offset}_{15})^{46}$ || offset || $0^2$
            condition ← (GPR [rs]$_{63}$ = 1) or (GPR [rs] = $0^{64}$)
     T+1:   if condition then
                PC ← PC + target
            endif
```

Exceptions:

None

BLEZL Branch On Less Than Or Equal To Zero Likely BLEZL

31 26	25 21	20 16	15 0
BLEZL 0 1 0 1 1 0	rs	0 0 0 0 0 0	offset
6	5	5	16

Format:

BLEZL rs, offset

Description:

A branch target address is computed from the sum of the address of the instruction in the delay slot and the 16-bit *offset*, shifted left two bits and sign-extended. The contents of general register *rs* is compared to zero. If the contents of general register *rs* are zero or smaller than zero, then the program branches to the target address, with a delay of one instruction.

If the conditional branch is not taken, the instruction in the branch delay slot is nullified.

Operation:

```
32   T:     target ← $(\text{offset}_{15})^{14}$ || offset || $0^2$
            condition ← $(\text{GPR [rs]}_{31} = 1)$ or $(\text{GPR [rs]} = 0^{32})$
     T+1:   if condition then
                PC ← PC + target
            else
                NullifyCurrentInstruction
            endif

64   T:     target ← $(\text{offset}_{15})^{46}$ || offset || $0^2$
            condition ← $(\text{GPR [rs]}_{63} = 1)$ or $(\text{GPR [rs]} = 0^{64})$
     T+1:   if condition then
                PC ← PC + target
            else
                NullifyCurrentInstruction
            endif
```

Exceptions:

None

BLTZ Branch On Less Than Zero BLTZ

31 26	25 21	20 16	15 0
REGIMM 000001	rs	BLTZ 00000	offset
6	5	5	16

Format:

BLTZ rs, offset

Description:

A branch target address is computed from the sum of the address of the instruction in the delay slot and the 16-bit *offset*, shifted left two bits and sign-extended. If the contents of general register *rs* are smaller than zero, then the program branches to the target address, with a delay of one instruction.

Operation:

```
32   T:     target ← $(\text{offset}_{15})^{14}$ || offset || $0^2$
            condition ← $(\text{GPR [rs]}_{31} = 1)$
     T+1:   if condition then
                PC ← PC + target
            endif

64   T:     target ← $(\text{offset}_{15})^{46}$ || offset || $0^2$
            condition ← $(\text{GPR [rs]}_{63} = 1)$
     T+1:   if condition then
                PC ← PC + target
            endif
```

Exceptions:

None

BLTZAL Branch On Less Than Zero And Link BLTZAL

31 26	25 21	20 16	15 0
REGIMM 000001	rs	BLTZAL 10000	offset
6	5	5	16

Format:

BLTZAL rs, offset

Description:

A branch target address is computed from the sum of the address of the instruction in the delay slot and the 16-bit *offset,* shifted left two bits and sign-extended. Unconditionally, the address of the instruction after the delay slot is placed in the link register, *r31*. If the contents of general register *rs* are smaller than zero when compared to zero, then the program branches to the target address, with a delay of one instruction.

General register *r31* should not be specified as general register *rs*. If register *r31* is specified, restarting may be impossible due to the destruction of *rs* contents caused by storing a link address. Even such instructions are executed, an exception does not result.

Operation:

```
32   T:     target ← $(\text{offset}_{15})^{14}$ || offset || $0^2$
            condition ← $(\text{GPR [rs]}_{31} = 1)$
            GPR [31] ← PC + 8
     T+1:   if condition then
                PC ← PC + target
            endif

64   T:     target ← $(\text{offset}_{15})^{46}$ || offset || $0^2$
            condition ← $(\text{GPR [rs]}_{63} = 1)$
            GPR [31] ← PC + 8
     T+1:   if condition then
                PC ← PC + target
            endif
```

Exceptions:

None

BLTZALL Branch On Less Than Zero And Link Likely BLTZALL

31 26	25 21	20 16	15 0
REGIMM 0 0 0 0 0 1	rs	BLTZALL 1 0 0 1 0	offset
6	5	5	16

Format:

BLTZALL rs, offset

Description:

A branch target address is computed from the sum of the address of the instruction in the delay slot and the 16-bit *offset,* shifted left two bits and sign-extended. Unconditionally, the address of the instruction after the delay slot is placed in the link register, *r31*. If the contents of general register *rs* are smaller than zero when compared to zero, then the program branches to the target address, with a delay of one instruction.

General register *r31* should not be specified as general register *rs*. If register *r31* is specified, restarting may be impossible due to the destruction of *rs* contents caused by storing a link address. Even such instructions are executed, an exception does not result.

Operation:

```
32   T:    target ← $(\text{offset}_{15})^{14}$ || offset || $0^2$
           condition ← (GPR $[\text{rs}]_{31}$ = 1)
           GPR [31] ← PC + 8
     T+1:  if condition then
                PC ← PC + target
           else
                NullifyCurrentInstruction
           endif

64   T:    target ← $(\text{offset}_{15})^{46}$ || offset || $0^2$
           condition ← (GPR $[\text{rs}]_{63}$ = 1)
           GPR [31] ← PC + 8
     T+1:  if condition then
                PC ← PC + target
           else
                NullifyCurrentInstruction
           endif
```

Exceptions:

None

BLTZL Branch On Less Than Zero Likely BLTZL

31 26	25 21	20 16	15 0
REGIMM 0 0 0 0 0 1	rs	BLTZL 0 0 0 1 0	offset
6	5	5	16

Format:

BLTZ rs, offset

Description:

A branch target address is computed from the sum of the address of the instruction in the delay slot and the 16-bit *offset*, shifted left two bits and sign-extended. If the contents of general register *rs* are smaller than zero when compared to zero, then the program branches to the target address, with a delay of one instruction. If the conditional branch is not taken, the instruction in the branch delay slot is nullified.

Operation:

```
32    T:     target ← $(\text{offset}_{15})^{14}$ || offset || $0^2$
             condition ← $(\text{GPR [rs]}_{31} = 1)$
      T+1:   if condition then
                   PC ← PC + target
             else
                   NullifyCurrentInstruction
             endif

64    T:     target ← $(\text{offset}_{15})^{46}$ || offset || $0^2$
             condition ← $(\text{GPR [rs]}_{63} = 1)$
      T+1:   if condition then
                   PC ← PC + target
             else
                   NullifyCurrentInstruction
             endif
```

Exceptions:

None

BNE Branch On Not Equal BNE

31 26	25 21	20 16	15 0
BNE 0 0 0 1 0 1	rs	rt	offset
6	5	5	16

Format:

BNE rs, rt, offset

Description:

A branch target address is computed from the sum of the address of the instruction in the delay slot and the 16-bit *offset,* shifted left two bits and sign-extended. The contents of general register *rs* and the contents of general register *rt* are compared. If the two registers are not equal, then the program branches to the target address, with a delay of one instruction.

Operation:

```
32   T:     target $\leftarrow$ (offset$_{15}$)$^{14}$ || offset || $0^2$
            condition $\leftarrow$ (GPR [rs] $\neq$ GPR [rt])
     T+1:   if condition then
                PC $\leftarrow$ PC + target
            endif

64   T:     target $\leftarrow$ (offset$_{15}$)$^{46}$ || offset || $0^2$
            condition $\leftarrow$ (GPR [rs] $\neq$ GPR [rt])
     T+1:   if condition then
                PC $\leftarrow$ PC + target
            endif
```

Exceptions:

None

BNEL Branch On Not Equal Likely BNEL

31 26	25 21	20 16	15 0
BNEL 0 1 0 1 0 1	rs	rt	offset
6	5	5	16

Format:

BNEL rs, rt, offset

Description:

A branch target address is computed from the sum of the address of the instruction in the delay slot and the 16-bit *offset,* shifted left two bits and sign-extended. The contents of general register *rs* and the contents of general register *rt* are compared. If the two registers are not equal, then the program branches to the target address, with a delay of one instruction.

If the conditional branch is not taken, the instruction in the branch delay slot is nullified.

Operation:

```
32   T:      target ← $(\text{offset}_{15})^{14}$ || offset || $0^2$
             condition ← (GPR [rs] ≠ GPR [rt])
     T+1:    if condition then
                 PC ← PC + target
             else
                 NullifyCurrentInstruction
             endif

64   T:      target ← $(\text{offset}_{15})^{46}$ || offset || $0^2$
             condition ← (GPR [rs] ≠ GPR [rt])
     T+1:    if condition then
                 PC ← PC + target
             else
                 NullifyCurrentInstruction
             endif
```

Exceptions:

None

BREAK Breakpoint BREAK

31 26	25 6	5 0
SPECIAL 0 0 0 0 0 0	code	BREAK 0 0 1 1 0 1
6	20	6

Format:

BREAK

Description:

A breakpoint trap occurs, immediately and unconditionally transferring control to the exception handler.

The code field is available for use as software parameters, but is retrieved by the exception handler only by loading the contents of the memory word containing the instruction.

Operation:

```
32, 64 T:     BreakpointException
```

Exceptions:

Breakpoint exception

CACHE Cache (1/4) CACHE

31 26	25 21	20 16	15 0
CACHE 1 0 1 1 1 1	base	op	offset
6	5	5	16

Format:

CACHE op, offset (base)

Description:

The 16-bit *offset* is sign-extended and added to the contents of general register *base* to form a virtual address. The virtual address is translated to a physical address using the TLB, and the 5-bit sub-opcode specifies a cache operation for that address.

If CP0 is not usable (User or Supervisor mode) and the CP0 enable bit in the Status register is clear, a coprocessor unusable exception is taken. The operation of this instruction on any operation/cache combination not listed below, or on a secondary cache that is not incorporated in VR4120A CPU, is undefined. The operation of this instruction on uncached addresses is also undefined.

The Index operation uses part of the virtual address to specify a cache block.

For a primary cache of $2^{CACHEBITS}$ bytes with $2^{LINEBITS}$ bytes per tag, vAddrCACHEBITS...LINEBITS specifies the block.

Index_Load_Tag also uses vAddrLINEBITS...3 to select the doubleword for reading parity. When the *CE* bit of the Status register is set, Fill Cache op uses the PErr register to store parity values into the cache.

The Hit operation accesses the specified cache as normal data references, and performs the specified operation if the cache block contains valid data with the specified physical address (a hit). If the cache block is invalid or contains a different address (a miss), no operation is performed.

CACHE Cache (2/4) CACHE

Write back from a cache goes to main memory.
The main memory address to be written is specified by the cache tag and not the physical address translated using TLB.
TLB Refill and TLB Invalid exceptions can occur on any operation. For Index operations[Note] for addresses in the unmapped areas, unmapped addresses may be used to avoid TLB exceptions. Index operations never cause a TLB Modified exception. Bits 17 and 16 of the instruction code specify the cache for which the operation is to be performed as follows.

Code	Name	Cache
0	I	Instruction cache
1	D	Data cache
2	—	Reserved
3	—	Reserved

Note Physical addresses here are used to index the cache, and they do not need to match the cache tag.

Bits 20 to 18 of this instruction specify the contents of cache operaiton. Details are provided from the next page.

CACHE Cache (3/4) CACHE

Code	Cache	Name	Operation
0	I	Index_Invalidate	Set the cache state of the cache block to Invalid.
0	D	Index_Write_ Back_Invalidate	Examine the cache state and W bit of the primary data cache block at the index specified by the virtual address. If the state is not Invalid and the W bit is set, then write back the block to memory. The address to write is taken from the primary cache tag. Set cache state of primary cache block to Invalid.
1	I, D	Index_Load_Tag	Read the tag for the cache block at the specified index and place it into the TagLo CP0 registers, ignoring parity errors. Also load the data parity bits into the ECC register.
2	I, D	Index_Store_ Tag	Write the tag for the cache block at the specified index from the TagLo and TagHi CP0 registers.
3	D	Create_Dirty_ Exclusive	This operation is used to avoid loading data needlessly from memory when writing new contents into an entire cache block. If the cache block does not contain the specified address, and the block is dirty, write it back to the memory. In all cases, set the cache state to Dirty.
4	I, D	Hit_Invalidate	If the cache block contains the specified address, mark the cache block invalid.
5	D	Hit_Write_Back Invalidate	If the cache block contains the specified address, write back the data if it is dirty, and mark the cache block invalid.
5	I	Fill	Fill the primary instruction cache block from memory. If the CE bit of the Status register is set, the contents of the ECC register is used instead of the computed parity bits for addressed doubleword when written to the instruction cache.
6	D	Hit_Write_Back	If the cache block contains the specified address, and the W bit is set, write back the data to memory and clear the W bit.
6	I	Hit_Write_Back	If the cache block contains the specified address, write back the data unconditionally.

CACHE Cache (4/4) CACHE

Operation:

```
32, 64 T:  $vAddr \leftarrow ((offset_{15})^{48} \,||\, offset_{15...0}) + GPR\ [base]$
           $(pAddr, uncached) \leftarrow AddressTranslation\ (vAddr, DATA)$
           CacheOp (op, vAddr, pAddr)
```

Exceptions:

Coprocessor unusable exception
TLB Refill exception
TLB Invalid exception
Bus Error exception
Address Error exception
Cache Error exception

DADD Doubleword Add DADD

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	rd	0 0 0 0 0 0	DADD 1 0 1 1 0 0
6	5	5	5	5	6

Format:

DADD rd, rs, rt

Description:

The contents of general register *rs* and the contents of general register *rt* are added to form the result. The result is placed into general register *rd*.

An overflow exception occurs if the carries out of bits 62 and 63 differ (2's complement overflow). The destination register *rd* is not modified when an integer overflow exception occurs.

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

Operation:

```
64    T:    GPR [rd] ← GPR [rs] + GPR [rt]
```

Exceptions:

Integer overflow exception

Reserved instruction exception (32-bit user mode/supervisor mode)

DADDI Doubleword Add Immediate DADDI

31 26	25 21	20 16	15 0
DADDI 0 1 1 0 0 0	rs	rt	immediate
6	5	5	16

Format:

DADDI rt, rs, immediate

Description:

The 16-bit *immediate* is sign-extended and added to the contents of general register *rs* to form the result. The result is placed into general register *rt.*

An overflow exception occurs if carries out of bits 62 and 63 differ (2's complement overflow). The destination register *rt* is not modified when an integer overflow exception occurs.

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

Operation:

```
64    T:    $GPR [rt] \leftarrow GPR [rs] + (immediate_{15})^{48} \| immediate_{15...0}$
```

Exceptions:

Integer overflow exception

Reserved instruction exception (32-bit user mode/supervisor mode)

DADDIU Doubleword Add Immediate Unsigned DADDIU

31 26	25 21	20 16	15 0
DADDIU 0 1 1 0 0 1	rs	rt	immediate
6	5	5	16

Format:

DADDIU rt, rs, immediate

Description:

The 16-bit *immediate* is sign-extended and added to the contents of general register *rs* to form the result. The result is placed into general register *rt.* No integer overflow exception occurs under any circumstances.

The only difference between this instruction and the DADDI instruction is that DADDIU never causes an overflow exception.

Operation:

64	T:	GPR [rt] ← GPR [rs] + $(\text{immediate}_{15})^{48}$ || $\text{immediate}_{15...0}$

Exceptions:

Reserved instruction exception (32-bit user mode/supervisor mode)

DADDU Doubleword Add Unsigned DADDU

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	rd	0 0 0 0 0 0	DADDU 1 0 1 1 0 1
6	5	5	5	5	6

Format:

DADDU rd, rs, rt

Description:

The contents of general register *rs* and the contents of general register *rt* are added to form the result. The result is placed into general register *rd*.

No overflow exception occurs under any circumstances.

The only difference between this instruction and the DADD instruction is that DADDU never causes an overflow exception.

Operation:

```
64    T:    GPR [rd] ← GPR [rs] + GPR [rt]
```

Exceptions:

Reserved instruction exception (32-bit user mode/supervisor mode)

DDIV Doubleword Divide DDIV

31 26	25 21	20 16	15 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	0 0 0 0 0 0 0 0 0 0 0	DDIV 0 1 1 1 1 0
6	5	5	10	6

Format:

DDIV rs, rt

Description:

The contents of general register *rs* are divided by the contents of general register *rt,* treating both operands as 2's complement values. No overflow exception occurs under any circumstances, and the result of this operation is undefined when the divisor is zero.

This instruction is typically followed by additional instructions to check for a zero divisor and for overflow.

When the operation completes, the quotient word of the double result is loaded into special register *LO*, and the remainder word of the double result is loaded into special register *HI*.

If either of the two preceding instructions is MFHI or MFLO, the results of those instructions are undefined. Correct operation requires separating reads of *HI* or *LO* from writes by two or more instructions.

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

Operation:

```
64   T-2:  LO   ← undefined
           HI   ← undefined
     T-1:  LO   ← undefined
           HI   ← undefined
     T:    LO   ← GPR [rs] div GPR [rt]
           HI   ← GPR [rs] mod GPR [rt]
```

Exceptions:

Reserved instruction exception (32-bit user mode/supervisor mode)

DDIVU Doubleword Divide Unsigned DDIVU

31 26	25 21	20 16	15 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	0 0 0 0 0 0 0 0 0 0 0	DDIVU 0 1 1 1 1 1
6	5	5	10	6

Format:

DDIVU rs, rt

Description:

The contents of general register *rs* are divided by the contents of general register *rt,* treating both operands as unsigned values. No integer overflow exception occurs under any circumstances, and the result of this operation is undefined when the divisor is zero.

This instruction may be followed by additional instructions to check for a zero divisor, inserted by the programmer.

When the operation completes, the quotient word of the double result is loaded into special register *LO*, and the remainder word of the double result is loaded into special register *HI*.

If either of the two preceding instructions is MFHI or MFLO, the results of those instructions are undefined. Correct operation requires separating reads of *HI* or *LO* from writes by two or more instructions.

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

Operation:

```
64   T-2:  LO   ← undefined
           HI   ← undefined
     T-1:  LO   ← undefined
           HI   ← undefined
     T:    LO   ←  (0 || GPR [rs]) div (0 || GPR [rt])
           HI   ← (0 || GPR [rs]) mod (0 || GPR [rt])
```

Exceptions:

Reserved instruction exception (32-bit user mode/supervisor mode)

DIV Divide DIV

31 26	25 21	20 16	15 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	0 0 0 0 0 0 0 0 0 0 0	DIV 0 1 1 0 1 0
6	5	5	10	6

Format:

DIV rs, rt

Description:

The contents of general register *rs* are divided by the contents of general register *rt,* treating both operands as 2's complement values. No overflow exception occurs under any circumstances, and the result of this operation is undefined when the divisor is zero.

In 64-bit mode, the operands must be valid sign-extended, 32-bit values.

This instruction is typically followed by additional instructions to check for a zero divisor and for overflow.

When the operation completes, the quotient word of the double result is loaded into special register *LO*, and the remainder word of the double result is loaded into special register *HI*.

If either of the two preceding instructions is MFHI or MFLO, the results of those instructions are undefined. Correct operation requires separating reads of *HI* or *LO* from writes by two or more instructions.

Operation:

```
32   T-2:  LO   ← undefined
           HI   ← undefined
     T-1:  LO   ← undefined
           HI   ← undefined
     T:    LO   ← GPR [rs] div GPR [rt]
           HI   ← GPR [rs] mod GPR [rt]

64   T-2:  LO   ← undefined
           HI   ← undefined
     T-1:  LO   ← undefined
           HI   ← undefined
     T:    q    ← GPR [rs]$_{31..0}$ div GPR [rt]$_{31..0}$
           r    ← GPR [rs]$_{31..0}$ mod GPR [rt]$_{31..0}$
           LO   ← $(q_{31})^{32}$ || $q_{31..0}$
           HI   ← $(r_{31})^{32}$|| $r_{31..0}$
```

Exceptions:

None

DIVU Divide Unsigned DIVU

31 26	25 21	20 16	15 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	0 0 0 0 0 0 0 0 0 0 0	DIVU 0 1 1 0 1 1
6	5	5	10	6

Format:

DIVU rs, rt

Description:

The contents of general register *rs* are divided by the contents of general register *rt,* treating both operands as unsigned values. No integer overflow exception occurs under any circumstances, and the result of this operation is undefined when the divisor is zero.

In 64-bit mode, the operands must be valid sign-extended, 32-bit values.

This instruction is typically followed by additional instructions to check for a zero divisor.

When the operation completes, the quotient word of the double result is loaded into special register *LO*, and the remainder word of the double result is loaded into special register *HI*.

If either of the two preceding instructions is MFHI or MFLO, the results of those instructions are undefined. Correct operation requires separating reads of *HI* or *LO* from writes by two or more instructions.

Operation:

```
32   T-2:  LO   ← undefined
           HI   ← undefined
     T-1:  LO   ← undefined
           HI   ← undefined
     T:    LO   ← (0 || GPR [rs]) div (0 || GPR [rt])
           HI   ←  0 || GPR [rs]) mod (0 || GPR [rt])

