Preliminary User's Manual

VR4121™

64/32-bit Microprocessor

μPD30121

Document No. U13569EJ2V0UM00 (2nd edition)
Date Published June 1999 N CP(K)

[MEMO]

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.

Exporting this product or equipment that include this product may require a governmental license from the U.S.A. for some countries because this product utilizes technologies limited by the export control regulations of the U.S.A.

- **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.

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC product in your application, please contact the NEC office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.

NEC Electronics Inc. (U.S.)
Santa Clara, California
Tel: 408-588-6000
800-366-9782
Fax: 408-588-6130
800-729-9288

NEC Electronics (Germany) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 02
Fax: 0211-65 03 490

NEC Electronics (UK) Ltd.
Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics Italiana s.r.l.
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

NEC Electronics (Germany) GmbH
Benelux Office
Eindhoven, The Netherlands
Tel: 040-2445845
Fax: 040-2444580

NEC Electronics (France) S.A.
Velizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

NEC Electronics (France) S.A.
Spain Office
Madrid, Spain
Tel: 91-504-2787
Fax: 91-504-2860

NEC Electronics (Germany) GmbH
Scandinavia Office
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics Singapore Pte. Ltd.
United Square, Singapore 1130
Tel: 65-253-8311
Fax: 65-250-3583

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-2719-2377
Fax: 02-2719-5951

NEC do Brasil S.A.
Electron Devices Division
Rodovia Presidente Dutra, Km 214
07210-902-Guarulhos-SP Brasil
Tel: 55-11-6465-6810
Fax: 55-11-6465-6829

J99.1

Major Revisions in This Edition (1/2)

Page	Description
p.38	Modification of description in **1.1 Features**
p.38	Modification of description in **1.2 Ordering Information**
p.41	Addition of registers in **Table 1-1 BCU Registers**
p.69	Addition of Caution in **Figure 2-1 VR4121 Signal Classification**
pp.70, 72	Addition of description in **Table 2-1 System Bus Interface Signals**
p.78	Modification of description in **Table 2-14 Initial Setting Signals**
pp.81, 82	Addition of description in **Table 2-16 Pin Status upon Specific States**
p.99	Modification of description in **Table 3-11 Multiply/Divide Instructions (Extended ISA)**
p.141	Addition of description in **5.2 Branch Delay**
p.143	Deletion of figure in **5.4 (1) Add instruction (ADD rd, rs, rt)**
p.145	Deletion of figure in **5.4 (3) Branch on Equal instruction (BEQ rs, rt, offset)**
p.150	Modification of description in **Table 5-2 Correspondence of Pipeline Stage to Interlock and Exception Conditions**
p.156	Addition of description in **5.6 Program Compatibility**
p.176	Modification in **Table 6-10 Example of ROM Addresses when Using 128-Mbit Expansion ROM (32-bit Data Bus)**
p.177	Addition of description in **6.3.2 System bus address space**
p.178	Modification of description in **Table 6-12 Internal I/O Space 2**
p.179	Addition of description in **6.3.4 LCD Space**
p.180	Modification of description in **6.3.5 (2) (b) During 32-bit bus mode (DBUS32 = 1)**
p.184	Addition of description in **6.5.1 Index register (0)**
p.185	Addition of description in **6.5.3 EntryLo0(2) and Entry Lo1(3) registers**
p.186	Addition of description in **6.5.4 PageMask register (5)**
p.188	Addition of description in **6.5.6 EntryHi register (10)**
p.189	Addition of description in **6.5.8 Config register (16)**
p.191	Addition of description in **6.5.9 Load linked address (LLAddr) register (17)**
p.191	Addition of description in **6.5.10 Cache tag resisters (TagLo (28) and TagHi (29))**
p.200	Addition of description in **7.3.4 Compare register (11)**
p.206	Addition of description in **7.3.8 WatchLo (18) and WatchHi (19) registers**
p.213	Modification of description in **Table 7-5 Exception Priority Order**
p.237	Addition to and modification of description in **8.1.1 RTC reset**
p.239	Addition of description in **8.1.1 (1) When using EDO-type DRAM**
p.240	Addition of description in **8.1.1 (2) When using SDRAM**
pp.242, 243	Addition to and modification of **8.1.4 Software shutdown**
p.243	Modification of **Figure 8-6 Software Shutdown**
p.244	Modification of description in **8.1.5 HALTimer shutdown**
p.245	Modification of description in **8.2 Power-on Sequence**
p.250	Modification of description in **8.4.1 (4) Hibernate mode**
p.272	Modification of description in **10.2 Ordinary Interrupts**, **10.3 Software Interrupts Generated in CPU Core**, **10.4 Timer Interrupt**, and **10.5.1 Detecting hardware interrupts**
p.273	Modification of **Figure 10-2 Hardware Interrupt Signals**
p.275	Addition of registers in **Table 11-1 BCU Registers**