64   T-2:  LO   ← undefined
           HI   ← undefined
     T-1:  LO   ← undefined
           HI   ← undefined
     T:    q    ← (0 || GPR $[rs]_{31..0}$) div (0 || GPR $[rt]_{31..0}$)
           r    ← (0 || GPR $[rs]_{31..0}$) mod (0 || GPR $[rt]_{31..0}$)
           LO   ← $(q_{31})^{32}$ || $q_{31..0}$
           HI   ← $(r_{31})^{32}$ || $r_{31..0}$
```

Exceptions:

None

DMACC Doubleword Multiply and Accumulate (1/3) DMACC

31 26	25 21	20 16	15 11	10	9 7	6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	rd	sat	0 0 0	us	DMACC 1 0 1 0 0 1
6	5	5	5	1	3	1	6

Format:

DMACC rd, rs, rt
DMACCU rd, rs, rt
DMACCS rd, rs, rt
DMACCUS rd, rs, rt

Description:

DMACC instruction differs mnemonics by each setting of op codes sat, hi and us as follows.

Mnemonic	sat	us
DMACC	0	0
DMACCU	0	1
DMACCS	1	0
DMACCUS	1	1

The number of significant bits in the operands of the DMACC instruction differ depending on whether saturation processing is executed (sat = 1) or not executed (sat = 0).

- When saturation processing is executed (sat = 1): DMACCS, DMACCUS instructions
  The contents of general register *rs* is multiplied by the contents of general register *rt*. If both operands are set as "us = 1" (DMACCUS instruction), the contents are handled as 16 bit unsigned data. If they are set as "us = 0" (DMACCS instruction), the contents are handled as 16 bit signed integers. Sign/zero expansion by software is required for any bits exceeding 16 bits in the operands.
  The product of this multiply operation is added to the value in the LO special register. If us = 1, this add operation handles the values being added as 32 bit unsigned data. If us = 0, the values are handled as 32 bit signed integers. Sign/zero expansion by software is required for any bits exceeding 32 bits in the LO special register.
  After saturation processing to 32 bits has been performed (see the table below), the sum from this add operation is loaded to the LO special register. When hi = 1, data that is the same as the data loaded to the HI special register is also loaded to the rd general register. When hi = 0, data that is the same as the data loaded to the LO special register is also loaded to the rd general register. Overflow exceptions do not occur.

DMACC Doubleword Multiply and Accumulate (2/3) DMACC

- When saturation processing is not executed (sat = 0): DMACC, DMACCU instructions
  The contents of general register *rs* is multiplied by the contents of general register *rt*. If both operands are set as "us = 1" (DMACCU instruction), the contents are handled as 32 bit unsigned data. If they are set as "us = 0" (DMACC instruction), the contents are handled as 32 bit signed integers. Sign/zero expansion by software is required for any bits exceeding 32 bits in the operands.
  The product of this multiply operation is added to the value in the LO special register. If us = 1, this add operation handles the values being added as 64 bit unsigned data. If us = 0, the values are handled as 64 bit signed integers.
  The sum from this add operation is loaded to the LO special register. When hi = 1, data that is the same as the data loaded to the HI special register is also loaded to the rd general register. When hi = 0, data that is the same as the data loaded to the LO special register is also loaded to the rd general register. Overflow exceptions do not occur.

These operations are defined for 64 bit mode and 32 bit kernel mode. A reserved instruction exception occurs if one of these instructions is executed during 32 bit user/supervisor mode.
The correspondence of us and sat settings and values stored during saturation processing is shown below, along with the hazard cycles required between execution of the instruction for manipulating the HI and LO registers and execution of the DMACC instruction.

Values Stored during Saturation Processing

us	sat	Overflow	Underflow
0	0	Store calculation result as is	Store calculation result as is
1	0	Store calculation result as is	Store calculation result as is
0	1	0000 0000 7FFF FFFFH	FFFF FFFF 8000 0000H
1	1	FFFF FFFF FFFF FFFFH	None

Hazard Cycle Counts

Instruction	Cycle count
MULT, MULTU	1
DMULT, DMULTU	3
DIV, DIVU	36
DDIV, DDIVU	68
MFHI, MFLO	2
MTHI, MTLO	0
MACC	0
DMACC	0

DMACC Doubleword Multiply and Accumulate (3/3) DMACC

Operation:

```
64, sat=0, us=0 (DMACC instruction)
    T:    $temp1 \leftarrow ((GPR[rs]_{31})^{32} || GPR [rs]) * ((GPR[rt]_{31})^{32} || GPR [rt])$
          $temp2 \leftarrow temp1 + LO$
          $LO \leftarrow temp2$
          $GPR[rd] \leftarrow LO$

64, sat=0, us=1 (DMACCU instruction)
    T:    $temp1 \leftarrow (0^{32} || GPR [rs]) * (0^{32} || GPR [rt])$
          $temp2 \leftarrow temp1 + LO$
          $LO \leftarrow temp2$
          $GPR[rd] \leftarrow LO$

64, sat=1, us=0 (DMACCS instruction)
    T:    $temp1 \leftarrow ((GPR[rs]_{31})^{32} || GPR [rs]) * ((GPR[rt]_{31})^{32} || GPR [rt])$
          $temp2 \leftarrow saturation(temp1 + LO)$
          $LO \leftarrow temp2$
          $GPR[rd] \leftarrow LO$

64, sat=1, us=1 (DMACCUS instruction)
    T:    $temp1 \leftarrow (0^{32} || GPR [rs]) * (0^{32} || GPR [rt])$
          $temp2 \leftarrow saturation(temp1 + LO)$
          $LO \leftarrow temp2$
          $GPR[rd] \leftarrow LO$
```

Exceptions:

Reserved instruction exception (32-bit user mode/supervisor mode)

DMFC0 Doubleword Move From System Control Coprocessor DMFC0

31 26	25 21	20 16	15 11	10 0
COP0 0 1 0 0 0 0	DMF 0 0 0 0 1	rt	rd	0 0 0 0 0 0 0 0 0 0 0 0
6	5	5	5	11

Format:

DMFC0 rt, rd

Description:

The contents of coprocessor register *rd* of the CP0 are loaded into general register *rt.*

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception. All 64-bits of the general register destination are written from the coprocessor register source. The operation of DMFC0 on a 32-bit coprocessor 0 register is undefined.

Operation:

```
64   T:     data ← CPR [0, rd]
     T+1:   GPR [rt] ← data
```

Exceptions:

Coprocessor unusable exception (user mode and supervisor mode if CP0 not enabled)

Reserved instruction exception (32-bit user mode/supervisor mode)

DMTC0 Doubleword Move To System Control Coprocessor DMTC0

31 26	25 21	20 16	15 11	10 0
COP0 0 1 0 0 0 0	DMT 0 0 1 0 1	rt	rd	0 0 0 0 0 0 0 0 0 0 0 0
6	5	5	5	11

Format:

DMTC0 rt, rd

Description:

The contents of general register *rt* are loaded into coprocessor register *rd* of the CP0.

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

All 64-bits of the coprocessor 0 register are written from the general register source. The operation of DMTC0 on a 32-bit coprocessor 0 register is undefined.

Because the state of the virtual address translation system may be altered by this instruction, the operation of load instructions, store instructions, and TLB operations immediately prior to and after this instruction are undefined.

Operation:

```
64   T:      data ← GPR [rt]
     T+1:    CPR [0, rd] ← data
```

Exceptions:

Coprocessor unusable exception (In 64-bit/32-bit user and supervisor mode if CP0 not enabled)

Reserved instruction exception (32-bit user mode/supervisor mode)

DMULT Doubleword Multiply DMULT

31 26	25 21	20 16	15 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	0 0 0 0 0 0 0 0 0 0 0	DMULT 0 1 1 1 0 0
6	5	5	10	6

Format:

DMULT rs, rt

Description:

The contents of general registers *rs* and *rt* are multiplied, treating both operands as 2's complement values. No integer overflow exception occurs under any circumstances.

When the operation completes, the low-order word of the double result is loaded into special register *LO*, and the high-order word of the double result is loaded into special register *HI*.

If either of the two preceding instructions is MFHI or MFLO, the results of these instructions are undefined. Correct operation requires separating reads of *HI* or *LO* from writes by a minimum of two other instructions.

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

Operation:

```
64   T-2:  LO   ← undefined
           HI   ← undefined
     T-1:  LO   ← undefined
           HI   ← undefined
     T:    t    ← GPR [rs] * GPR [rt]
           LO   ← $t_{63..0}$
           HI   ← $t_{127..64}$
```

Exceptions:

Reserved instruction exception (32-bit user mode/supervisor mode)

DMULTU Doubleword Multiply Unsigned DMULTU

31 26	25 21	20 16	15 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	0 0 0 0 0 0 0 0 0 0 0	DMULTU 0 1 1 1 0 1
6	5	5	10	6

Format:

DMULTU rs, rt

Description:

The contents of general register *rs* and the contents of general register *rt* are multiplied, treating both operands as unsigned values. No overflow exception occurs under any circumstances.

When the operation completes, the low-order word of the double result is loaded into special register *LO*, and the high-order word of the double result is loaded into special register *HI*.

If either of the two preceding instructions is MFHI or MFLO, the results of these instructions are undefined. Correct operation requires separating reads of *HI* or *LO* from writes by a minimum of two instructions.

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

Operation:

```
64   T-2:  LO   ← undefined
           HI   ← undefined
     T-1:  LO   ← undefined
           HI   ← undefined
     T:    t    ← (0 || GPR [rs]) * (0 || GPR [rt])
           LO   ← $t_{63..0}$
           HI   ← $t_{127..64}$
```

Exceptions:

Reserved instruction exception (32-bit user mode/supervisor mode)

DSLL Doubleword Shift Left Logical DSLL

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	0 0 0 0 0 0	rt	rd	sa	DSLL 1 1 1 0 0 0
6	5	5	5	5	6

Format:

DSLL rd, rt, sa

Description:

The contents of general register *rt* are shifted left by *sa* bits, inserting zeros into the low-order bits. The result is placed in register *rd.*

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

Operation:

```
64    T:    $s \leftarrow 0 \| sa$
            $GPR [rd] \leftarrow GPR [rt]_{(63 - s)..0} \| 0^{s}$
```

Exceptions:

Reserved instruction exception (32-bit user mode/supervisor mode)

DSLLV Doubleword Shift Left Logical Variable DSLLV

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	rd	0 0 0 0 0 0	DSLLV 0 1 0 1 0 0
6	5	5	5	5	6

Format:

DSLLV rd, rt, rs

Description:

The contents of general register *rt* are shifted left by the number of bits specified by the low-order six bits contained in general register *rs*, inserting zeros into the low-order bits. The result is placed in register *rd*.

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

Operation:

```
64   T:   $s \leftarrow GPR[rs]_{5..0}$
          $GPR[rd] \leftarrow GPR[rt]_{(63 - s)..0} \,||\, 0^{s}$
```

Exceptions:

Reserved instruction exception (32-bit user mode/supervisor mode)

DSLL32 Doubleword Shift Left Logical + 32 DSLL32

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	0 0 0 0 0 0	rt	rd	sa	DSLL32 1 1 1 1 0 0
6	5	5	5	5	6

Format:

DSLL32 rd, rt, sa

Description:

The contents of general register *rt* are shifted left by *32* + *sa* bits, inserting zeros into the low-order bits. The result is placed in register *rd*.

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

Operation:

```
64    T:    $s \leftarrow 1 \| sa$
            $GPR [rd] \leftarrow GPR [rt]_{(63 - s)..0} \| 0^s$
```

Exceptions:

Reserved instruction exception (32-bit user mode/supervisor mode)

DSRA Doubleword Shift Right Arithmetic DSRA

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	0 0 0 0 0 0	rt	rd	sa	DSRA 1 1 1 0 1 1
6	5	5	5	5	6

Format:

DSRA rd, rt, sa

Description:

The contents of general register *rt* are shifted right by *sa* bits, sign-extending the high-order bits. The result is placed in register *rd*.

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

Operation:

```
64   T:   $s \leftarrow 0 \,||\, sa$
          $GPR\ [rd] \leftarrow (GPR\ [rt]_{63})^{s} \,||\, GPR\ [rt]_{63..s}$
```

Exceptions:

Reserved instruction exception (32-bit user mode/supervisor mode)

DSRAV Doubleword Shift Right Arithmetic Variable DSRAV

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 000000	rs	rt	rd	0 00000	DSRAV 010111
6	5	5	5	5	6

Format:

DSRAV rd, rt, rs

Description:

The contents of general register *rt* are shifted right by the number of bits specified by the low-order six bits of general register *rs*, sign-extending the high-order bits. The result is placed in register *rd*.

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

Operation:

```
64   T:   $s \leftarrow GPR\ [rs]_{5..0}$
          $GPR\ [rd] \leftarrow (GPR\ [rt]_{63})^{s} \ ||\ GPR\ [rt]_{63..s}$
```

Exceptions:

Reserved instruction exception (32-bit user mode/supervisor mode)

DSRA32 Doubleword Shift Right Arithmetic + 32 DSRA32

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	0 0 0 0 0 0	rt	rd	sa	DSRA32 1 1 1 1 1 1
6	5	5	5	5	6

Format:

DSRA32 rd, rt, sa

Description:

The contents of general register *rt* are shifted right by *32* + *sa* bits, sign-extending the high-order bits. The result is placed in register *rd*.

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

Operation:

```
64    T:    $s \leftarrow 1 \parallel sa$
            $GPR\ [rd] \leftarrow (GPR\ [rt]_{63})^{s} \parallel GPR\ [rt]_{63..s}$
```

Exceptions:

Reserved instruction exception (32-bit user mode/supervisor mode)

DSRL Doubleword Shift Right Logical DSRL

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	0 0 0 0 0 0	rt	rd	sa	DSRL 1 1 1 0 1 0
6	5	5	5	5	6

Format:

DSRL rd, rt, sa

Description:

The contents of general register *rt* are shifted right by *sa* bits, inserting zeros into the high-order bits. The result is placed in register *rd*.

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

Operation:

```
64   T:   $s \leftarrow 0 \,||\, sa$
          $GPR\ [rd] \leftarrow 0^{s} \,||\, GPR\ [rt]_{63..s}$
```

Exceptions:

Reserved instruction exception (32-bit user mode/supervisor mode)

DSRLV Doubleword Shift Right Logical Variable DSRLV

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	rd	0 0 0 0 0 0	DSRLV 0 1 0 1 1 0
6	5	5	5	5	6

Format:

DSRLV rd, rt, rs

Description:

The contents of general register *rt* are shifted right by the number of bits specified by the low-order six bits of general register *rs,* inserting zeros into the high-order bits. The result is placed in register *rd*.

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

Operation:

```
64    T:    $s \leftarrow \text{GPR [rs]}_{5..0}$
            $\text{GPR [rd]} \leftarrow 0^{s} \,||\, \text{GPR [rt]}_{63..s}$
```

Exceptions:

Reserved instruction exception (32-bit user mode/supervisor mode)

DSRL32 Doubleword Shift Right Logical + 32 DSRL32

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	0 0 0 0 0 0	rt	rd	sa	DSRL32 1 1 1 1 1 0
6	5	5	5	5	6

Format:

DSRL32 rd, rt, sa

Description:

The contents of general register *rt* are shifted right by *32* + *sa* bits, inserting zeros into the high-order bits. The result is placed in register *rd*.

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

Operation:

```
64   T:   $s \leftarrow 1 \,||\, sa$
          $GPR\ [rd] \leftarrow 0^{s} \,||\, GPR\ [rt]_{63..s}$
```

Exceptions:

Reserved instruction exception (32-bit user mode/supervisor mode)

DSUB Doubleword Subtract DSUB

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 000000	rs	rt	rd	0 00000	DSUB 101110
6	5	5	5	5	6

Format:

DSUB rd, rs, rt

Description:

The contents of general register *rt* are subtracted from the contents of general register *rs* to form a result. The result is placed into general register *rd.*

An integer overflow exception takes place if the carries out of bits 62 and 63 differ (2's complement overflow). The destination register *rd* is not modified when an integer overflow exception occurs.

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

Operation:

```
64    T:    GPR [rd] ← GPR [rs] – GPR [rt]
```

Exceptions:

Integer overflow exception

Reserved instruction exception (32-bit user mode/supervisor mode)

DSUBU Doubleword Subtract Unsigned DSUBU

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	rd	0 0 0 0 0 0	DSUBU 1 0 1 1 1 1
6	5	5	5	5	6

Format:

DSUBU rd, rs, rt

Description:

The contents of general register *rt* are subtracted from the contents of general register *rs* to form a result. The result is placed into general register *rd*.

The only difference between this instruction and the DSUB instruction is that DSUBU never traps on overflow. No integer overflow exception occurs under any circumstances.

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

Operation:

```
64    T:    GPR [rd] ← GPR [rs] – GPR [rt]
```

Exceptions:

Reserved instruction exception (32-bit user mode/supervisor mode)

ERET Exception Return ERET

31 26	25	24 6	5 0
COP0 0 1 0 0 0 0	CO 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	ERET 0 1 1 0 0 0
6	1	19	6

Format:

ERET

Description:

ERET is the instruction for returning from an interrupt, exception, or error trap. Unlike a branch or jump instruction, ERET does not execute the next instruction.

ERET must not itself be placed in a branch delay slot.

If the processor is servicing an error trap (*SR2* = 1), then load the PC from the ErrorEPC register and clear the *ERL* bit of the Status register (*SR2*). Otherwise (*SR2* = 0), load the PC from the EPC register, and clear the *EXL* bit of the Status register (*SR1* = 0).

When a MIPS16 instruction can be executed, the value of clearing the least significant bit of the EPC or error EPC register to 0 is loaded to PC. This means the content of the least significant bit is reflected on the ISA mode bit (internal).

Operation:

```
32, 64  T:  if $SR_2$ = 1 then
               if MIPS16EN = 1 then
                  PC ← $ErrorEPC_{63..1}$ || 0
                  ISA MODE ← $ErrorEPC_0$
               else
                  PC ← ErrorEPC
               endif
                  SR ← $SR_{31..3}$ || 0 || $SR_{1..0}$
            else
                  if MIPS16EN = 1 then
                     PC ← $EPC_{63..1}$ || 0
                     ISA MODE ← $EPC_0$
                  else
                     PC ← EPC
                  endif
                     SR ← $SR_{31..2}$ || 0 || $SR_0$
            endif
```

Exceptions:

Coprocessor unusable exception

HIBERNATE Hibernate HIBERNATE

31 26	25	24 6	5 0
COP0 0 1 0 0 0 0	CO 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	HIBERNATE 1 0 0 0 1 1
6	1	19	6

Format:

HIBERNATE

Description:

HIBERNATE instruction starts mode transition from Fullspeed mode to Hibernate mode.

When the HIBERNATE instruction finishes the WB stage, the processor wait by the SysAD bus is idle state, after then the internal clocks and the system interface clocks will shut down, thus freezing the pipeline.

Cold Reset causes the Hibernate mode to the Fullspeed mode transition.

Operation:

```
32, 64  T:
        T+1:  Hibernate operation ()
```

Exceptions:

Coprocessor unusable exception

J Jump J

31 26	25 0
J 0 0 0 0 1 0	target
6	26

Format:

J target

Description:

The 26-bit target address is shifted left two bits and combined with the high-order four bits of the address of the delay slot. The program unconditionally jumps to this calculated address with a delay of one instruction.

Operation:

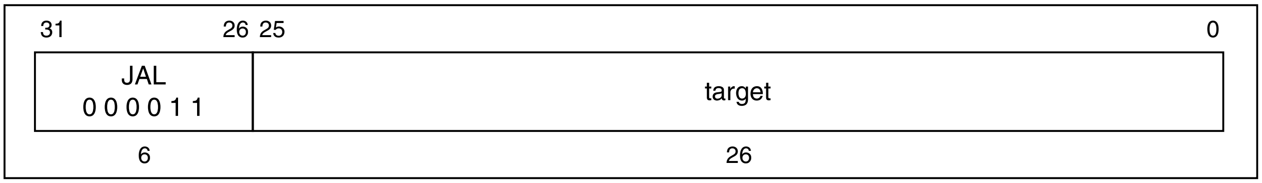
```
32   T:     temp ← target
     T+1:   PC ← $PC_{31..28}$ || temp || $0^2$

64   T:     temp ← target
     T+1:   PC ← $PC_{63..28}$ || temp || $0^2$
```

Exceptions:

None

JAL Jump And Link JAL



Format:

JAL target

Description:

The 26-bit target address is shifted left two bits and combined with the high-order four bits of the address of the delay slot. The program unconditionally jumps to this calculated address with a delay of one instruction. The address of the instruction after the delay slot is placed in the link register, *r31*. The address of the instruction immediately after a delay slot is placed in the link register (r31). When a MIPS16 instruction can be executed, the value of bit 0 of r31 indicates the ISA mode bit before jump.

Operation:

```
32   T:     temp ← target
            If MIPS16En = 1 then
                GPR[31] ← $(PC+8)_{31..1}$ || ISA MODE
            else
                GPR[31] ← PC+8
            endif
     T+1:   PC ← $PC_{31..28}$ || temp || $0^2$

64   T:     temp ← target
            If MIPS16EN = 1 then
                GPR[31] ← $(PC+8)_{63..1}$ || ISA MODE
            else
                GPR[31] ← PC+8
            endif
     T+1:   PC ← $PC_{63..28}$ || temp || $0^2$
```

Exceptions:

None

JALR Jump And Link Register JALR

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	rs	0 0 0 0 0 0	rd	0 0 0 0 0 0	JALR 0 0 1 0 0 1
6	5	5	5	5	6

Format:

JALR rs
JALR rd, rs

Description:

The program unconditionally jumps to the address contained in general register *rs*, with a delay of one instruction. When a MIPS16 instruction can be executed, the program unconditionally jumps with a delay of one instruction to the address indicated by the value of clearing the least significant bit of the general-purpose register *rs* to 0. Then, the content of the least significant bit of the general-purpose register *rs* is set to the ISA mode bit (internal). The address of the instruction after the delay slot is placed in general register *rd*. The default value of *rd*, if omitted in the assembly language instruction, is 31. When a MIPS16 instruction can be executed, the value of bit 0 of *rd* indicates the ISA mode bit before jump. Register specifiers *rs* and *rd* may not be equal, because such an instruction does not have the same effect when re-executed. Because storing a link address destroys the contents of *rs* if they are equal. However, an attempt to execute this instruction is *not* trapped, and the result of executing such an instruction is undefined.

Since 32-bit length instructions must be word-aligned, a **JALR** instruction must specify a target register (*rs*) that contains an address whose two low-order bits are zero when a MIPS16 instruction can be executed. If these low-order bits are not zero, an address error exception will occur when the jump target instruction is subsequently fetched.

Operation:

```
32, 64  T:     temp ← GPR [rs]
               If MIPS16EN = 1 then
                  GPR [rd] ← $(PC + 8)_{63..1}$ || ISA MODE
               else
                  GPR [rd] ← PC + 8
               endif
        T+1:   If MIPS16EN = 1 then
                  PC ← $temp_{63..1}$ || 0
                  ISA MODE ← $temp_0$
               else
                  PC ← temp
               endif
```

Exceptions:

None

JALX Jump And Link Exchange JALX

31 26	25 0
JALX 011101	target
6	26

Format:

JALX target

Description:

When a MIPS16 instruction can be executed, a 26-bit target is shifted to left by 2 bits and then added to higher 4 bits of the delay slot's address to make a target address. The program unconditionally jumps to the target address with a delay of one instruction. The address of the instruction that follows the delay slot is stored to the link register (r31). The ISA mode bit is inverted with a delay of one instruction. The value of bit 0 of the link register (r31) indicates the ISA mode bit before jump.

Operation:

```
32   T:    temp ← target
           GPR [31] ← $(PC + 8)_{31..1}$ || ISA MODE
     T+1:  PC ← $PC_{31..28}$ || temp || $0^2$
           ISA MODE toggle

64   T:    temp ← target
           GPR [31] ← $(PC + 8)_{63..1}$ || ISA MODE
     T+1:  PC ← $PC_{63..28}$ || temp || $0^2$
           ISA MODE toggle
```

Exceptions:

Reserved instruction exception (when MIPS16 instruction execution disabled)

JR Jump Register JR

31 26	25 21	20 6	5 0
SPECIAL 000000	rs	0 000 0000 0000 0000	JR 001000
6	5	15	6

Format:

JR rs

Description:

The program unconditionally jumps to the address contained in general register *rs*, with a delay of one instruction.

When a MIPS16 instruction can be executed, the program unconditionally jumps with a delay of one instruction to the address indicated by the value of clearing the least significant bit of the general register *rs* to 0. Then, the content of the least significant bit of the general register *rs* is set to the ISA mode bit (internal).

Since 32-bit length instructions must be word-aligned, a JR instruction must specify a target register (*rs*) that contains an address whose two low-order bits are zero when a MIPS16 instruction can be executed. If these low-order bits are not zero, an address error exception will occur when the jump target instruction is subsequently fetched.

Operation:

```
32, 64  T:      temp ← GPR [rs]
        T+1:    If MIPS16EN = 1 then
                    PC ← $temp_{63..1}$ || 0
                    ISA MODE ← $temp_0$
                else
                    PC ← temp
                endif
```

Exceptions:

None

LB Load Byte LB

31 26	25 21	20 16	15 0
LB 1 0 0 0 0 0	base	rt	offset
6	5	5	16

Format:

LB rt, offset (base)

Description:

The 16-bit *offset* is sign-extended and added to the contents of general register *base* to form a virtual address. The contents of the byte at the memory location specified by the effective address are sign-extended and loaded into general register *rt*.

Operation:

```
32   T:   $vAddr \leftarrow ((offset_{15})^{16} \| offset_{15..0}) + GPR [base]$
          $(pAddr, uncached) \leftarrow AddressTranslation (vAddr, DATA)$
          $pAddr \leftarrow pAddr_{PSIZE - 1..3} \| (pAddr_{2..0}$ xor $ReverseEndian^{3})$
          $mem \leftarrow LoadMemory (uncached, BYTE, pAddr, vAddr, DATA)$
          $byte \leftarrow vAddr_{2..0}$ xor $BigEndianCPU^{3}$
          $GPR [rt] \leftarrow (mem_{7 + 8* byte})^{24} \| mem_{7 + 8* byte..8* byte}$

64   T:   $vAddr \leftarrow ((offset_{15})^{48} \| offset_{15..0}) + GPR [base]$
          $(pAddr, uncached) \leftarrow AddressTranslation (vAddr, DATA)$
          $pAddr \leftarrow pAddr_{PSIZE - 1..3} \| (pAddr_{2..0}$ xor $ReverseEndian^{3})$
          $mem \leftarrow LoadMemory (uncached, BYTE, pAddr, vAddr, DATA)$
          $byte \leftarrow vAddr_{2..0}$ xor $BigEndianCPU^{3}$
          $GPR [rt] \leftarrow (mem_{7 + 8* byte})^{56} \| mem_{7 + 8* byte..8* byte}$
```

Exceptions:

TLB refill exception
TLB invalid exception
Bus error exception
Address error exception

LBU Load Byte Unsigned LBU

31 26	25 21	20 16	15 0
LBU 1 0 0 1 0 0	base	rt	offset
6	5	5	16

Format:

LBU rt, offset (base)

Description:

The 16-bit *offset* is sign-extended and added to the contents of general register *base* to form a virtual address. The contents of the byte at the memory location specified by the effective address are zero-extended and loaded into general register *rt*.

Operation:

```
32   T:   vAddr ← $((offset_{15})^{16} || offset_{15..0})$ + GPR [base]
          (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
          pAddr ← $pAddr_{PSIZE - 1..3} || (pAddr_{2..0}$ xor $ReverseEndian^3)$
          mem ← LoadMemory (uncached, BYTE, pAddr, vAddr, DATA)
          byte ← $vAddr_{2..0}$ xor $BigEndianCPU^3$
          GPR [rt] ← $0^{24} || mem_{7 + 8* byte..8* byte}$

64   T:   vAddr ← $((offset_{15})^{48} || offset_{15..0})$ + GPR [base]
          (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
          pAddr ← $pAddr_{PSIZE - 1..3} || (pAddr_{2..0}$ xor $ReverseEndian^3)$
          mem ← LoadMemory (uncached, BYTE, pAddr, vAddr, DATA)
          byte ← $vAddr_{2..0}$ xor $BigEndianCPU^3$
          GPR [rt] ← $0^{56} || mem_{7 + 8* byte..8* byte}$
```

Exceptions:

TLB refill exception
TLB invalid exception
Bus error exception
Address error exception

LD Load Doubleword LD

31 26	25 21	20 16	15 0
LD 1 1 0 1 1 1	base	rt	offset
6	5	5	16

Format:

LD rt, offset (base)

Description:

The 16-bit *offset* is sign-extended and added to the contents of general register *base* to form a virtual address. The contents of the 64-bit doubleword at the memory location specified by the effective address are loaded into general register *rt*.

If any of the three least-significant bits of the effective address are non-zero, an address error exception occurs.

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

Operation:

```
64    T:    vAddr ← $((\text{offset}_{15})^{48} \,||\, \text{offset}_{15..0})$ + GPR [base]
            (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
            data ← LoadMemory (uncached, DOUBLEWORD, pAddr, vAddr, DATA)
            GPR [rt] ← data
```

Exceptions:

TLB refill exception

TLB invalid exception

Bus error exception

Address error exception

Reserved instruction exception (32-bit user mode/supervisor mode)

LDL Load Doubleword Left (1/3) LDL

31 26	25 21	20 16	15 0
LDL 0 1 1 0 1 0	base	rt	offset
6	5	5	16

Format:

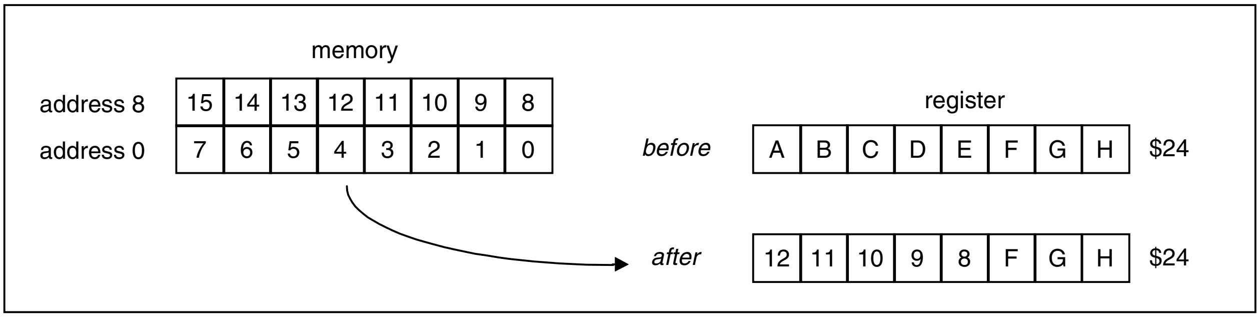
LDL rt, offset (base)

Description:

This instruction can be used in combination with the LDR instruction to load a register with eight consecutive bytes from memory, when the bytes cross a doubleword boundary. LDL loads the left portion of the register with the appropriate part of the high-order doubleword; LDR loads the right portion of the register with the appropriate part of the low-order doubleword.

The LDL instruction adds its sign-extended 16-bit *offset* to the contents of general register *base* to form a virtual address that can specify an arbitrary byte. It reads bytes only from the doubleword in memory that contains the specified starting byte. From one to eight bytes will be loaded, depending on the starting byte specified.

Conceptually, it starts at the specified byte in memory and loads that byte into the high-order (left-most) byte of the register; then it loads bytes from memory into the register until it reaches the low-order byte of the doubleword in memory. The least-significant (right-most) byte(s) of the register will not be changed.



LDL Load Doubleword Left (2/3) LDL

The contents of general register *rt* are internally bypassed within the processor so that no NOP is needed between an immediately preceding load instruction which specifies register *rt* and a following LDL (or LDR) instruction which also specifies register *rt*.

No address error exceptions due to alignment are possible.

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

Operation:

```
64    T:    $vAddr \leftarrow ((offset_{15})^{48} \,||\, offset_{15..0}) + GPR\ [base]$
            $(pAddr, uncached) \leftarrow AddressTranslation\ (vAddr, DATA)$
            $pAddr \leftarrow pAddr_{PSIZE - 1..3} \,||\, (pAddr_{2..0}\ xor\ ReverseEndian^{3})$
            if BigEndianMem = 0 then
                $pAddr \leftarrow pAddr_{PSIZE - 1..3} \,||\, 0^{3}$
            endif
            $byte \leftarrow vAddr_{2..0}\ xor\ BigEndianCPU^{3}$
            $mem \leftarrow LoadMemory\ (uncached, byte, pAddr, vAddr, DATA)$
            $GPR\ [rt] \leftarrow mem_{7 + 8 * byte..0} \,||\, GPR\ [rt]_{55 - 8 * byte..0}$
```

LDL Load Doubleword Left (3/3) LDL

Given a doubleword in a register and a doubleword in memory, the operation of LDL is as follows:

LDL

Register	A	B	C	D	E	F	G	H
Memory	I	J	K	L	M	N	O	P

vAddr2..0	Destination	Type	Offset (LEM)
0	PBCDEFGH	0	0
1	OPCDEFGH	1	0
2	NOPDEFGH	2	0
3	MNOPEFGH	3	0
4	LMNOPFGH	4	0
5	KLMNOPGH	5	0
6	JKLMNOPH	6	0
7	IJKLMNOP	7	0

Remark *LEM* Little-endian memory (BigEndianMem = 0)

Type AccessType (see **Table 2-3. Byte Specification Related to Load and Store Instructions**) sent to memory

Offset pAddr2..0 sent to memory

Exceptions:

TLB refill exception

TLB invalid exception

Bus error exception

Address error exception

Reserved instruction exception (32-bit user mode/supervisor mode)

LDR Load Doubleword Right (1/3) LDR

31 26	25 21	20 16	15 0
LDR 0 1 1 0 1 1	base	rt	offset
6	5	5	16

Format:

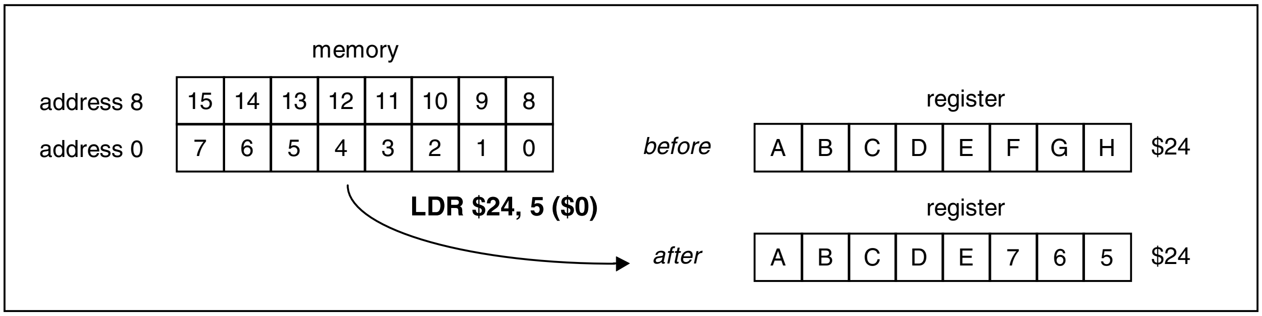
LDR rt, offset (base)

Description:

This instruction can be used in combination with the LDL instruction to load a register with eight consecutive bytes from memory, when the bytes cross a doubleword boundary. LDL instruction loads the high-order portion of data and LDR instruction loads the low-order portion of data.

The LDR instruction adds its sign-extended 16-bit *offset* to the contents of general register *base* to form a virtual address that can specify an arbitrary byte. It reads bytes only from the doubleword in memory that contains the specified starting byte. From one to eight bytes will be loaded, depending on the starting byte specified.

Conceptually, it starts at the specified byte in memory and loads that byte into the low-order (right-most) byte of the register; then it loads bytes from memory into the register until it reaches the high-order byte of the doubleword in memory. The most significant (left-most) byte(s) of the register will not be changed.



LDR Load Doubleword Right (2/3) LDR

The contents of general register *rt* are internally bypassed within the processor so that no NOP is needed between an immediately preceding load instruction which specifies register *rt* and a following LDR (or LDL) instruction which also specifies register *rt*.

No address error exceptions due to alignment are possible.

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

Operation:

```
64    T:    vAddr ← $((offset_{15})^{48} \| offset_{15..0})$ + GPR [base]
            (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
            pAddr ← $pAddr_{PSIZE - 1..3} \| (pAddr_{2..0}$ xor $ReverseEndian^{3})$
            if BigEndianMem = 1 then
                  pAddr ← $pAddr_{PSIZE - 1..3} \| 0^{3}$
            endif
            byte ← $vAddr_{2..0}$ xor $BigEndianCPU^{3}$
            mem ← LoadMemory (uncached, DOUBLEWORD-byte, pAddr, vAddr, DATA)
            GPR [rt] ← $GPR [rt]_{63..64 - 8 * byte} \| mem_{63..8 * byte}$
```

LDR Load Doubleword Right (3/3) LDR

Given a doubleword in a register and a doubleword in memory, the operation of LDR is as follows:

LDR

Register	A	B	C	D	E	F	G	H
Memory	I	J	K	L	M	N	O	P

vAddr2..0	Destination	Type	Offset (LEM)
0	I J K L M N O P	7	0
1	A I J K L M N O	6	1
2	A B I J K L M N	5	2
3	A B C I J K L M	4	3
4	A B C D I J K L	3	4
5	A B C D E I J K	2	5
6	A B C D E F I J	1	6
7	A B C D E F G I	0	7

Remark *LEM* Little-endian memory (BigEndianMem = 0)

Type AccessType (see **Table 2-3. Byte Specification Related to Load and Store Instructions**) sent to memory

Offset pAddr2..0 sent to memory

Exceptions:

TLB refill exception

TLB invalid exception

Bus error exception

Address error exception

Reserved instruction exception (32-bit user mode/supervisor mode)

LH Load Halfword LH

31 26	25 21	20 16	15 0
LH 1 0 0 0 0 1	base	rt	offset
6	5	5	16

Format:

LH rt, offset (base)

Description:

The 16-bit *offset* is sign-extended and added to the contents of general register *base* to form a virtual address. The contents of the halfword at the memory location specified by the effective address are sign-extended and loaded into general register *rt*.

If the least-significant bit of the effective address is non-zero, an address error exception occurs.

Operation:

```
32   T:   $vAddr \leftarrow ((offset_{15})^{16} \| offset_{15...0}) + GPR[base]$
          $(pAddr, uncached) \leftarrow AddressTranslation(vAddr, DATA)$
          $pAddr \leftarrow pAddr_{PSIZE-1...3} \| (pAddr_{2...0} \text{ xor } (ReverseEndian^{2} \| 0))$
          $mem \leftarrow LoadMemory(uncached, HALFWORD, pAddr, vAddr, DATA)$
          $byte \leftarrow vAddr_{2...0} \text{ xor } (BigEndianCPU^{2} \| 0)$
          $GPR[rt] \leftarrow (mem_{15+8*byte})^{16} \| mem_{15+8*byte...8*byte}$

64   T:   $vAddr \leftarrow ((offset_{15})^{48} \| offset_{15...0}) + GPR[base]$
          $(pAddr, uncached) \leftarrow AddressTranslation(vAddr, DATA)$
          $pAddr \leftarrow pAddr_{PSIZE-1...3} \| (pAddr_{2...0} \text{ xor } (ReverseEndian^{2} \| 0))$
          $mem \leftarrow LoadMemory(uncached, HALFWORD, pAddr, vAddr, DATA)$
          $byte \leftarrow vAddr_{2...0} \text{ xor } (BigEndianCPU^{2} \| 0)$
          $GPR[rt] \leftarrow (mem_{15+8*byte})^{48} \| mem_{15+8*byte...8*byte}$
```

Exceptions:

TLB refill exception

TLB invalid exception

Bus error exception

Address error exception

LHU Load Halfword Unsigned LHU

31 … 26	25 … 21	20 … 16	15 … 0
LHU 1 0 0 1 0 1	base	rt	offset
6	5	5	16

Format:

LHU rt, offset (base)

Description:

The 16-bit *offset* is sign-extended and added to the contents of general register *base* to form a virtual address. The contents of the halfword at the memory location specified by the effective address are zero-extended and loaded into general register *rt*.

If the least-significant bit of the effective address is non-zero, an address error exception occurs.

Operation:

```
32    T:    vAddr ← $((\text{offset}_{15})^{16} \,||\, \text{offset}_{15...0}) + \text{GPR [base]}$
            (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
            pAddr ← $\text{pAddr}_{\text{PSIZE - 1...3}} \,||\, (\text{pAddr}_{2...0} \text{ xor } (\text{ReverseEndian}^2 \,||\, 0))$
            mem ← LoadMemory (uncached, HALFWORD, pAddr, vAddr, DATA)
            byte ← $\text{vAddr}_{2...0} \text{ xor } (\text{BigEndianCPU}^2 \,||\, 0)$
            GPR [rt] ← $0^{16} \,||\, \text{mem}_{15 + 8 * \text{byte}...8 * \text{byte}}$