Major Revisions in This Edition (2/2)

Page	Description
pp.276 to 278	Addition of description in **11.2.1 BCUCNTREG1 (0x0B00 0000)**
pp.280, 282	Addition to and modification of description in **11.2.3 ROMSIZEREG (0x0B00 0004)**
pp.283, 284	Addition to and modification of description in **11.2.4 RAMSIZEREG (0x0B00 0006)**
pp.285, 286	Addition to and modification of description in **11.2.5 BCUSPEEDREG (0x0B00 000A)**
p.288	Addition to and modification of description in **11.2.7 BCURFCNTREG(0x0B00 000E)**
p.289	Modification of description in **11.2.8 REVIDREG (0x0B00 0010)**
p.290	Modification of description in **11.2.9 BCURFCOUNTREG (0x0B00 0012)**
p.294	Modification of description in **11.2.13 SROMMODEREG (0x0B00 001C)**
pp.295, 296	Addition to and modification of description in **11.2.14 SDRAMCNTREG (0x0B00 001E)**
p.297	Addition of description in **11.2.15 BCUTOUTCNTREG (0x0B00 0300)**
p.298	Addition of description in **11.2.16 BCUTOUTCOUNTREG (0x0B00 0302)**
p.299	Modification of description in **Table 11-2 Address Bit Correspondence Between ADD Bus and External Device**
p.302	Addition of description in **11.3.1 Connection to DRAM**
p.306	Modification of description in **Table 11-7 Access Size Restrictions for Address Spaces**
p.309	Addition of description in **11.4.5 (1) Access size**
p.310	Addition of **11.4.5 (4) Access cycle reduction mode**
p.314	Addition of description in **11.5 Bus Operation**, Modification of description in **11.5.1 ROM access**
pp.318 to 321	Modification of **Figures 11-7** to **11-12**
p.322	Modification of description in **11.5.2 (3) Bus operations in high-speed system bus**
pp.323, 324	Modification of **Figures 11-13** to **11-16**
p.342	Modification of **Figure 11-40 Bus Hold in Fullspeed Mode and Standby Mode (When Using EDO RAM)**
p.344	Addition of **Figure 11-42 Bus Hold in Fullspeed Mode and Standby Mode (When Using SDRAM)**
p.345	Addition of **Figure 11-43 Bus Hold in Suspend Mode (When Using SDRAM)**
p.470	Addition to and modification of **20.2 (1) Disable state**
p.471	Modification of description in **20.2 (5) WaitPenTouch state**
p.488	Addition of **20.4 (4) Transition flow when entering WaitPenTouch state (WaitPenTouch state)** and **(5) Transition flow when returning to Suspend mode (WaitPenTouch state)**
p.488	Addition of description in **20.4 (6) Transition flow when entering Suspend mode (Disable state)**
p.489	Addition of description in **20.4 (7) Transition flow when returning to Suspend mode (Disable state)**
p.587	Addition of **Figure 27-1 FIR Internal Block**
p.611	Modification of description in **27.2.18 TXFL (0x0C00 006E)**
p.612	Modification of description in **27.2.19 MRXF (0x0C00 0070)**
p.613	Modification of description in **27.2.20 RXFL (0x0C00 0074)**

The mark ★ shows major revised points.

[MEMO]

INTRODUCTION

Readers

This manual is intended for users who wish to understand the functions of the VR4121 and to develop application systems using this microprocessor.

Purpose

This manual introduces the architecture and hardware functions of the VR4121 to users, following the organization described below.

Organization

This manual consists of the following contents:

- Introduction
- Pipeline operation
- Cache organization and memory management system
- Exception processing
- Initialization interface
- Interrupts
- Peripheral units
- Instruction set details

How to read this manual

It is assumed that the reader of this manual has general knowledge in the fields of electrical engineering, logic circuits, and microcontrollers.

The VR4000™ in this manual includes the VR4400™.

To learn in detail about the function of a specific instruction,

→ Read **CHAPTER 3 MIPS™ III INSTRUCTION SET SUMMARY**, **CHAPTER 4 MIPS16 INSTRUCTION SET**, **CHAPTER 28 MIPS III INSTRUCTION SET DETAILS**, and **CHAPTER 29 MIPS16 INSTRUCTION SET FORMAT**.

To know about the overall functions of the VR4121:

→ Read this manual in the order of the contents.

To know about electrical specifications:

→ Refer to **Data Sheet** which is separately available.

Conventions

Data significance:	Higher digits on the left and lower digits on the right
Active low representation:	XXX# (trailing # after pin and signal names)
Note:	Footnote for item marked with **Note** in the text
Caution:	Information requiring particular attention
Remark:	Supplementary information
Numerical representation:	Binary/Decimal ... XXXX
	Hexadecimal ... 0xXXXX

Prefix indicating the power of 2 (address space, memory capacity):

K (kilo)	$2^{10} = 1,024$
M (mega)	$2^{20} = 1,024^2$
G (giga)	$2^{30} = 1,024^3$
T (tera)	$2^{40} = 1,024^4$
P (peta)	$2^{50} = 1,024^5$
E (exa)	$2^{60} = 1,024^6$

Related Documents

The related documents indicated in this publication may include preliminary versions. However preliminary versions are not marked as such.