64    T:    vAddr ← $((\text{offset}_{15})^{48} \,||\, \text{offset}_{15...0}) + \text{GPR [base]}$
            (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
            pAddr ← $\text{pAddr}_{\text{PSIZE - 1...3}} \,||\, (\text{pAddr}_{2...0} \text{ xor } (\text{ReverseEndian}^2 \,||\, 0))$
            mem ← LoadMemory (uncached, HALFWORD, pAddr, vAddr, DATA)
            byte ← $\text{vAddr}_{2...0} \text{ xor } (\text{BigEndianCPU}^2 \,||\, 0)$
            GPR [rt] ← $0^{48} \,||\, \text{mem}_{15 + 8 * \text{byte}...8 * \text{byte}}$
```

Exceptions:

TLB refill exception
TLB invalid exception
Bus Error exception
Address error exception

LUI Load Upper Immediate LUI

31 26	25 21	20 16	15 0
LUI 0 0 1 1 1 1	0 0 0 0 0 0	rt	immediate
6	5	5	16

Format:

LUI rt, immediate

Description:

The 16-bit *immediate* is shifted left 16 bits and concatenated to 16 bits of zeros. The result is placed into general register *rt*. In 64-bit mode, the loaded word is sign-extended.

Operation:

```
32    T:    GPR [rt] ← immediate || $0^{16}$

64    T:    GPR [rt] ← $(\text{immediate}_{15})^{32}$ || immediate || $0^{16}$
```

Exceptions:

None

LW Load Word LW

31 26	25 21	20 16	15 0
LW 1 0 0 0 1 1	base	rt	offset
6	5	5	16

Format:

LW rt, offset (base)

Description:

The 16-bit *offset* is sign-extended and added to the contents of general register *base* to form a virtual address. The contents of the word at the memory location specified by the effective address are loaded into general register *rt*. In 64-bit mode, the loaded word is sign-extended.

If either of the two least-significant bits of the effective address is non-zero, an address error exception occurs.

Operation:

```
32   T:   $vAddr \leftarrow ((offset_{15})^{16} \| offset_{15...0}) + GPR\ [base]$
          $(pAddr, uncached) \leftarrow AddressTranslation\ (vAddr, DATA)$
          $pAddr \leftarrow pAddr_{PSIZE - 1...3} \| (pAddr_{2...0}\ xor\ (ReverseEndian \| 0^2))$
          $mem \leftarrow LoadMemory\ (uncached, WORD, pAddr, vAddr, DATA)$
          $byte \leftarrow vAddr_{2...0}\ xor\ (BigEndianCPU \| 0^2)$
          $GPR\ [rt] \leftarrow mem_{31 + 8 * byte...8 * byte}$

64   T:   $vAddr \leftarrow ((offset_{15})^{48} \| offset_{15...0}) + GPR\ [base]$
          $(pAddr, uncached) \leftarrow AddressTranslation\ (vAddr, DATA)$
          $pAddr \leftarrow pAddr_{PSIZE - 1...3} \| (pAddr_{2...0}\ xor\ (ReverseEndian \| 0^2))$
          $mem \leftarrow LoadMemory\ (uncached, WORD, pAddr, vAddr, DATA)$
          $byte \leftarrow vAddr_{2...0}\ xor\ (BigEndianCPU \| 0^2)$
          $GPR\ [rt] \leftarrow (mem_{31 + 8 * byte})^{32} \| mem_{31 + 8 * byte...8 * byte}$
```

Exceptions:

TLB refill exception
TLB invalid exception
Bus error exception
Address error exception

LWL Load Word Left (1/3) LWL

31 26	25 21	20 16	15 0
LWL 1 0 0 0 1 0	base	rt	offset
6	5	5	16

Format:

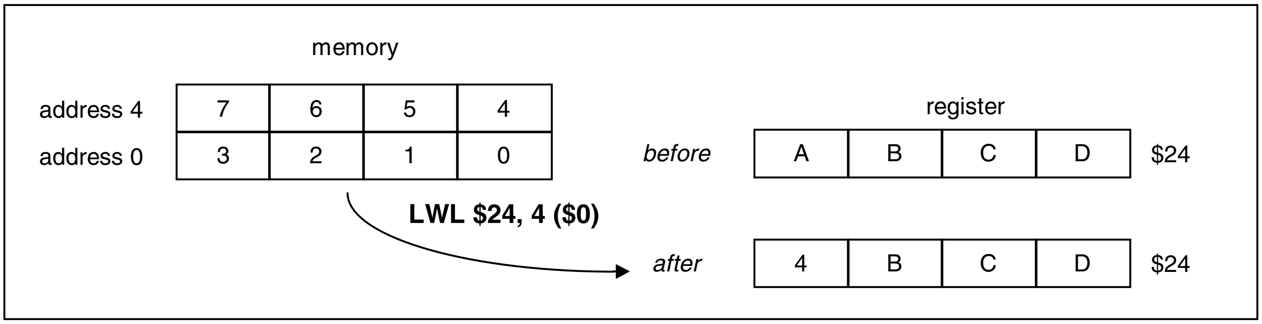
LWL rt, offset (base)

Description:

This instruction can be used in combination with the LWR instruction to load a register with four consecutive bytes from memory, when the bytes cross a word boundary. LWL loads the left portion of the register with the appropriate part of the high-order word; LWR loads the right portion of the register with the appropriate part of the low-order word.

The LWL instruction adds its sign-extended 16-bit *offset* to the contents of general register *base* to form a virtual address that can specify an arbitrary byte. It reads bytes only from the word in memory that contains the specified starting byte. From one to four bytes will be loaded, depending on the starting byte specified. In 64-bit mode, the loaded word is sign-extended.

Conceptually, it starts at the specified byte in memory and loads that byte into the high-order (left-most) byte of the register; then it loads bytes from memory into the register until it reaches the low-order byte of the word in memory. The least-significant (right-most) byte(s) of the register will not be changed.



LWL Load Word Left (2/3) LWL

The contents of general register *rt* are internally bypassed within the processor so that no NOP is needed between an immediately preceding load instruction which specifies register *rt* and a following LWL (or LWR) instruction which also specifies register *rt*.

No address error exceptions due to alignment are possible.

Operation:

```
32   T:   $vAddr \leftarrow ((offset_{15})^{16} \parallel offset_{15...0}) + GPR [base]$
          $(pAddr, uncached) \leftarrow AddressTranslation (vAddr, DATA)$
          $pAddr \leftarrow pAddr_{PSIZE - 1...3} \parallel (pAddr_{2...0}$ xor $ReverseEndian^{3})$
          if $BigEndianMem = 0$ then
                $pAddr \leftarrow pAddr_{PSIZE - 1...2} \parallel 0^{2}$
          endif
          $byte \leftarrow vAddr_{1...0}$ xor $BigEndianCPU^{2}$
          $word \leftarrow vAddr_{2}$ xor $BigEndianCPU$
          $mem \leftarrow LoadMemory (uncached, byte, pAddr, vAddr, DATA)$
          $temp \leftarrow mem_{32 * word + 8 * byte + 7...32 * word} \parallel GPR [rt]_{23 - 8 * byte...0}$
          $GPR [rt] \leftarrow temp$

64   T:   $vAddr \leftarrow ((offset_{15})^{48} \parallel offset_{15...0}) + GPR [base]$
          $(pAddr, uncached) \leftarrow AddressTranslation (vAddr, DATA)$
          $pAddr \leftarrow pAddr_{PSIZE - 1...3} \parallel (pAddr_{2...0}$ xor $ReverseEndian^{3})$
          if $BigEndianMem = 0$ then
                $pAddr \leftarrow pAddr_{PSIZE - 1...2} \parallel 0^{2}$
          endif
          $byte \leftarrow vAddr_{1...0}$ xor $BigEndianCPU^{2}$
          $word \leftarrow vAddr_{2}$ xor $BigEndianCPU$
          $mem \leftarrow LoadMemory (uncached, 0 \parallel byte, pAddr, vAddr, DATA)$
          $temp \leftarrow mem_{32 * word + 8 * byte + 7...32 * word} \parallel GPR [rt]_{23 - 8 * byte...0}$
          $GPR [rt] \leftarrow (temp_{31})^{32} \parallel temp$
```

LWL Load Word Left (3/3) LWL

Given a doubleword in a register and a doubleword in memory, the operation of LWL is as follows:

LWL

Register	A	B	C	D	E	F	G	H
Memory	I	J	K	L	M	N	O	P

vAddr2..0	Destination	Type	Offset (LEM)
0	S S S S P F G H	0	0
1	S S S S O P G H	1	0
2	S S S S N O P H	2	0
3	S S S S M N O P	3	0
4	S S S S L F G H	0	4
5	S S S S K L G H	1	4
6	S S S S J K L H	2	4
7	S S S S I J K L	3	4

Remark *LEM* Little-endian memory (BigEndianMem = 0)

Type AccessType (see **Table 2-3. Byte Specification Related to Load and Store Instructions**) sent to memory

Offset pAddr2..0 sent to memory

S sign-extend of destination bit 31

Exceptions:

TLB refill exception

TLB invalid exception

Bus error exception

Address error exception

LWR Load Word Right (1/3) LWR

31 26	25 21	20 16	15 0
LWR 1 0 0 1 1 0	base	rt	offset
6	5	5	16

Format:

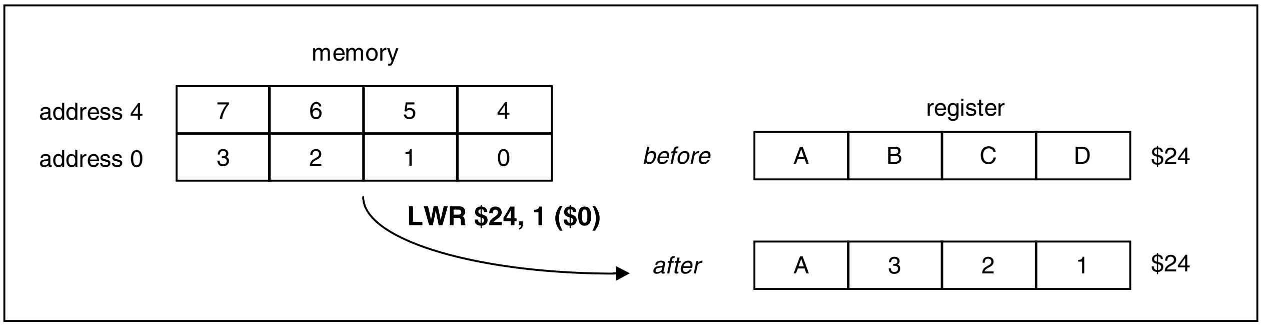
LWR rt, offset (base)

Description:

This instruction can be used in combination with the LWL instruction to load a register with four consecutive bytes from memory, when the bytes cross a word boundary. LWR loads the right portion of the register with the appropriate part of the low-order word; LWL loads the left portion of the register with the appropriate part of the high-order word.

The LWR instruction adds its sign-extended 16-bit *offset* to the contents of general register *base* to form a virtual address that can specify an arbitrary byte. It reads bytes only from the word in memory that contains the specified starting byte. From one to four bytes will be loaded, depending on the starting byte specified. In 64-bit mode, the loaded word is sign-extended.

Conceptually, it starts at the specified byte in memory and loads that byte into the low-order (right-most) byte of the register; then it loads bytes from memory into the register until it reaches the high-order byte of the word in memory. The most significant (left-most) byte(s) of the register will not be changed.



LWR Load Word Right (2/3) LWR

The contents of general register *rt* are internally bypassed within the processor so that no NOP is needed between an immediately preceding load instruction which specifies register *rt* and a following LWR (or LWL) instruction which also specifies register *rt*.

No address error exceptions due to alignment are possible.

Operation:

```
32   T:   $vAddr \leftarrow ((offset_{15})^{16} \| offset_{15...0}) + GPR\ [base]$
          $(pAddr, uncached) \leftarrow AddressTranslation\ (vAddr, DATA)$
          $pAddr \leftarrow pAddr_{PSIZE - 1...3} \| (pAddr_{2...0}\ xor\ ReverseEndian^{3})$
          if $BigEndianMem = 1$ then
               $pAddr \leftarrow pAddr_{PSIZE - 1...3} \| 0^{3}$
          endif
          $byte \leftarrow vAddr_{1...0}\ xor\ BigEndianCPU^{2}$
          $word \leftarrow vAddr_{2}\ xor\ BigEndianCPU$
          $mem \leftarrow LoadMemory\ (uncached, 0 \| byte, pAddr, vAddr, DATA)$
          $temp \leftarrow GPR\ [rt]_{31...32 - 8 * byte} \| mem_{31 + 32 * word...32 * word + 8 * byte}$
          $GPR\ [rt] \leftarrow temp$

64   T:   $vAddr \leftarrow ((offset_{15})^{48} \| offset_{15...0}) + GPR\ [base]$
          $(pAddr, uncached) \leftarrow AddressTranslation\ (vAddr, DATA)$
          $pAddr \leftarrow pAddr_{PSIZE - 1...3} \| (pAddr_{2...0}\ xor\ ReverseEndian^{3})$
          if $BigEndianMem = 1$ then
               $pAddr \leftarrow pAddr_{PSIZE - 1...3} \| 0^{3}$
          endif
          $byte \leftarrow vAddr_{1...0}\ xor\ BigEndianCPU^{2}$
          $word \leftarrow vAddr_{2}\ xor\ BigEndianCPU$
          $mem \leftarrow LoadMemory\ (uncached, WORD-byte, pAddr, vAddr, DATA)$
          $temp \leftarrow GPR\ [rt]_{31...32 - 8 * byte} \| mem_{31 + 32 * word...32 * word + 8 * byte}$
          $GPR\ [rt] \leftarrow (temp_{31})^{32} \| temp$
```

LWR Load Word Right (3/3) LWR

Given a word in a register and a word in memory, the operation of LWR is as follows:

LWR

Register	A	B	C	D	E	F	G	H
Memory	I	J	K	L	M	N	O	P

vAddr2..0	Destination	Type	Offset (LEM)
0	S S S S M N O P	3	0
1	S S S S E M N O	2	1
2	S S S S E F M N	1	2
3	S S S S E F G M	0	3
4	S S S S I J K L	3	4
5	S S S S E I J K	2	5
6	S S S S E F I J	1	6
7	S S S S E F G I	0	7

Remark *LEM* Little-endian memory (BigEndianMem = 0)

Type AccessType (see **Table 2-3. Byte Specification Related to Load and Store Instructions**) sent to memory

Offset pAddr2..0 sent to memory

S sign-extend of destination31

Exceptions:

TLB refill exception

TLB invalid exception

Bus error exception

Address error exception

LWU Load Word Unsigned LWU

31 26	25 21	20 16	15 0
LWU 1 0 1 1 1 1	base	rt	offset
6	5	5	16

Format:

LWU rt, offset (base)

Description:

The 16-bit *offset* is sign-extended and added to the contents of general register *base* to form a virtual address. The contents of the word at the memory location specified by the effective address are loaded into general register *rt*. The loaded word is zero-extended.

If either of the two least-significant bits of the effective address is non-zero, an address error exception occurs.

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

Operation:

```
32   T:   $vAddr \leftarrow ((offset_{15})^{16} \,||\, offset_{15...0}) + GPR\ [base]$
          $(pAddr, uncached) \leftarrow AddressTranslation\ (vAddr, DATA)$
          $pAddr \leftarrow pAddr_{PSIZE - 1...3} \,||\, (pAddr_{2...0}\ xor\ (ReverseEndian \,||\, 0^2))$
          $mem \leftarrow LoadMemory\ (uncached, WORD, pAddr, vAddr, DATA)$
          $byte \leftarrow vAddr_{2...0}\ xor\ (BigEndianCPU \,||\, 0^2)$
          $GPR\ [rt] \leftarrow 0^{32} \,||\, mem_{31 + 8 * byte...8 * byte}$

64   T:   $vAddr \leftarrow ((offset_{15})^{48} \,||\, offset_{15...0}) + GPR\ [base]$
          $(pAddr, uncached) \leftarrow AddressTranslation\ (vAddr, DATA)$
          $pAddr \leftarrow pAddr_{PSIZE - 1...3} \,||\, (pAddr_{2...0}\ xor\ (ReverseEndian \,||\, 0^2))$
          $mem \leftarrow LoadMemory\ (uncached, WORD, pAddr, vAddr, DATA)$
          $byte \leftarrow vAddr_{2...0}\ xor\ (BigEndianCPU \,||\, 0^2)$
          $GPR\ [rt] \leftarrow 0^{32} \,||\, mem_{31 + 8 * byte...8 * byte}$
```

Exceptions:

TLB refill exception

TLB invalid exception

Bus error exception

Address error exception

Reserved instruction exception (32-bit user mode/supervisor mode)

MACC Multiply and Accumulate (1/5) MACC

31 26	25 21	20 16	15 11	10	9	8 7	6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	rd	sat	hi	0 0	us	MACC 1 0 1 0 0 0
6	5	5	5	1	1	2	1	6

Format:

MACC rd, rs, rt
MACCU rd, rs, rt
MACCHI rd, rs, rt
MACCHIU rd, rs, rt
MACCS rd, rs, rt
MACCUS rd, rs, rt
MACCHIS rd, rs, rt
MACCHIUS rd, rs, rt

Description:

MACC instruction differs mnemonics by each setting of op codes sat, hi and us as follows.

Mnemonic	sat	hi	us
MACC	0	0	0
MACCU	0	0	1
MACCHI	0	1	0
MACCHIU	0	1	1
MACCS	1	0	0
MACCUS	1	0	1
MACCHIS	1	1	0
MACCHIUS	1	1	1

The number of significant bits in the operands of the MACC instruction differ depending on whether saturation processing is executed (sat = 1) or not executed (sat = 0).

- When saturation processing is executed (sat = 1): MACCS, MACCUS, MACCHIS, MACCHIUS instructions
  The contents of general register *rs* is multiplied by the contents of general register *rt*. If both operands are set as "us = 1" (MACCUS, MACCHIUS instructions), the contents are handled as 16-bit unsigned data. If they are set as "us = 0" (MACCS, MACCHIS instructions), the contents are handled as 16-bit signed integers. Sign/zero expansion by software is required for any bits exceeding 16 bits in the operands. The product of this multiply operation is added to a 64-bit value (of which only the low-order 32 bits are valid) that is linked to the HI and LO special registers. If us = 1, this add operation handles the values being added as 32-bit unsigned data. If us = 0, the values are handled as 32-bit signed integers. Sign/zero expansion by software is required for any bits exceeding 32 bits in the linked HI and LO special registers. After saturation processing to 32 bits has been performed (see the table below), the sum from this add operation is loaded to the HI and LO special register. When hi = 1 (MACCHIS, MACCHIUS instructions), data that is the same as the data loaded to the HI special register is also loaded to the rd general register. When hi = 0 (MACCS, MACCUS instructions), data that is the same as the data loaded to the LO special register is also loaded to the rd general register. Overflow exceptions do not occur.

MACC Multiply and Accumulate (2/5) MACC

- When saturation processing is not executed (sat = 0): MACC, MACCU, MACCHI, MACCHIU instructions
  The contents of general register *rs* is multiplied to the contents of general register *rt*. If both operands are set as "us = 1" (MACCU, MACCHIU instructions), the contents are handled as 32 bit unsigned data. If they are set as "us = 0" (MACC, MACCHI instructions), the contents are handled as 32 bit signed integers. Sign/zero expansion by software is required for any bits exceeding 32 bits in the operands. The product of this multiply operation is added to a 64-bit value that is linked to the HI and LO special registers. If us = 1, this add operation handles the values being added as 64 bit unsigned data. If us = 0, the values are handled as 64 bit signed integers.
  The low-order word from the 64-bit sum from this add operation is loaded to the LO special register and the high-order word is loaded to the HI special register. When hi = 1 (MACCHI, MACCHIU instructions), data that is the same as the data loaded to the HI special register is also loaded to the rd general register. When hi = 0 (MACC, MACCU instructions), data that is the same as the data loaded to the LO special register is also loaded to the rd general register. Overflow exceptions do not occur.

The correspondence of us and sat settings and values stored during saturation processing is shown below, along with the hazard cycles required between execution of the instruction for manipulating the HI and LO registers and execution of the MACC instruction.

Values Stored during Saturation Processing

us	sat	Overflow	Underflow
0	0	Store calculation result as is	Store calculation result as is
1	0	Store calculation result as is	Store calculation result as is
0	1	0000 0000 7FFF FFFFH	FFFF FFFF 8000 0000H
1	1	FFFF FFFF FFFF FFFFH	None

Hazard Cycle Counts

Instruction	Cycle Count
MULT, MULTU	1
DMULT, DMULTU	3
DIV, DIVU	36
DDIV, DDIVU	68
MFHI, MFLO	2
MTHI, MTLO	0
MACC	0
DMACC	0

MACC Multiply and Accumulate (3/5) MACC

Operation:

```
32, sat=0, hi=0, us=0 (MACC instruction)
    T:  temp1 ← GPR[rs] * GPR[rt]
        temp2 ← temp1 + (HI || LO)
        LO ← $temp2_{63..32}$
        HI ← $temp2_{31..0}$
        GPR[rd] ← LO
32, sat=0, hi=0, us=1 (MACCU instruction)
    T:  temp1 ← (0 || GPR[rs]) * (0 || GPR[rt])
        temp2 ← temp1 + ((0 || HI) || (0 || LO))
        LO ← $temp2_{63..32}$
        HI ← $temp2_{31..0}$
        GPR[rd] ← LO
32, sat=0, hi=1, us=0 (MACCHI instruction)
    T:  temp1 ← GPR[rs] * GPR[rt]
        temp2 ← temp1 + (HI || LO)
        LO ← $temp2_{63..32}$
        HI ← $temp2_{31..0}$
        GPR[rd] ← HI
32, sat=0, hi=1, us=1 (MACCHIU instruction)
    T:  temp1 ← (0 || GPR[rs]) * (0 || GPR[rt])
        temp2 ← temp1 + ((0 || HI) || (0 || LO))
        LO ← $temp2_{63..32}$
        HI ← $temp2_{31..0}$
        GPR[rd] ← HI
32, sat=1, hi=0, us=0 (MACCS instruction)
    T:  temp1 ← GPR[rs] * GPR[rt]
        temp2 ← saturation(temp1 + (HI || LO))
        LO ← $temp2_{63..32}$
        HI ← $temp2_{31..0}$
        GPR[rd] ← LO
32, sat=1, hi=0, us=1 (MACCUS instruction)
    T:  temp1 ← (0 || GPR[rs]) * (0 || GPR[rt])
        temp2 ← saturation(temp1 + ((0 || HI) || (0 || LO)))
        LO ← $temp2_{63..32}$
        HI ← $temp2_{31..0}$
        GPR[rd] ← LO
```

MACC Multiply and Accumulate (4/5) MACC

```
32, sat=1, hi=1, us=0 (MACCHIS instruction)
    T:  temp1 ← GPR[rs] * GPR[rt]
        temp2 ← saturation(temp1 + (HI || LO))
        LO ← $temp2_{63..32}$
        HI ← $temp2_{31..0}$
        GPR[rd] ← HI
32, sat=1, hi=1, us=1 (MACCHIUS instruction)
    T:  temp1 ← (0 || GPR[rs]) * (0 || GPR[rt])
        temp2 ← saturation(temp1 + ((0 || HI) || (0 || LO)))
        LO ← $temp2_{63..32}$
        HI ← $temp2_{31..0}$
        GPR[rd] ← HI
64, sat=0, hi=0, us=0 (MACC instruction)
    T:  temp1 ← $((GPR[rs]_{31})^{32} \| GPR[rs]) * ((GPR[rt]_{31})^{32} \| GPR[rt])$
        temp2 ← $temp1 + (HI_{31..0} \| LO_{31..0})$
        LO ← $((temp2_{63})^{32} \| temp2_{63..32})$
        HI ← $((temp2_{31})^{32} \| temp2_{31..0})$
        GPR[rd] ← LO
64, sat=0, hi=0, us=1 (MACCU instruction)
    T:  temp1 ← $(0^{32} \| GPR[rs]) * (0^{32} \| GPR[rt])$
        temp2 ← $temp1 + (HI_{31..0} \| LO_{31..0})$
        LO ← $((temp2_{63})^{32} \| temp2_{63..32})$
        HI ← $((temp2_{31})^{32} \| temp2_{31..0})$
        GPR[rd] ← LO
64, sat=0, hi=1, us=0 (MACCHI instruction)
    T:  temp1 ← $((GPR[rs]_{31})^{32} \| GPR[rs]) * ((GPR[rt]_{31})^{32} \| GPR[rt])$
        temp2 ← $temp1 + (HI_{31..0} \| LO_{31..0})$
        LO ← $((temp2_{63})^{32} \| temp2_{63..32})$
        HI ← $((temp2_{31})^{32} \| temp2_{31..0})$
        GPR[rd] ← HI
64, sat=0, hi=1, us=1 (MACCHIU instruction)
    T:  temp1 ← $(0^{32} \| GPR[rs]) * (0^{32} \| GPR[rt])$
        temp2 ← $temp1 + (HI_{31..0} \| LO_{31..0})$
        LO ← $((temp2_{63})^{32} \| temp2_{63..32})$
        HI ← $((temp2_{31})^{32} \| temp2_{31..0})$
        GPR[rd] ← HI
```

MACC Multiply and Accumulate (5/5) MACC

```
64, sat=1, hi=0, us=0 (MACCS instruction)
    T:    $temp1 \leftarrow ((GPR[rs]_{31})^{32} \| GPR[rs]) * ((GPR[rt]_{31})^{32} \| GPR[rt])$
          $temp2 \leftarrow saturation(temp1 + (HI_{31..0} \| LO_{31..0}))$
          $LO \leftarrow ((temp2_{63})^{32} \| temp2_{63..32})$
          $HI \leftarrow ((temp2_{31})^{32} \| temp2_{31..0})$
          $GPR[rd] \leftarrow LO$
64, sat=1, hi=0, us=1 (MACCUS instruction)
    T:    $temp1 \leftarrow (0^{32} \| GPR[rs]) * (0^{32} \| GPR[rt])$
          $temp2 \leftarrow saturation(temp1 + (HI_{31..0} \| LO_{31..0}))$
          $LO \leftarrow ((temp2_{63})^{32} \| temp2_{63..32})$
          $HI \leftarrow ((temp2_{31})^{32} \| temp2_{31..0})$
          $GPR[rd] \leftarrow LO$
64, sat=1, hi=1, us=0 (MACCHIS instruction)
    T:    $temp1 \leftarrow ((GPR[rs]_{31})^{32} \| GPR[rs]) * ((GPR[rt]_{31})^{32} \| GPR[rt])$
          $temp2 \leftarrow saturation(temp1 + (HI_{31..0} \| LO_{31..0}))$
          $LO \leftarrow ((temp2_{63})^{32} \| temp2_{63..32})$
          $HI \leftarrow ((temp2_{31})^{32} \| temp2_{31..0})$
          $GPR[rd] \leftarrow HI$
64, sat=1, hi=1, us=1 (MACCHIUS instruction)
    T:    $temp1 \leftarrow (0^{32} \| GPR[rs]) * (0^{32} \| GPR[rt])$
          $temp2 \leftarrow saturation(temp1 + (HI_{31..0} \| LO_{31..0})$
          $LO \leftarrow ((temp2_{63})^{32} \| temp2_{63..32})$
          $HI \leftarrow ((temp2_{31})^{32} \| temp2_{31..0})$
          $GPR[rd] \leftarrow HI$
```

Exceptions:

None

MFC0 Move From System Control Coprocessor MFC0

31 26	25 21	20 16	15 11	10 0
COP0 0 1 0 0 0 0	MF 0 0 0 0 0	rt	rd	0 0 0 0 0 0 0 0 0 0 0 0
6	5	5	5	11

Format:

MFC0 rt, rd

Description:

The contents of coprocessor register *rd* of the CP0 are loaded into general register *rt.*

Operation:

```
32   T:     data ← CPR [0, rd]
     T+1:   GPR [rt] ← data

64   T:     data ← CPR [0, rd]
     T+1:   GPR [rt] ← $(data_{31})^{32}$ || $data_{31...0}$
```

Exceptions:

Coprocessor unusable exception (in 64-bit/32-bit user and supervisor mode if CP0 not enabled)

MFHI Move From HI MFHI

31 26	25 16	15 11	10 6	5 0
SPECIAL 000000	0 00 0000 0000	rd	0 00000	MFHI 010000
6	10	5	5	6

Format:

MFHI rd

Description:

The contents of special register *HI* are loaded into general register *rd*.

To ensure proper operation in the event of interruptions, the two instructions which follow a MFHI instruction may not be any of the instructions which modify the *HI* register: MULT, MULTU, DIV, DIVU, MTHI, DMULT, DMULTU, DDIV, DDIVU.

Operation:

```
32, 64  T:    GPR [rd] ← HI
```

Exceptions:

None

MFLO Move From LO MFLO

31 26	25 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0	rd	0 0 0 0 0 0	MFLO 0 1 0 0 1 0
6	10	5	5	6

Format:

MFLO rd

Description:

The contents of special register *LO* are loaded into general register *rd*.

To ensure proper operation in the event of interruptions, the two instructions which follow a MFLO instruction may not be any of the instructions which modify the *LO* register: MULT, MULTU, DIV, DIVU, MTLO, DMULT, DMULTU, DDIV, DDIVU.

Operation:

```
32, 64  T:     GPR [rd] ← LO
```

Exceptions:

None

MTC0 Move To Coprocessor0 MTC0

31 26	25 21	20 16	15 11	10 0
COP0 0 1 0 0 0 0	MT 0 0 1 0 0	rt	rd	0 0 0 0 0 0 0 0 0 0 0 0
6	5	5	5	11

Format:

MTC0 rt, rd

Description:

The contents of general register *rt* are loaded into coprocessor register *rd* of coprocessor 0.

Because the state of the virtual address translation system may be altered by this instruction, the operation of load instructions, store instructions, and TLB operations immediately prior to and after this instruction are undefined.

When using a register used by the MTC0 by means of instructions before and after it, refer to **APPENDIX B VR4120A COPROCESSOR 0 HAZARDS** and place the instructions in the appropriate location.

Operation:

```
32, 64  T:      data ← GPR [rt]
        T+1:    CPR [0, rd] ← data
```

Exceptions:

Coprocessor unusable exception (in 64-bit/32-bit user and supervisor mode if CP0 not enabled)

MTHI Move To HI MTHI

31 26	25 21	20 6	5 0
SPECIAL 0 0 0 0 0 0	rs	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	MTHI 0 1 0 0 0 1
6	5	15	6

Format:

MTHI rs

Description:

The contents of general register *rs* are loaded into special register *HI*.

If a MTHI operation is executed following a MULT, MULTU, DIV, or DIVU instruction, but before any MFLO, MFHI, MTLO, or MTHI instructions, the contents of special register *HI* are undefined.

Operation:

```
32, 64 T-2:  HI ← undefined
       T-1:  HI ← undefined
       T:    HI ← GPR [rs]
```

Exceptions:

None

MTLO Move To LO MTLO

31 26	25 21	20 6	5 0
SPECIAL 0 0 0 0 0 0	rs	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	MTLO 0 1 0 0 1 1
6	5	15	6

Format:

MTLO rs

Description:

The contents of general register *rs* are loaded into special register *LO.*

If an MTLO operation is executed following a MULT, MULTU, DIV, or DIVU instruction, but before any MFLO, MFHI, MTLO, or MTHI instructions, the contents of special register *LO* are undefined.

Operation:

```
32, 64 T-2:  LO ← undefined
       T-1:  LO ← undefined
       T:    LO ← GPR [rs]
```

Exceptions:

None

MULT Multiply MULT

31 26	25 21	20 16	15 6	5 0
SPECIAL 000000	rs	rt	0 00 0000 0000	MULT 011000
6	5	5	10	6

Format:

MULT rs, rt

Description:

The contents of general registers *rs* and *rt* are multiplied, treating both operands as signed 32-bit integer. No integer overflow exception occurs under any circumstances.

In 64-bit mode, the operands must be valid 32-bit, sign-extended values.

When the operation completes, the low-order word of the double result is loaded into special register *LO*, and the high-order word of the double result is loaded into special register *HI*.

If either of the two preceding instructions is MFHI or MFLO, the results of these instructions are undefined. Correct operation requires separating reads of *HI* or *LO* from writes by a minimum of two other instructions.

Operation:

```
32   T-2:  LO   ← undefined
           HI   ← undefined
     T-1:  LO   ← undefined
           HI   ← undefined
     T:    t    ← GPR [rs] * GPR [rt]
           LO   ← $t_{31...0}$
           HI   ← $t_{63...32}$

64   T-2:  LO   ← undefined
           HI   ← undefined
     T-1:  LO   ← undefined
           HI   ← undefined
     T:    t    ← GPR $[rs]_{31...0}$ * GPR $[rt]_{31...0}$
           LO   ← $(t_{31})^{32}$ || $t_{31...0}$
           HI   ← $(t_{63})^{32}$ || $t_{63...32}$
```

Exceptions:

None

MULTU Multiply Unsigned MULTU

31 26	25 21	20 16	15 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	0 0 0 0 0 0 0 0 0 0 0	MULTU 0 1 1 0 0 1
6	5	5	10	6

Format:

MULTU rs, rt

Description:

The contents of general register *rs* and the contents of general register *rt* are multiplied, treating both operands as unsigned values. No overflow exception occurs under any circumstances.

In 64-bit mode, the operands must be valid 32-bit, sign-extended values.

When the operation completes, the low-order word of the double result is loaded into special register *LO*, and the high-order word of the double result is loaded into special register *HI*.

If either of the two preceding instructions is MFHI or MFLO, the results of these instructions are undefined. Correct operation requires separating reads of *HI* or *LO* from writes by a minimum of two instructions.

Operation:

```
32   T-2:  LO   ← undefined
           HI   ← undefined
     T-1:  LO   ← undefined
           HI   ← undefined
     T:    t    ← (0 || GPR [rs]) * (0 || GPR [rt])
           LO   ← $t_{31...0}$
           HI   ← $t_{63...32}$

64   T-2:  LO   ← undefined
           HI   ← undefined
     T-1:  LO   ← undefined
           HI   ← undefined
     T:    t    ← (0 || GPR $[rs]_{31...0}$) * (0 || GPR $[rt]_{31...0}$)
           LO   ← $(t_{31})^{32}$ || $t_{31...0}$
           HI   ← $(t_{63})^{32}$ || $t_{63...32}$
```

Exceptions:

None

NOR Nor NOR

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	rd	0 0 0 0 0 0	NOR 1 0 0 1 1 1
6	5	5	5	5	6

Format:

NOR rd, rs, rt

Description:

The contents of general register *rs* are combined with the contents of general register *rt* in a bit-wise logical NOR operation. The result is placed into general register *rd*.

Operation:

```
32, 64  T:    GPR [rd] ← GPR [rs] nor GPR [rt]
```

Exceptions:

None

OR Or OR

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	rd	0 0 0 0 0 0	OR 1 0 0 1 0 1
6	5	5	5	5	6

Format:

OR rd, rs, rt

Description:

The contents of general register *rs* are combined with the contents of general register *rt* in a bit-wise logical OR operation. The result is placed into general register *rd*.

Operation:

```
32, 64  T:    GPR [rd] ← GPR [rs] or GPR [rt]
```

Exceptions:

None

ORI Or Immediate ORI

31 26	25 21	20 16	15 0
ORI 0 0 1 1 0 1	rs	rt	immediate
6	5	5	16

Format:

ORI rt, rs, immediate

Description:

The 16-bit *immediate* is zero-extended and combined with the contents of general register *rs* in a bit-wise logical OR operation. The result is placed into general register *rt*.

Operation:

```
32    T:    $GPR [rt] \leftarrow GPR [rs]_{31...16} \| (immediate\ or\ GPR [rs]_{15...0})$

64    T:    $GPR [rt] \leftarrow GPR [rs]_{63...16} \| (immediate\ or\ GPR [rs]_{15...0})$
```

Exceptions:

None

SB Store Byte SB

31 26	25 21	20 16	15 0
SB 1 0 1 0 0 0	base	rt	offset
6	5	5	16

Format:

SB rt, offset (base)

Description:

The 16-bit *offset* is sign-extended and added to the contents of general register *base* to form a virtual address. The least-significant byte of register *rt* is stored at the effective address.

Operation:

```
32   T:   $vAddr \leftarrow ((offset_{15})^{16} \| offset_{15...0}) + GPR\ [base]$
          $(pAddr, uncached) \leftarrow AddressTranslation\ (vAddr, DATA)$
          $pAddr \leftarrow pAddr_{PSIZE-1...3} \| (pAddr_{2...0}\ xor\ (ReverseEndian^3))$
          $byte \leftarrow vAddr_{2...0}\ xor\ BigEndianCPU^3$
          $data \leftarrow GPR\ [rt]_{63-8*byte...0} \| 0^{8*byte}$
          StoreMemory (uncached, BYTE, data, pAddr, vAddr, DATA)

64   T:   $vAddr \leftarrow ((offset_{15})^{48} \| offset_{15...0}) + GPR\ [base]$
          $(pAddr, uncached) \leftarrow AddressTranslation\ (vAddr, DATA)$
          $pAddr \leftarrow pAddr_{PSIZE-1...3} \| (pAddr_{2...0}\ xor\ (ReverseEndian^3))$
          $byte \leftarrow vAddr_{2...0}\ xor\ BigEndianCPU^3$
          $data \leftarrow GPR\ [rt]_{63-8*byte...0} \| 0^{8*byte}$
          StoreMemory (uncached, BYTE, data, pAddr,  vAddr, DATA)
```

Exceptions:

TLB refill exception

TLB invalid exception

TLB modification exception

Bus error exception

Address error exception

SD Store Doubleword SD

31 26	25 21	20 16	15 0
SD 1 1 1 1 1 1	base	rt	offset
6	5	5	16

Format:

SD rt, offset (base)

Description:

The 16-bit *offset* is sign-extended and added to the contents of general register *base* to form a virtual address. The contents of general register *rt* are stored at the memory location specified by the effective address.

If either of the three least-significant bits of the effective address are non-zero, an address error exception occurs.

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

Operation:

```
64    T:    vAddr ← $((\text{offset}_{15})^{48} \| \text{offset}_{15...0})$ + GPR [base]
            (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
            data ← GPR [rt]
            StoreMemory (uncached, DOUBLEWORD, data, pAddr, vAddr, DATA)
```

Exceptions:

TLB refill exception

TLB invalid exception

TLB modification exception

Bus error exception

Address error exception

Reserved instruction exception (32-bit user mode/supervisor mode)

SDL Store Doubleword Left (1/3) SDL

31 26	25 21	20 16	15 0
SDL 1 0 1 1 0 0	base	rt	offset
6	5	5	16

Format:

SDL rt, offset (base)

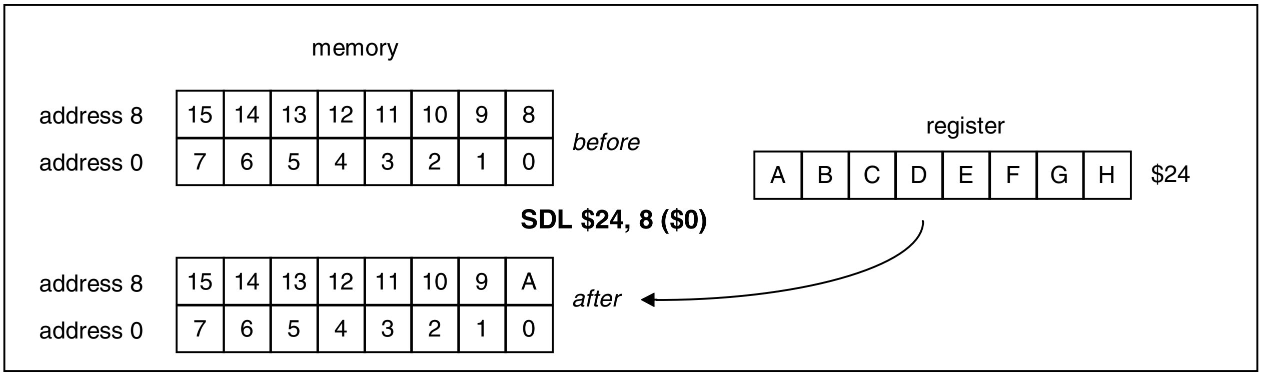
Description:

This instruction can be used with the SDR instruction to store the contents of a register into eight consecutive bytes of memory, when the bytes cross a doubleword boundary. SDL stores the register into the appropriate part of the high-order doubleword of memory; SDR stores the register into the appropriate part of the low-order doubleword.

The SDL instruction adds its sign-extended 16-bit *offset* to the contents of general register *base* to form a virtual address that may specify an arbitrary byte. It alters only the word in memory that contains that byte. From one to four bytes will be stored, depending on the starting byte specified.

Conceptually, it starts at the most-significant byte of the register and copies it to the specified byte in memory; then it copies bytes from register to memory until it reaches the low-order byte of the word in memory.

No address error exceptions due to alignment are possible.



SDL Store Doubleword Left (2/3) SDL

An address error exception is not occurred that specify address is not located in doubleword boundary.

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

Operation:

```
64   T:   vAddr ← ((offset$_{15}$)$^{48}$ || offset$_{15...0}$) + GPR [base]
          (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
          pAddr ← pAddr$_{PSIZE - 1...3}$ || (pAddr$_{2...0}$ xor ReverseEndian$^{3}$)
          if BigEndianMem = 0 then
               pAddr ← pAddr$_{PSIZE - 1...3}$ || 0$^{3}$
          endif
          byte ← vAddr$_{2...0}$ xor BigEndianCPU$^{3}$
          data ← 0$^{56 - 8 * byte}$ || GPR [rt]$_{63...56 - 8 * byte}$
          StoreMemory (uncached, byte, data, pAddr, vAddr, DATA)
```

SDL Store Doubleword Left (3/3) SDL

Given a doubleword in a register and a doubleword in memory, the operation of SDL instruction is as follows:

SDL

Register	A	B	C	D	E	F	G	H
Memory	I	J	K	L	M	N	O	P

vAddr2..0	Destination	Type	Offset (LEM)
0	I J K L M N O A	0	0
1	I J K L M N A B	1	0
2	I J K L M A B C	2	0
3	I J K L A B C D	3	0
4	I J K A B C D E	4	0
5	I J A B C D E F	5	0
6	I A B C D E F G	6	0
7	A B C D E F G H	7	0

Remark *LEM* Little-endian memory (BigEndianMem = 0)

Type AccessType (see **Table 2-3. Byte Specification Related to Load and Store Instructions**) sent to memory

Offset pAddr2..0 sent to memory

Exceptions:

TLB refill exception

TLB invalid exception

TLB modification exception

Bus error exception

Address error exception

Reserved instruction exception (32-bit user mode/supervisor mode)

SDR Store Doubleword Right (1/3) SDR

31 26	25 21	20 16	15 0
SDR 1 0 1 1 0 1	base	rt	offset
6	5	5	16

Format:

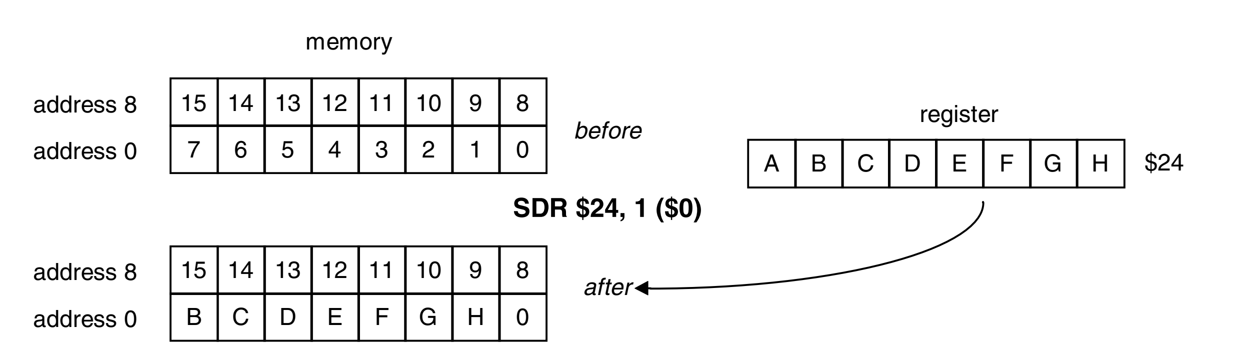
SDR rt, offset (base)

Description:

This instruction can be used with the SDL instruction to store the contents of a register into eight consecutive bytes of memory, when the bytes cross a boundary between two doublewords. SDR stores the register into the appropriate part of the low-order doubleword; SDL stores the register into the appropriate part of the low-order doubleword of memory.

The SDR instruction adds its sign-extended 16-bit *offset* to the contents of general register *base* to form a virtual address that may specify an arbitrary byte. It alters only the word in memory that contains that byte. From one to eight bytes will be stored, depending on the starting byte specified.

Conceptually, it starts at the least-significant byte of the register and copies it to the specified byte in memory; then it copies bytes from register to memory until it reaches the high-order byte of the word in memory. No address error exceptions due to alignment are possible.



SDR Store Doubleword Right (2/3) SDR

An address error exception is not occurred that specify address is not located in doubleword boundary.

This operation is defined in 64-bit mode or in 32-bit kernel mode. Execution of this instruction in 32-bit user or supervisor mode causes a reserved instruction exception.

Operation:

```
64    T:    $\text{vAddr} \leftarrow ((\text{offset}_{15})^{48} \,||\, \text{offset}_{15...0}) + \text{GPR [base]}$
            $(\text{pAddr, uncached}) \leftarrow \text{AddressTranslation (vAddr, DATA)}$
            $\text{pAddr} \leftarrow \text{pAddr}_{\text{PSIZE - 1...3}} \,||\, (\text{pAddr}_{2...0} \text{ xor ReverseEndian}^{3})$
            if BigEndianMem = 0 then
                $\text{pAddr} \leftarrow \text{pAddr}_{\text{PSIZE - 1...3}} \,||\, 0^{3}$
            endif
            $\text{byte} \leftarrow \text{vAddr}_{2...0} \text{ xor BigEndianCPU}^{3}$
            $\text{data} \leftarrow \text{GPR [rt]}_{63 - 8 * \text{byte}} \,||\, 0^{8 * \text{byte}}$
            StoreMemory (uncached, DOUBLEWORD-byte, data, pAddr, vAddr, DATA)
```

SDR Store Doubleword Right (3/3) SDR

Given a doubleword in a register and a doubleword in memory, the operation of SDR instruction is as follows:

SDR

Register	A	B	C	D	E	F	G	H
Memory	I	J	K	L	M	N	O	P

vAddr2..0	Destination	Type	Offset (LEM)
0	A B C D E F G H	7	0
1	B C D E F G H P	6	1
2	C D E F G H O P	5	2
3	D E F G H N O P	4	3
4	E F G H M N O P	3	4
5	F G H L M N O P	2	5
6	G H K L M N O P	1	6
7	H J K L M N O P	0	7

Remark *LEM* Little-endian memory (BigEndianMem = 0)

Type AccessType (see **Table 2-3. Byte Specification Related to Load and Store Instructions**) sent to memory

Offset pAddr2..0 sent to memory

Exceptions:

TLB refill exception

TLB invalid exception

TLB modification exception

Bus error exception

Address error exception

Reserved instruction exception (32-bit user mode/supervisor mode)

SH Store Halfword SH

31 26	25 21	20 16	15 0
SH 1 0 1 0 0 1	base	rt	offset
6	5	5	16

Format:

SH rt, offset (base)

Description:

The 16-bit *offset* is sign-extended and added to the contents of general register *base* to form an unsigned effective address. The least-significant halfword of register *rt* is stored at the effective address. If the least-significant bit of the effective address is non-zero, an address error exception occurs.

Operation:

```
32  T:  $vAddr \leftarrow ((offset_{15})^{16} \| offset_{15...0}) + GPR\ [base]$
        $(pAddr, uncached) \leftarrow AddressTranslation\ (vAddr, DATA)$
        $pAddr \leftarrow pAddr_{PSIZE - 1...3} \| (pAddr_{2...0}\ xor\ (ReverseEndian^2 \| 0))$
        $byte \leftarrow vAddr_{2...0}\ xor\ (BigEndianCPU^2 \| 0)$
        $data \leftarrow GPR\ [rt]_{63 - 8 * byte...0} \| 0^{8 * byte}$
        $StoreMemory\ (uncached, HALFWORD, data, pAddr, vAddr, DATA)$

64  T:  $vAddr \leftarrow ((offset_{15})^{48} \| offset_{15...0}) + GPR\ [base]$
        $(pAddr, uncached) \leftarrow AddressTranslation\ (vAddr, DATA)$
        $pAddr \leftarrow pAddr_{PSIZE - 1...3} \| (pAddr_{2...0}\ xor\ (ReverseEndian^2 \| 0))$
        $byte \leftarrow vAddr_{2...0}\ xor\ (BigEndianCPU^2 \| 0)$
        $data \leftarrow GPR\ [rt]_{63 - 8 * byte...0} \| 0^{8 * byte}$
        $StoreMemory\ (uncached, HALFWORD, data, pAddr, vAddr, DATA)$
```

Exceptions:

TLB refill exception
TLB invalid exception
TLB modification exception
Bus error exception
Address error exception

SLL Shift Left Logical SLL

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 000000	0 00000	rt	rd	sa	SLL 000000
6	5	5	5	5	6

Format:

SLL rd, rt, sa

Description:

The contents of general register *rt* are shifted left by *sa* bits, inserting zeros into the low-order bits.

The result is placed in register *rd.*

In 64-bit mode, the 32-bit result is sign-extended when placed in the destination register. It is sign extended for all shift amounts, including zero; SLL with zero shift amount truncates a 64-bit value to 32 bits and then sign extends this 32-bit value. SLL, unlike nearly all other word operations, does not require an operand to be a properly sign-extended word value to produce a valid sign-extended word result.

Operation:

```
32   T:   $GPR[rd] \leftarrow GPR[rt]_{31-sa...0} \parallel 0^{sa}$

64   T:   $s \leftarrow 0 \parallel sa$
          $temp \leftarrow GPR[rt]_{31-s...0} \parallel 0^{s}$
          $GPR[rd] \leftarrow (temp_{31})^{32} \parallel temp$
```

Exceptions:

None

Caution **SLL with a shift amount of zero may be treated as a NOP by some assemblers, at some optimization levels. If using SLL with a purpose of sign-extension, check the assembler specification.**

SLLV Shift Left Logical Variable SLLV

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 000000	rs	rt	rd	0 00000	SLLV 000100
6	5	5	5	5	6

Format:

SLLV rd, rt, rs

Description:

The contents of general register *rt* are shifted left the number of bits specified by the low-order five bits contained in general register *rs*, inserting zeros into the low-order bits.

The result is placed in register *rd*.

In 64-bit mode, the 32-bit result is sign-extended when placed in the destination register. It is sign extended for all shift amounts, including zero; SLLV with zero shift amount truncates a 64-bit value to 32 bits and then sign extends this 32-bit value. SLLV, unlike nearly all other word operations, does not require an operand to be a properly sign-extended word value to produce a valid sign-extended word result.

Operation:

```
32   T:   $s \leftarrow GPR[rs]_{4...0}$
          $GPR[rd] \leftarrow GPR[rt]_{(31-s)...0} \| 0^{s}$

64   T:   $s \leftarrow 0 \| GPR[rs]_{4...0}$
          $temp \leftarrow GPR[rt]_{(31-s)...0} \| 0^{s}$
          $GPR[rd] \leftarrow (temp_{31})^{32} \| temp$
```

Exceptions:

None

Caution SLLV with a shift amount of zero may be treated as a NOP by some assemblers, at some optimization levels. If using SLLV with a purpose of sign-extension, check the assembler specification.

SLT Set On Less Than SLT

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 000000	rs	rt	rd	0 00000	SLT 101010
6	5	5	5	5	6

Format:

SLT rd, rs, rt

Description:

The contents of general register *rt* are subtracted from the contents of general register *rs*. Considering both quantities as signed integers, if the contents of general register *rs* are less than the contents of general register *rt*, the result is set to one; otherwise the result is set to zero.

No integer overflow exception occurs under any circumstances. The comparison is valid even if the subtraction used during the comparison overflows.

Operation:

```
32    T:    if GPR [rs] < GPR [rt] then
                  GPR [rd] ← $0^{31}$ || 1
            else
                  GPR [rd] ← $0^{32}$
            endif

64    T:    if GPR [rs] < GPR [rt] then
                  GPR [rd] ← $0^{63}$ || 1
            else
                  GPR [rd] ← $0^{64}$
            endif
```

Exceptions:

None

SLTI Set On Less Than Immediate SLTI

31 26	25 21	20 16	15 0
SLTI 0 0 1 0 1 0	rs	rt	immediate
6	5	5	16

Format:

SLTI rt, rs, immediate

Description:

The 16-bit *immediate* is sign-extended and subtracted from the contents of general register *rs.* Considering both quantities as signed integers, if *rs* is less than the sign-extended *immediate*, the result is set to 1; otherwise the result is set to 0.

No integer overflow exception occurs under any circumstances. The comparison is valid even if the subtraction used during the comparison overflows.

Operation:

```
32    T:    if GPR [rs] < $(\text{immediate}_{15})^{16}$ || $\text{immediate}_{15...0}$ then
                 GPR [rt] ← $0^{31}$ || 1
            else
                 GPR [rt] ← $0^{32}$
            endif

64    T:    if GPR [rs] < $(\text{immediate}_{15})^{48}$ || $\text{immediate}_{15...0}$ then
                 GPR [rt] ← $0^{63}$ || 1
            else
                 GPR [rt] ← $0^{64}$
            endif
```

Exceptions:

None

SLTIU Set On Less Than Immediate Unsigned SLTIU

31 26	25 21	20 16	15 0
SLTIU 0 0 1 0 1 1	rs	rt	immediate
6	5	5	16

Format:

SLTIU rt, rs, immediate

Description:

The 16-bit *immediate* is sign-extended and subtracted from the contents of general register *rs.* Considering both quantities as unsigned integers, if *rs* is less than the sign-extended *immediate*, the result is set to 1; otherwise the result is set to 0.

No integer overflow exception occurs under any circumstances. The comparison is valid even if the subtraction used during the comparison overflows.

Operation:

```
32   T:   if (0 || GPR [rs]) < (0 || $(\text{immediate}_{15})^{16}$ || $\text{immediate}_{15...0}$) then
                GPR [rt] ← $0^{31}$ || 1
          else
                GPR [rt] ← $0^{32}$
          endif

64   T:   if (0 || GPR [rs]) < (0 || $(\text{immediate}_{15})^{48}$ || $\text{immediate}_{15...0}$) then
                GPR [rt] ← $0^{63}$ || 1
          else
                GPR [rt] ← $0^{64}$
          endif
```

Exceptions:

None

SLTU Set On Less Than Unsigned SLTU

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	rd	0 0 0 0 0 0	SLTU 1 0 1 0 1 1
6	5	5	5	5	6

Format:

SLTU rd, rs, rt

Description:

The contents of general register *rt* are subtracted from the contents of general register *rs.* Considering both quantities as unsigned integers, if the contents of general register *rs* are less than the contents of general register *rt*, the result is set to 1; otherwise the result is set to 0.

No integer overflow exception occurs under any circumstances. The comparison is valid even if the subtraction used during the comparison overflows.

Operation:

```
32    T:    if (0 || GPR [rs]) < 0 || GPR [rt] then
                  GPR [rd] ← $0^{31}$ || 1
            else
                  GPR [rd] ← $0^{32}$
            endif

64    T:    if (0 || GPR [rs]) < 0 || GPR [rt] then
                  GPR [rd] ← $0^{63}$ || 1
            else
                  GPR [rd] ← $0^{64}$
            endif
```

Exceptions:

None

SRA Shift Right Arithmetic SRA

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	0 0 0 0 0 0	rt	rd	sa	SRA 0 0 0 0 1 1
6	5	5	5	5	6

Format:

SRA rd, rt, sa

Description:

The contents of general register *rt* are shifted right by *sa* bits, sign-extending the high-order bits.

The result is placed in register *rd*.

In 64-bit mode, the operand must be a valid sign-extended, 32-bit value.

Operation:

```
32   T:   $GPR\ [rd] \leftarrow (GPR\ [rt]_{31})^{sa} \ || \ GPR\ [rt]_{31...sa}$

64   T:   $s \leftarrow 0 \ || \ sa$
          $temp \leftarrow (GPR\ [rt]_{31})^{s} \ || \ GPR\ [rt]_{31...s}$
          $GPR\ [rd] \leftarrow (temp_{31})^{32} \ || \ temp$
```

Exceptions:

None

SRAV Shift Right Arithmetic Variable SRAV

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	rd	0 0 0 0 0 0	SRAV 0 0 0 1 1 1
6	5	5	5	5	6

Format:

SRAV rd, rt, rs

Description:

The contents of general register *rt* are shifted right by the number of bits specified by the low-order five bits of general register *rs*, sign-extending the high-order bits.

The result is placed in register *rd*.

In 64-bit mode, the operand must be a valid sign-extended, 32-bit value.

Operation:

```
32   T:   $s \leftarrow GPR[rs]_{4...0}$
          $GPR[rd] \leftarrow (GPR[rt]_{31})^{s} || GPR[rt]_{31...s}$

64   T:   $s \leftarrow GPR[rs]_{4...0}$
          $temp \leftarrow (GPR[rt]_{31})^{s} || GPR[rt]_{31...s}$
          $GPR[rd] \leftarrow (temp_{31})^{32} || temp$
```

Exceptions:

None

SRL Shift Right Logical SRL

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	0 0 0 0 0 0	rt	rd	sa	SRL 0 0 0 0 1 0
6	5	5	5	5	6

Format:

SRL rd, rt, sa

Description:

The contents of general register *rt* are shifted right by *sa* bits, inserting zeros into the high-order bits.

The result is placed in register *rd*.

In 64-bit mode, the operand must be a valid sign-extended, 32-bit value.

Operation:

```
32   T:   $GPR [rd] \leftarrow 0^{sa} \| GPR [rt]_{31...sa}$

64   T:   $s \leftarrow 0 \| sa$
          $temp \leftarrow 0^{s} \| GPR [rt]_{31...s}$
          $GPR [rd] \leftarrow (temp_{31})^{32} \| temp$
```

Exceptions:

None

SRLV Shift Right Logical Variable SRLV

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	rd	0 0 0 0 0 0	SRLV 0 0 0 1 1 0
6	5	5	5	5	6

Format:

SRLV rd, rt, rs

Description:

The contents of general register *rt* are shifted right by the number of bits specified by the low-order five bits of general register *rs,* inserting zeros into the high-order bits.

The result is placed in register *rd*.

In 64-bit mode, the operand must be a valid sign-extended, 32-bit value.

Operation:

```
32    T:    $s \leftarrow \mathrm{GPR}\ [rs]_{4...0}$
            $\mathrm{GPR}\ [rd] \leftarrow 0^{s}\ ||\ \mathrm{GPR}\ [rt]_{31...s}$

64    T:    $s \leftarrow \mathrm{GPR}\ [rs]_{4...0}$
            $temp \leftarrow 0^{s}\ ||\ \mathrm{GPR}\ [rt]_{31...s}$
            $\mathrm{GPR}\ [rd] \leftarrow (temp_{31})^{32}\ ||\ temp$
```

Exceptions:

None

STANDBY Standby STANDBY

31 26	25	24 6	5 0
COP0 0 1 0 0 0 0	CO 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	STANDBY 1 0 0 0 0 1
6	1	19	6

Format:

STANDBY

Description:

STANDBY instruction starts mode transition from Fullspeed mode to Standby mode.

When the STANDBY instruction finishes the WB stage, this processor wait by the SysAD bus is idle state, after then the internal clocks will shut down, thus freezing the pipeline. The PLL, Timer/Interrupt clocks and the internal bus clocks (TClock and MasterOut) will continue to run.

Once this processor is in Standby mode, any interrupt, including the internally generated timer interrupt, NMI, Soft Reset, and Cold Reset will cause this processor to exit Standby mode and to enter Fullspeed mode.

Operation:

```
32, 64  T:
        T+1:  Standby operation ()
```

Exceptions:

Coprocessor unusable exception

Remark Refer to **Section 2.7.3.1 Power modes** for details about the operation of the peripheral units at mode transition.

SUB Subtract SUB

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 000000	rs	rt	rd	0 00000	SUB 100010
6	5	5	5	5	6

Format:

SUB rd, rs, rt

Description:

The contents of general register *rt* are subtracted from the contents of general register *rs* to form a result. The result is placed into general register *rd.* In 64-bit mode, the operands must be valid sign-extended, 32-bit values.

The only difference between this instruction and the SUBU instruction is that SUBU never traps on overflow.

An integer overflow exception takes place if the carries out of bits 30 and 31 differ (2's complement overflow). The destination register *rd* is not modified when an integer overflow exception occurs.

Operation:

```
32   T:   GPR [rd] ← GPR [rs] - GPR [rt]

64   T:   temp ← GPR [rs] - GPR [rt]
          GPR [rd] ← $(temp_{31})^{32}$ || $temp_{31...0}$
```

Exceptions:

Integer overflow exception

SUBU Subtract Unsigned SUBU

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 000000	rs	rt	rd	0 00000	SUBU 100011
6	5	5	5	5	6

Format:

SUBU rd, rs, rt

Description:

The contents of general register *rt* are subtracted from the contents of general register *rs* to form a result.

The result is placed into general register *rd*.

In 64-bit mode, the operands must be valid sign-extended, 32-bit values.

The only difference between this instruction and the SUB instruction is that SUBU never traps on overflow.

Operation:

```
32   T:   GPR [rd] ← GPR [rs] - GPR [rt]

64   T:   temp ← GPR [rs] - GPR [rt]
          $\mathrm{GPR}\ [rd] \leftarrow (\mathrm{temp}_{31})^{32} \,\|\, \mathrm{temp}_{31\ldots0}$
```

Exceptions:

None

SUSPEND Suspend SUSPEND

31 26	25	24 6	5 0
COP0 0 1 0 0 0 0	CO 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	SUSPEND 1 0 0 0 1 0
6	1	19	6

Format:

SUSPEND

Description:

SUSPEND instruction starts mode transition from Fullspeed mode to Suspend mode.

When the SUSPEND instruction finishes the WB stage, this processor wait by the SysAD bus is idle state, after then the internal clocks including the TClock will shut down, thus freezing the pipeline. The PLL, Timer/Interrupt clocks and MasterOut, will continue to run.

Once this processor is in Suspend mode, any interrupt, including the internally generated timer interrupt, NMI, Soft Reset and Cold Reset will cause this processor to exit Suspend mode and to enter Fullspeed mode.

Operation:

```
32, 64  T:
        T+1:  Suspend Operation ()
```

Exceptions:

Coprocessor unusable exception

Remark Refer to **Section 2.6.3.1 Power modes** for details about the operation of the peripheral units at mode transition.

SW Store Word SW

31 26	25 21	20 16	15 0
SW 1 0 1 0 1 1	base	rt	offset
6	5	5	16

Format:

SW rt, offset (base)

Description:

The 16-bit *offset* is sign-extended and added to the contents of general register *base* to form a virtual address. The contents of general register *rt* are stored at the memory location specified by the effective address.

If either of the two least-significant bits of the effective address are non-zero, an address error exception occurs.

Operation:

```
32   T:   $vAddr \leftarrow ((offset_{15})^{16} || offset_{15...0}) + GPR [base]$
          $(pAddr, uncached) \leftarrow AddressTranslation (vAddr, DATA)$
          $pAddr \leftarrow pAddr_{PSIZE - 1...3} || (pAddr_{2...0} xor (ReverseEndian || 0^{2}))$
          $byte \leftarrow vAddr_{2...0} xor (BigEndianCPU || 0^{2})$
          $data \leftarrow GPR [rt]_{63 - 8 * byte} || 0^{8 * byte}$
          StoreMemory (uncached, WORD, data, pAddr, vAddr, DATA)

64   T:   $vAddr \leftarrow ((offset_{15})^{48} || offset_{15...0}) + GPR [base]$
          $(pAddr, uncached) \leftarrow AddressTranslation (vAddr, DATA)$
          $pAddr \leftarrow pAddr_{PSIZE - 1...3} || (pAddr_{2...0} xor (ReverseEndian || 0^{2}))$
          $byte \leftarrow vAddr_{2...0} xor (BigEndianCPU || 0^{2})$
          $data \leftarrow GPR [rt]_{63 - 8 * byte} || 0^{8 * byte}$
          StoreMemory (uncached, WORD, data, pAddr, vAddr, DATA)
```

Exceptions:

TLB refill exception

TLB invalid exception

TLB modification exception

Bus error exception

Address error exception

SWL Store Word Left (1/3) SWL

31 26	25 21	20 16	15 0
SWL 1 0 1 0 1 0	base	rt	offset
6	5	5	16

Format:

SWL rt, offset (base)

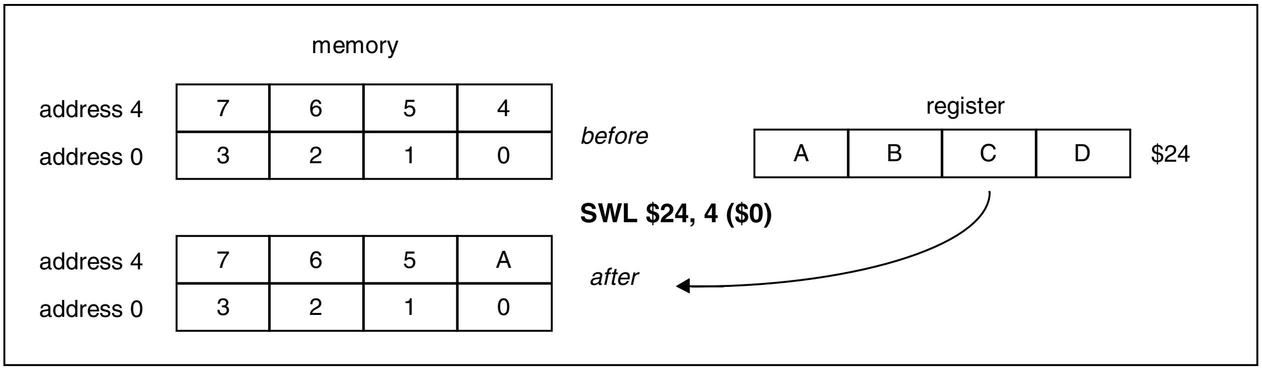
Description:

This instruction can be used with the SWR instruction to store the contents of a register into four consecutive bytes of memory, when the bytes cross a word boundary. SWL stores the register into the appropriate part of the high-order word of memory; SWR stores the register into the appropriate part of the low-order word.

The SWL instruction adds its sign-extended 16-bit *offset* to the contents of general register *base* to form a virtual address that may specify an arbitrary byte. It alters only the word in memory that contains that byte. From one to four bytes will be stored, depending on the starting byte specified.

Conceptually, it starts at the most-significant byte of the register and copies it to the specified byte in memory; then it copies bytes from register to memory until it reaches the low-order byte of the word in memory.

No address error exceptions due to alignment are possible.



SWL Store Word Left (2/3) SWL

Operation:

```
32   T:   vAddr ← ((offset$_{15}$)$^{16}$ || offset$_{15...0}$) + GPR [base]
          (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
          pAddr ← pAddr$_{PSIZE - 1...3}$ || (pAddr$_{2...0}$ xor ReverseEndian$^3$)
          if BigEndianMem = 0 then
               pAddr ← pAddr$_{PSIZE - 1...2}$ || 0$^2$
          endif
          byte ← vAddr$_{1...0}$ xor BigEndianCPU$^2$
          if (vAddr$_2$ xor BigEndianCPU) = 0 then
               data ← 0$^{32}$ || 0$^{24 - 8 * byte}$ || GPR [rt]$_{31...24 - 8 * byte}$
          else
               data ← 0$^{24 - 8 * byte}$ || GPR [rt]$_{31...24 - 8 * byte}$ || 0$^{32}$
          endif
          StoreMemory (uncached, byte, data, pAddr, vAddr, DATA)

64   T:   vAddr ← ((offset$_{15}$)$^{48}$ || offset$_{15...0}$) + GPR [base]
          (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
          pAddr ← pAddr$_{PSIZE - 1...3}$ || (pAddr$_{2...0}$ xor ReverseEndian$^3$)
          if BigEndianMem = 0 then
               pAddr ← pAddr$_{PSIZE - 1...2}$ || 0$^2$
          endif
          byte ← vAddr$_{1...0}$ xor BigEndianCPU$^2$
          if (vAddr2 xor BigEndianCPU) = 0 then
               data ← 0$^{32}$ || 0$^{24 - 8 * byte}$ || GPR [rt]$_{31...24 - 8 * byte}$
          else
               data ← 0$^{24 - 8 * byte}$ || GPR [rt]$_{31...24 - 8 * byte}$ || 0$^{32}$
          endif
          StoreMemory (uncached, byte, data, pAddr, vAddr, DATA)
```

SWL Store Word Left (3/3) SWL

Given a doubleword in a register and a doubleword in memory, the operation of SWL is as follows:

SWL

Register	A	B	C	D	E	F	G	H
Memory	I	J	K	L	M	N	O	P

vAddr2..0	Destination	Type	Offset (LEM)
0	IJKLMNOE	0	0
1	IJKLMNEF	1	0
2	IJKLMEFG	2	0
3	IJKLEFGH	3	0
4	IJKEMNOP	0	4
5	IJEFMNOP	1	4
6	IEFGMNOP	2	4
7	EFGHMNOP	3	4

Remark *LEM* Little-endian memory (BigEndianMem = 0)
Type AccessType (see **Table 2-3. Byte Specification Related to Load and Store Instructions**) sent to memory
Offset pAddr2..0 sent to memory

Exceptions:

TLB refill exception
TLB invalid exception
TLB modification exception
Bus error exception
Address error exception

SWR Store Word Right (1/3) SWR

31 26	25 21	20 16	15 0
SWR 1 0 1 1 1 0	base	rt	offset
6	5	5	16

Format:

SWR rt, offset (base)

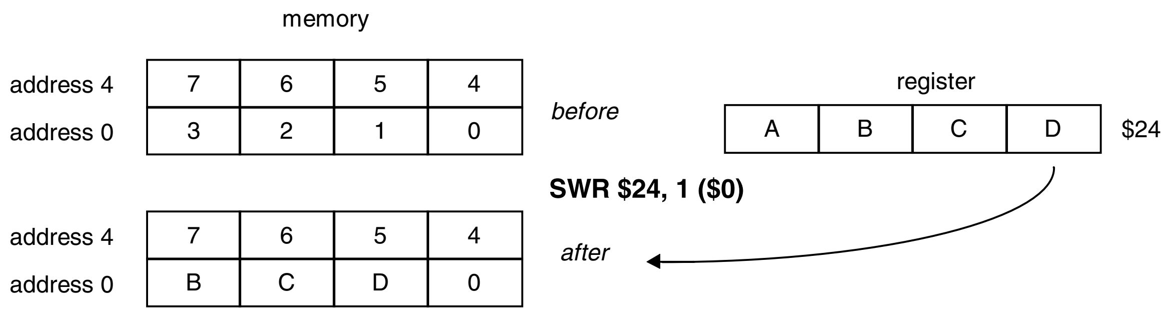
Description:

This instruction can be used with the SWL instruction to store the contents of a register into four consecutive bytes of memory, when the bytes cross a boundary between two words. SWR stores the register into the appropriate part of the low-order word; SWL stores the register into the appropriate part of the low-order word of memory.

The SWR instruction adds its sign-extended 16-bit *offset* to the contents of general register *base* to form a virtual address that may specify an arbitrary byte. It alters only the word in memory that contains that byte. From one to four bytes will be stored, depending on the starting byte specified.

Conceptually, it starts at the least-significant (rightmost) byte of the register and copies it to the specified byte in memory; then copies bytes from register to memory until it reaches the high-order byte of the word in memory.

No address error exceptions due to alignment are possible.



SWR Store Word Right (2/3) SWR

Operation:

```
32   T:   $vAddr \leftarrow ((offset_{15})^{16} \| offset_{15...0}) + GPR\ [base]$
          $(pAddr, uncached) \leftarrow AddressTranslation\ (vAddr, DATA)$
          $pAddr \leftarrow pAddr_{PSIZE - 1...3} \| (pAddr_{2...0}\ xor\ ReverseEndian^{3})$
          if BigEndianMem = 1 then
                $pAddr \leftarrow pAddr_{PSIZE - 1...2} \| 0^{2}$
          endif
          $byte \leftarrow vAddr_{1...0}\ xor\ BigEndianCPU^{2}$
          if $(vAddr_{2}\ xor\ BigEndianCPU) = 0$ then
                $data \leftarrow 0^{32} \| GPR\ [rt]_{31 - 8 * byte...0} \| 0^{8 * byte}$
          else
                $data \leftarrow GPR\ [rt]_{31 - 8 * byte} \| 0^{8 * byte} \| 0^{32}$
          endif
          StoreMemory (uncached, WORD-byte, data, pAddr, vAddr, DATA)

64   T:   $vAddr \leftarrow ((offset_{15})^{48} \| offset_{15...0}) + GPR\ [base]$
          $(pAddr, uncached) \leftarrow AddressTranslation\ (vAddr, DATA)$
          $pAddr \leftarrow pAddr_{PSIZE - 1...3} \| (pAddr_{2...0}\ xor\ ReverseEndian^{3})$
          if BigEndianMem = 1 then
                $pAddr \leftarrow pAddr_{PSIZE - 1...2} \| 0^{2}$
          endif
          $byte \leftarrow vAddr_{1...0}\ xor\ BigEndianCPU^{2}$
          if $(vAddr_{2}\ xor\ BigEndianCPU) = 0$ then
                $data \leftarrow 0^{32} \| GPR\ [rt]_{31 - 8 * byte...0} \| 0^{8 * byte}$
          else
                $data \leftarrow GPR\ [rt]_{31 - 8 * byte} \| 0^{8 * byte} \| 0^{32}$
          endif
          StoreMemory (uncached, WORD-byte, data, pAddr, vAddr, DATA)
```

SWR Store Word Right (3/3) SWR

Given a doubleword in a register and a doubleword in memory, the operation of SWR instruction is as follows:

SWR

Register	A	B	C	D	E	F	G	H
Memory	I	J	K	L	M	N	O	P

vAddr2.. 0	Destination	Type	Offset (LEM)
0	I J K L E F G H	3	0
1	I J K L F G H P	2	1
2	I J K L G H O P	1	2
3	I J K L H N O P	0	3
4	E F G H M N O P	3	4
5	F G H L M N O P	2	5
6	G H K L M N O P	1	6
7	H J K L M N O P	0	7

Remark *LEM* Little-endian memory (BigEndianMem = 0)

Type AccessType (see **Table 2-3. Byte Specification Related to Load and Store Instructions**) sent to memory

Offset pAddr2..0 sent to memory

Exceptions:

TLB refill exception

TLB invalid exception

TLB modification exception

Bus error exception

Address error exception

SYNC Synchronize SYNC

31 26	25 6	5 0
SPECIAL 000000	0 0000 0000 0000 0000 0000	SYNC 001111
6	20	6

Format:

SYNC

Description:

The SYNC instruction is executed as a NOP on the VR4121. This operation maintains compatibility with code compiled for the VR4000.

This instruction is defined to maintain the compatibility with VR4000 and VR4400.

Operation:

```
32, 64  T:      SyncOperation ( )
```

Exceptions:

None

SYSCALL System Call SYSCALL

31 26	25 6	5 0
SPECIAL 0 0 0 0 0 0	Code	SYSCALL 0 0 1 1 0 0
6	20	6

Format:

SYSCALL

Description:

A system call exception occurs, immediately and unconditionally transferring control to the exception handler.

The code field is available for use as software parameters, but is retrieved by the exception handler only by loading the contents of the memory word containing the instruction.

Operation:

```
32, 64  T:    SystemCallException
```

Exceptions:

System Call exception

TEQ Trap If Equal TEQ

31 26	25 21	20 16	15 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	code	TEQ 1 1 0 1 0 0
6	5	5	10	6

Format:

TEQ rs, rt

Description:

The contents of general register *rt* are compared to general register *rs*. If the contents of general register *rs* are equal to the contents of general register *rt*, a trap exception occurs.

The code field is available for use as software parameters, but is retrieved by the exception handler only by loading the contents of the memory word containing the instruction.

Operation:

```
32, 64  T:    if GPR [rs] = GPR [rt] then
                      TrapException
              endif
```

Exceptions:

Trap exception

TEQI Trap If Equal Immediate TEQI

31 … 26	25 … 21	20 … 16	15 … 0
REGIMM 0 0 0 0 0 1	rs	TEQI 0 1 1 0 0	immediate
6	5	5	16

Format:

TEQI rs, immediate

Description:

The 16-bit *immediate* is sign-extended and compared to the contents of general register *rs*. If the contents of general register *rs* are equal to the sign-extended *immediate*, a trap exception occurs.

Operation:

```
32    T:    if GPR [rs] = $(\text{immediate}_{15})^{16} \,||\, \text{immediate}_{15...0}$ then
                TrapException
            endif

64    T:    if GPR [rs] = $(\text{immediate}_{15})^{48} \,||\, \text{immediate}_{15...0}$ then
                TrapException
            endif
```

Exceptions:

Trap exception

TGE Trap If Greater Than Or Equal TGE

31 26	25 21	20 16	15 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	code	TGE 1 1 0 0 0 0
6	5	5	10	6

Format:

TGE rs, rt

Description:

The contents of general register *rt* are compared to the contents of general register *rs*. Considering both quantities as signed integers, if the contents of general register *rs* are greater than or equal to the contents of general register *rt*, a trap exception occurs.

The code field is available for use as software parameters, but is retrieved by the exception handler only by loading the contents of the memory word containing the instruction.

Operation:

```
32, 64  T:    if GPR [rs] ≥ GPR [rt] then
                   TrapException
              endif
```

Exceptions:

Trap exception

TGEI Trap If Greater Than Or Equal Immediate TGEI

31 26	25 21	20 16	15 0
REGIMM 0 0 0 0 0 1	rs	TGEI 0 1 0 0 0	immediate
6	5	5	16

Format:

TGEI rs, immediate

Description:

The 16-bit *immediate* is sign-extended and compared to the contents of general register *rs*. Considering both quantities as signed integers, if the contents of general register *rs* are greater than or equal to the sign-extended *immediate*, a trap exception occurs.

Operation:

```
32    T:    if GPR [rs] $\geq$ $(\text{immediate}_{15})^{16}$ || $\text{immediate}_{15...0}$ then
                  TrapException
            endif

64    T:    if GPR [rs] $\geq$ $(\text{immediate}_{15})^{48}$ || $\text{immediate}_{15...0}$ then
                  TrapException
            endif
```

Exceptions:

Trap exception

TGEIU Trap If Greater Than Or Equal Immediate Unsigned TGEIU

31 26	25 21	20 16	15 0
REGIMM 000001	rs	TGEIU 01001	immediate
6	5	5	16

Format:

TGEIU rs, immediate

Description:

The 16-bit *immediate* is sign-extended and compared to the contents of general register *rs*. Considering both quantities as unsigned integers, if the contents of general register *rs* are greater than or equal to the sign-extended *immediate*, a trap exception occurs.

Operation:

```
32   T:   if $(0 \| GPR[rs]) \geq (0 \| (immediate_{15})^{16} \| immediate_{15...0})$ then
               TrapException
          endif

64   T:   if $(0 \| GPR[rs]) \geq (0 \| (immediate_{15})^{48} \| immediate_{15...0})$ then
               TrapException
          endif
```

Exceptions:

Trap exception

TGEU Trap If Greater Than Or Equal Unsigned TGEU

31 26	25 21	20 16	15 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	code	TGEU 1 1 0 0 0 1
6	5	5	10	6

Format:

TGEU rs, rt

Description:

The contents of general register *rt* are compared to the contents of general register *rs*. Considering both quantities as unsigned integers, if the contents of general register *rs* are greater than or equal to the contents of general register *rt*, a trap exception occurs.

The code field is available for use as software parameters, but is retrieved by the exception handler only by loading the contents of the memory word containing the instruction.

Operation:

```
32, 64  T:    if (0 || GPR [rs]) ≥ (0 || GPR [rt]) then
                      TrapException
              endif
```

Exceptions:

Trap exception

TLBP Probe TLB For Matching Entry TLBP

31 26	25	24 6	5 0
COP0 0 1 0 0 0 0	CO 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	TLBP 0 0 1 0 0 0
6	1	19	6

Format:

TLBP

Description:

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.

The architecture does not specify the operation of memory references associated with the instruction immediately after a TLBP instruction, nor is the operation specified if more than one TLB entry matches.

Operation:

```
32   T:   Index ← 1 || $0^{25}$ || $\text{Undefined}^{6}$
          for i in 0...TLBEntries - 1
              if (TLB [i]$_{95...77}$ = EntryHi$_{31...13}$) and (TLB [i]$_{76}$ or
              (TLB [i]$_{71...64}$ = EntryHi$_{7...0}$)) then
                  Index ← $0^{26}$ || i$_{5...0}$
              endif
          endfor

64   T:   Index ← 1 || $0^{25}$ || $\text{Undefined}^{6}$
          for i in 0...TLBEntries - 1
              if (TLB [i]$_{167...141}$ and not ($0^{15}$ || TLB [i]$_{216...205}$))
              = (EntryHi$_{39...13}$) and not ($0^{15}$ || TLB [i]$_{216...205}$)) and
              (TLB [i]$_{140}$ or (TLB [i]$_{135...128}$ = EntryHi$_{7...0}$)) then
                  Index ← $0^{26}$ || i$_{5...0}$
              endif
          endfor
```

Exceptions:

Coprocessor unusable exception

TLBR Read Indexed TLB Entry TLBR

31 26	25	24 6	5 0
COP0 0 1 0 0 0 0	CO 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	TLBR 0 0 0 0 0 1
6	1	19	6

Format:

TLBR

Description:

The EntryHi and EntryLo registers are loaded with the contents of the TLB entry pointed at by the contents of the TLB Index register.

The *G* bit (which controls ASID matching) read from the TLB is written into both of the EntryLo0 and EntryLo1 registers. The operation is invalid (and the results are unspecified) if the contents of the TLB Index register are greater than the number of TLB entries in the processor.

Operation:

```
32   T:   $\text{PageMask} \leftarrow \text{TLB}\ [\text{Index}_{5...0}]_{127...96}$
          $\text{EntryHi} \leftarrow \text{TLB}\ [\text{Index}_{5...0}]_{95...64}$ and not $\text{TLB}\ [\text{Index}_{5...0}]_{127...96}$
          $\text{EntryLo1} \leftarrow \text{TLB}\ [\text{Index}_{5...0}]_{63...33} \,||\, \text{TLB}\ [\text{Index}_{5...0}]_{76}$
          $\text{EntryLo0} \leftarrow \text{TLB}\ [\text{Index}_{5...0}]_{31...1} \,||\, \text{TLB}\ [\text{Index}_{5...0}]_{76}$

64   T:   $\text{PageMask} \leftarrow \text{TLB}\ [\text{Index}_{5...0}]_{255...192}$
          $\text{EntryHi} \leftarrow \text{TLB}\ [\text{Index}_{5...0}]_{191...128}$ and not $\text{TLB}\ [\text{Index}_{5...0}]_{255...192}$
          $\text{EntryLo1} \leftarrow \text{TLB}\ [\text{Index}_{5...0}]_{127...65} \,||\, \text{TLB}\ [\text{Index}_{5...0}]_{140}$
          $\text{EntryLo0} \leftarrow \text{TLB}\ [\text{Index}_{5...0}]_{63...1} \,||\, \text{TLB}\ [\text{Index}_{5...0}]_{140}$
```

Exceptions:

Coprocessor unusable exception

TLBWI Write Indexed TLB Entry TLBWI

31 26	25	24 6	5 0
COP0 0 1 0 0 0 0	CO 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	TLBWI 0 0 0 0 1 0
6	1	19	6

Format:

TLBWI

Description:

The TLB entry pointed at by the contents of the TLB Index register is loaded with the contents of the EntryHi and EntryLo registers.

The *G* bit of the TLB is written with the logical AND of the *G* bits in the EntryLo0 and EntryLo1 registers.

The operation is invalid (and the results are unspecified) if the contents of the TLB Index register are greater than the number of TLB entries in the processor.

Operation:

```
32, 64  T:    TLB [Index_5...0] ←
                    PageMask || (EntryHi and not PageMask) || EntryLo1 || EntryLo0
```

Exceptions:

Coprocessor unusable exception

TLBWR Write Random TLB Entry TLBWR

31 26	25	24 6	5 0
COP0 0 1 0 0 0 0	CO 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	TLBWR 0 0 0 1 1 0
6	1	19	6

Format:

TLBWR

Description:

The TLB entry pointed at by the contents of the TLB Random register is loaded with the contents of the EntryHi and EntryLo registers.

The *G* bit of the TLB is written with the logical AND of the *G* bits in the EntryLo0 and EntryLo1 registers.

Operation:

```
32, 64  T:   TLB [$\text{Random}_{5...0}$] ←
                   PageMask || (EntryHi and not PageMask) || EntryLo1 || EntryLo0
```

Exceptions:

Coprocessor unusable exception

TLT Trap If Less Than TLT

31 26	25 21	20 16	15 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	code	TLT 1 1 0 0 1 0
6	5	5	10	6

Format:

TLT rs, rt

Description:

The contents of general register *rt* are compared to general register *rs*. Considering both quantities as signed integers, if the contents of general register *rs* are less than the contents of general register *rt*, a trap exception occurs.

The code field is available for use as software parameters, but is retrieved by the exception handler only by loading the contents of the memory word containing the instruction.

Operation:

```
32, 64  T:    if GPR [rs] < GPR [rt] then
                      TrapException
              endif
```

Exceptions:

Trap exception

TLTI Trap If Less Than Immediate TLTI

31 26	25 21	20 16	15 0
REGIMM 0 0 0 0 0 1	rs	TLTI 0 1 0 1 0	immediate
6	5	5	16

Format:

TLTI rs, immediate

Description:

The 16-bit *immediate* is sign-extended and compared to the contents of general register *rs*. Considering both quantities as signed integers, if the contents of general register *rs* are less than the sign-extended *immediate*, a trap exception occurs.

Operation:

```
32    T:    if GPR [rs] < $(\text{immediate}_{15})^{16}$ || $\text{immediate}_{15...0}$ then
                TrapException
            endif

64    T:    if GPR [rs] < $(\text{immediate}_{15})^{48}$ || $\text{immediate}_{15...0}$ then
                TrapException
            endif
```

Exceptions:

Trap exception

TLTIU Trap If Less Than Immediate Unsigned TLTIU

31 26	25 21	20 16	15 0
REGIMM 0 0 0 0 0 1	rs	TLTIU 0 1 0 1 1	immediate
6	5	5	16

Format:

TLTIU rs, immediate

Description:

The 16-bit *immediate* is sign-extended and compared to the contents of general register *rs*. Considering both quantities as unsigned integers, if the contents of general register *rs* are less than the sign-extended *immediate*, a trap exception occurs.

Operation:

```
32   T:   if $(0 \| GPR[rs]) < (0 \| (immediate_{15})^{16} \| immediate_{15...0})$ then
               TrapException
          endif

64   T:   if $(0 \| GPR[rs]) < (0 \| (immediate_{15})^{48} \| immediate_{15...0})$ then
               TrapException
          endif
```

Exceptions:

Trap exception

TLTU Trap If Less Than Unsigned TLTU

31 26	25 21	20 16	15 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	code	TLTU 1 1 0 0 1 1
6	5	5	10	6

Format:

TLTU rs, rt

Description:

The contents of general register *rt* are compared to general register *rs*. Considering both quantities as unsigned integers, if the contents of general register *rs* are less than the contents of general register *rt*, a trap exception occurs.

The code field is available for use as software parameters, but is retrieved by the exception handler only by loading the contents of the memory word containing the instruction.

Operation:

```
32, 64  T:    if (0 || GPR [rs]) < (0 || GPR [rt]) then
                      TrapException
              endif
```

Exceptions:

Trap exception

TNE Trap If Not Equal TNE

31 26	25 21	20 16	15 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	code	TNE 1 1 0 1 1 0
6	5	5	10	6

Format:

TNE rs, rt

Description:

The contents of general register *rt* are compared to general register *rs*. If the contents of general register *rs* are not equal to the contents of general register *rt*, a trap exception occurs.

The code field is available for use as software parameters, but is retrieved by the exception handler only by loading the contents of the memory word containing the instruction.

Operation:

```
32, 64  T:   if GPR [rs] ≠ GPR [rt] then
                  TrapException
             endif
```

Exceptions:

Trap exception

TNEI Trap If Not Equal Immediate TNEI

31 26	25 21	20 16	15 0
REGIMM 0 0 0 0 0 1	rs	TNEI 0 1 1 1 0	immediate
6	5	5	16

Format:

TNEI rs, immediate

Description:

The 16-bit *immediate* is sign-extended and compared to the contents of general register *rs*. If the contents of general register *rs* are not equal to the sign-extended *immediate*, a trap exception occurs.

Operation:

```
32   T:   if $\text{GPR [rs]} \neq (\text{immediate}_{15})^{16} \,||\, \text{immediate}_{15...0}$ then
                TrapException
          endif

64   T:   if $\text{GPR [rs]} \neq (\text{immediate}_{15})^{48} \,||\, \text{immediate}_{15...0}$ then
                TrapException
          endif
```

Exceptions:

Trap exception

XOR Exclusive Or XOR

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	rs	rt	rd	0 0 0 0 0 0	XOR 1 0 0 1 1 0
6	5	5	5	5	6

Format:

XOR rd, rs, rt

Description:

The contents of general register *rs* are combined with the contents of general register *rt* in a bit-wise logical exclusive OR operation.

The result is placed into general register *rd.*

Operation:

```
32, 64  T:    GPR [rd] ← GPR [rs] xor GPR [rt]
```

Exceptions:

None

XORI Exclusive OR Immediate XORI

31 26	25 21	20 16	15 0
XORI 0 0 1 1 1 0	rs	rt	immediate
6	5	5	16

Format:

XORI rt, rs, immediate

Description:

The 16-bit *immediate* is zero-extended and combined with the contents of general register *rs* in a bit-wise logical exclusive OR operation.

The result is placed into general register *rt.*

Operation:

32	T:	GPR [rt] ← GPR [rs] xor (0^{16} || immediate)
64	T:	GPR [rt] ← GPR [rs] xor (0^{48} || immediate)

Exceptions:

None

A.6 CPU Instruction Opcode Bit Encoding

Figure A-1 lists the VR4120A Opcode Bit Encoding.

Figure A-1. VR4120A Opcode Bit Encoding (1/2)

28...26 **Opcode**

31...29	0	1	2	3	4	5	6	7
0	SPECIAL	REGIMM	J	JAL	BEQ	BNE	BLEZ	BGTZ
1	ADDI	ADDIU	SLTI	SLTIU	ANDI	ORI	XORI	LUI
2	COP0	π	π	*	BEQL	BNEL	BLEZL	BGTZL
3	DADDIε	DADDIUε	LDLε	LDRε	*	JALXθ	*	*
4	LB	LH	LWL	LW	LBU	LHU	LWR	LWUε
5	SB	SH	SWL	SW	SDLε	SDRε	SWR	CACHEδ
6	*	π	π	*	*	π	π	LDε
7	*	π	π	*	*	π	π	SDε

2...0 **SPECIAL function**

5...3	0	1	2	3	4	5	6	7
0	SLL	*	SRL	SRA	SLLV	*	SRLV	SRAV
1	JR	JALR	*	*	SYSCALL	BREAK	*	SYNC
2	MFHI	MTHI	MFLO	MTLO	DSLLVε	*	DSRLVε	DSRAVε
3	MULT	MULTU	DIV	DIVU	DMULTε	DMULTUε	DDIVε	DDIVUε
4	ADD	ADDU	SUB	SUBU	AND	OR	XOR	NOR
5	MACC	DMACC	SLT	SLTU	DADDε	DADDUε	DSUBε	DSUBUε
6	TGE	TGEU	TLT	TLTU	TEQ	*	TNE	*
7	DSLLε	*	DSRLε	DSRAε	DSLL32ε	*	DSRL32ε	DSRA32ε

18...16 **REGIMM rt**

20...19	0	1	2	3	4	5	6	7
0	BLTZ	BGEZ	BLTZL	BGEZL	*	*	*	*
1	TGEI	TGEIU	TLTI	TLTIU	TEQI	*	TNEI	*
2	BLTZAL	BGEZAL	BLTZALL	BGEZALL	*	*	*	*
3	*	*	*	*	*	*	*	*

Figure A-1. VR4120AOpcode Bit Encoding (2/2)

23...21 **COP0 rs**

25, 24	0	1	2	3	4	5	6	7
0	MF	DMFε	γ	γ	MT	DMTε	γ	γ
1	BC	γ	γ	γ	γ	γ	γ	γ
2	CO							
3								

18...16 **COP0 rt**

20...19	0	1	2	3	4	5	6	7
0	BCF	BCT	BCFL	BCTL	γ	γ	γ	γ
1	γ	γ	γ	γ	γ	γ	γ	γ
2	γ	γ	γ	γ	γ	γ	γ	γ
3	γ	γ	γ	γ	γ	γ	γ	γ

2...0 **CP0 Function**

5...3	0	1	2	3	4	5	6	7
0	φ	TLBR	TLBWI	φ	φ	φ	TLBWR	φ
1	TLBP	φ	φ	φ	φ	φ	φ	φ
2	ξ	φ	φ	φ	φ	φ	φ	φ
3	ERET χ	φ	φ	φ	φ	φ	φ	φ
4	φ	STANDBY	SUSPEND	HIBERNAT	φ	φ	φ	φ
5	φ	φ	φ	φ	φ	φ	φ	φ
6	φ	φ	φ	φ	φ	φ	φ	φ
7	φ	φ	φ	φ	φ	φ	φ	φ

Key:

*	Operation codes marked with an asterisk cause reserved instruction exceptions in all current implementations and are reserved for future versions of the architecture.
γ	Operation codes marked with a gamma cause a reserved instruction exception. They are reserved for future versions of the architecture.
δ	Operation codes marked with a delta are valid only for VR4400 Series processors with CP0 enabled, and cause a reserved instruction exception on other processors.
φ	Operation codes marked with a phi are invalid but do not cause reserved instruction exceptions in VR4121 implementations.
ξ	Operation codes marked with a xi cause a reserved instruction exception on VR4121 processor.
χ	Operation codes marked with a chi are valid on VR4000 Series only.
ε	Operation codes marked with epsilon are valid when the processor operating as a 64-bit processor. These instructions will cause a reserved instruction exception if 64-bit operation is not enabled.
π	Operation codes marked with a pi are invalid and cause coprocessor unusable exception.
θ	Operation codes marked with a theta are valid when MIPS16 instruction execution is enabled, and cause a reserved instruction exception when MIPS16 instruction execution is disabled.

APPENDIX B VR4120A COPROCESSOR 0 HAZARDS

The VR4120A core avoids contention of its internal resources by causing a pipeline interlock in such cases as when the contents of the destination register of an instruction are used as a source in the succeeding instruction. Therefore, instructions such as NOP must not be inserted between instructions.

However, interlocks do not occur on the operations related to the CP0 registers and the TLB. Therefore, contention of internal resources should be considered when composing a program that manipulates the CP0 registers or the TLB. The CP0 hazards define the number of NOP instructions that is required to avoid contention of internal resources, or the number of instructions unrelated to contention. This chapter describes the CP0 hazards.

The CP0 hazards of the VR4120A core are as or less stringent than those of the VR4000. Table B-1 lists the Coprocessor 0 hazards of the VR4120A core. Code that complies with these hazards will run without modification on the VR4000.

The contents of the CP0 registers or the bits in the "Source" column of this table can be used as a source after they are fixed.

The contents of the CP0 registers or the bits in the "Destination" column of this table can be available as a destination after they are stored.

Based on this table, the number of NOP instructions required between instructions related to the TLB is computed by the following formula, and so is the number of instructions unrelated to contention:

(Destination Hazard number of A) – [(Source Hazard number of B) + 1]

As an example, to compute the number of instructions required between an MTC0 and a subsequent MFC0 instruction, this is:

(5) – (3 + 1) = 1 instruction

The CP0 hazards do not generate interlocks of pipeline. Therefore, the required number of instruction must be controlled by program.

Table B-1. VR4120A CPU Coprocessor 0 Hazards

Operation	Source		Destination	
	Source Name	No. of Cycles	Destination Name	No. of Cycles
MTC0			cpr rd	5
MFC0	cpr rd	3		
TLBR	Index, TLB	2	PageMask, EntryHi, EntryLo0, EntryLo1	5
TLBWI TLBWR	Index or Random, PageMask, EntryHi, EntryLo0, EntryLo1	2	TLB	5
TLBP	PageMask, EntryHi	2	Index	6
ERET	EPC or ErrorEPC, TLB	2	Status.EXL, Status.ERL	4
	Status	2		
CACHE Index Load Tag			TagLo, TagHi, PErr	5
CACHE Index Store Tag	TagLo, TagHi, PErr	3		
CACHE Hit ops.	cache line	3	cache line	5
Coprocessor usable test	Status.CU, Status.KSU, Status.EXL, Status.ERL	2		
Instruction fetch	EntryHi.ASID, Status.KSU, Status.EXL, Status.ERL, Status.RE, Config.K0C	2		
	TLB	2		
Instruction fetch exception			EPC, Status	4
			Cause, BadVAddr, Context, XContext	5
Interrupt signals	Cause.IP, Status.IM, Status.IE, Status.EXL, Status.ERL	2		
Load/Store	EntryHi.ASID, Status.KSU, Status.EXL, Status.ERL, Status.RE, Config.K0C, TLB	3		
	Config.AD, Config.EP	3		
	WatchHi, WatchLo	3		
Load/Store exception			EPC, Status, Cause, BadVAddr, Context, XContext	5
TLB shutdown			Status.TS	2 (Inst.), 4 (Data)

Cautions 1. If the setting of the K0 bit in the Config register is changed to uncached mode by MTC0, the accessed memory area is switched to the uncached one at the instruction fetch of the third instruction after MTC0.

2. A stall of several instructions occurs if a jump or branch instruction is executed immediately after the setting of the ITS bit in the Status register.

Remarks 1. The instruction following MTC0 must not be MFC0.

2. The five instructions following MTC0 to Status register that changes KSU and sets EXL and ERL may be executed in the new mode, and not kernel mode. This can be avoided by setting EXL first, leaving KSU set to kernel, and later changing KSU.

3. There must be two non-load, non-CACHE instructions between a store and a CACHE instruction directed to the same primary cache line as the store.

The status during execution of the following instruction for which CP0 hazards must be considered is described below.

(1) MTC0

Destination: The completion of writing to a destination register (CP0) of MTC0.

(2) MFC0

Source: The confirmation of a source register (CP0) of MFC0.

(3) TLBR

Source: The confirmation of the status of TLB and the Index register before the execution of TLBR.

Destination: The completion of writing to a destination register (CP0) of TLBR.

(4) TLBWI, TLBWR

Source: The confirmation of a source register of these instructions and registers used to specify a TLB entry.

Destination: The completion of writing to TLB by these instructions.

(5) TLBP

Source: The confirmation of the PageMask register and the EntryHi register before the execution of TLBP.

Destination: The completion of writing the result of execution of TLBP to the Index register.

(6) ERET

Source: The confirmation of registers containing information necessary for executing ERET.

Destination: The completion of the processor state transition by the execution of ERET.

(7) CACHE Index Load Tag

Destination: The completion of writing the results of execution of this instruction to the related registers.

(8) CACHE Index Store Tag

Source: The confirmation of registers containing information necessary for executing this instruction.

(9) Coprocessor Usable Test

Source: The confirmation of modes set by the bits of the CP0 registers in the "Source" column.

Examples 1. When accessing the CP0 registers in User mode after the contents of the CU0 bit of the Status register are modified, or when executing an instruction such as TLB instructions, CACHE instructions, or branch instructions that use the resource of the CP0.

2. When accessing the CP0 registers in the operating mode set in the Status register after the KSU, EXL, and ERL bits of the Status register are modified.

(10) Instruction Fetch

Source: The confirmation of the operating mode and TLB necessary for instruction fetch.

Examples 1. When changing the operating mode from User to Kernel and fetching instructions after the KSU, EXL, and ERL bits of the Status register are modified.

2. When fetching instructions using the modified TLB entry after TLB modification.

(11) Instruction Fetch Exception

Destination: The completion of writing to registers containing information related to the exception when an exception occurs on instruction fetch.

(12) Interrupts

Source: The confirmation of registers judging the condition of occurrence of interrupt when an interrupt factor is detected.

(13) Loads/Sores

Source: The confirmation of the operating mode related to the address generation of Load/Store instructions, TLB entries, the cache mode set in the K0 bit of the Config register, and the registers setting the condition of occurrence of a Watch exception.

Example When Loads/Stores are executed in the kernel field after changing the mode from User to Kernel.

(14) Load/Store Exception

Destination: The completion of writing to registers containing information related to the exception when an exception occurs on load or store operation.

(15) TLB Shutdown

Destination: The completion of writing to the TS bit of the Status register when a TLB shutdown occurs.

Table B-2 indicates examples of calculation.

Table B-2. Calculation Example of CP0 Hazard and Number of Instructions Inserted

Destination	Source	Contending Internal Resource	Number of Instructions Inserted	Formula
TLBWR/TLBWI	TLBP	TLB Entry	2	5 – (2 + 1)
TLBWR/TLBWI	Load or Store using newly modified TLB	TLB Entry	1	5 – (3 + 1)
TLBWR/TLBWI	Instruction fetch using newly modified TLB	TLB Entry	2	5 – (2 + 1)
MTC0, Status [CU]	Coprocessor instruction that requires the setting of CU	Status [CU]	2	5 – (2 + 1)
TLBR	MFC0 EntryHi	EntryHi	1	5 – (3 + 1)
MTC0 EntryLo0	TLBWR/TLBWI	EntryLo0	2	5 – (2 + 1)
TLBP	MFC0 Index	Index	2	6 – (3 + 1)
MTC0 EntryHi	TLBP	EntryHi	2	5 – (2 + 1)
MTC0 EPC	ERET	EPC	2	5 – (2 + 1)
MTC0 Status	ERET	Status	2	5 – (2 + 1)
MTC0 Status [IE] Note	Instruction that causes an interrupt	Status [IE]	2	5 – (2 + 1)

Note The number of hazards is undefined if the instruction execution sequence is changed by exceptions. In such a case, the minimum number of hazards until the IE bit value is confirmed may be the same as the maximum number of hazards until an interrupt request occurs that is pending and enabled.

NEC

Facsimile Message

From:

Name

Company

Tel.   FAX

Address

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

Thank you for your kind support.

North America
NEC Electronics Inc.
Corporate Communications Dept.
Fax: +1-800-729-9288
+1-408-588-6130

Europe
NEC Electronics (Europe) GmbH
Market Communication Dept.
Fax: +49-211-6503-274

South America
NEC do Brasil S.A.
Fax: +55-11-6462-6829

Hong Kong, Philippines, Oceania
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

Korea
NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: +82-2-528-4411

Taiwan
NEC Electronics Taiwan Ltd.
Fax: +886-2-2719-5951

Asian Nations except Philippines
NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Japan
NEC Semiconductor Technical Hotline
Fax: +81- 44-435-9608

I would like to report the following error/make the following suggestion:

Document title: ____________________

Document number: ____________________ Page number: __________

If possible, please fax the referenced page or drawing.

Document Rating	Excellent	Good	Acceptable	Poor
Clarity	❑	❑	❑	❑
Technical Accuracy	❑	❑	❑	❑
Organization	❑	❑	❑	❑

CS 01.11