Documents Related to Devices

Document Name	Document No.
μPD30121 (VR4121) Data Sheet	To be prepared
VR4121 User's Manual	This Manual
VR4111 User's Manual	U13137E

Application Note

Document Name	Document No.
VR Series^{TM} Application Note programming guide	U10710E

CONTENTS

LIST OF FIGURES (1/6)

Fig. No. Title Page

LIST OF FIGURES (2/6)

LIST OF FIGURES (3/6)

Fig. No. Title Page

LIST OF FIGURES (4/6)

Fig. No. Title Page

LIST OF FIGURES (5/6)

LIST OF FIGURES (6/6)

Fig. No. Title Page

LIST OF TABLES (1/5)

LIST OF TABLES (2/5)

LIST OF TABLES (3/5)

Table No. Title Page

LIST OF TABLES (4/5)

LIST OF TABLES (5/5)

Table No. Title Page

[MEMO]

CHAPTER 1 INTRODUCTION

This chapter describes the outline of the VR4121 (μPD30121), which is a 64-/32-bit RISC microprocessor.

1.1 Features

The VR4121, which is a high-performance 64-/32-bit microprocessor employing the RISC (reduced instruction set computer) architecture developed by MIPS, is one of the RISC microprocessor VR-Series products manufactured by NEC.

The VR4121 consists of the ultra-low-power consumption VR4120$^{\text{TM}}$ CPU core with cache memory, high-speed product-sum operation unit, and address management unit. It also has interface units for the peripheral circuits, such as DMA, software modem interface, serial interface, keyboard interface, IrDA interface, touch panel interface, real-time clock, A/D converter, and D/A converter required for the battery-driven portable information equipment. The external bus width of this device can be selected between 32 bits and 16 bits and supports external devices that require the performance level of a color LCD controller.

This processor supports instruction set architecture (ISA) of MIPS I, MIPS II, MIPS III, and MIPS16. It does not support LL, LLD, SC, SCD, and floating point instructions.

Note that this processor does not incorporate a secondary cache, multi-processor processing function, or floating point arithmetic function.

The features of the VR4121 are described below.

○ Employs 64-bit RISC CPU Core (VR4120 equivalent)
○ Internal 64-bit data processing
○ Optimized 6-stage pipeline
○ Conforms to MIPS I, II, III instruction sets (with the FPU, LL, LLD, SC, and SCD instructions left out)
○ Supports MIPS16 instruction
○ Supports high-speed product-sum operation instructions to execute applications in high speed
○ On-chip cache memory   Instruction cache: 16 Kbytes
                                        Data cache: 8 Kbytes
○ Translation lookaside buffer (TLB) for virtual address management
○ Address space   Physical address space: 32 bits
                            Virtual address space: 40 bits
○ On-chip peripheral units suited for portable equipment

- Memory controller (supports ROM, EDO-type DRAM, synchronous DRAM (SDRAM), synchronous ROM (SROM), and flash memory)
- Supports ISA-bus interface
- Keyboard interface
- Touch panel interface (on-chip 4-channel A/D converter)
- Controller complying with IrDA 1.1 (FIR)
- Software modem interface supporting the HSP modem$^{\text{TM}}$ of PC-TEL
- DMA controller
- Serial interface
- Debug serial interfaces
- Interrupt controller
- Audio interface (on-chip digital I/O, A/D and D/A converters)
- General-purpose A/D converter: 3 channels
- General-purpose ports

○ Effective power management features, which include the following four operating modes:
- Full-speed mode: normal operating mode in which all clocks operate
- Standby mode: all internal clocks stop except for interrupt-related clocks
- Suspend mode: bus clock and all internal clocks stop except for interrupt-related clocks
- Hibernate mode: all clocks generated by the CPU core stop

○ External input clock: 32.768 kHz, 18.432 MHz (for internal CPU core and peripheral unit operation), 48 MHz (dedicated for IrDA interface)

○ Clock supply management function for each on-chip peripheral unit to implement low-power consumption

★ ○ Operation supply voltage: V$_{DD}$2 = 2.5 V (internal), V$_{DD}$3 = 3.3 V (external)

★ 1.2 Ordering Information

Part Number	Package	Maximum Operation Frequency
μPD30121F1-131-GA	224-pin fine-pitch BGA (16 × 16 mm)	131 MHz
μPD30121F1-168-GA	224-pin fine-pitch BGA (16 × 16 mm)	168 MHz

1.3 64-bit Architecture

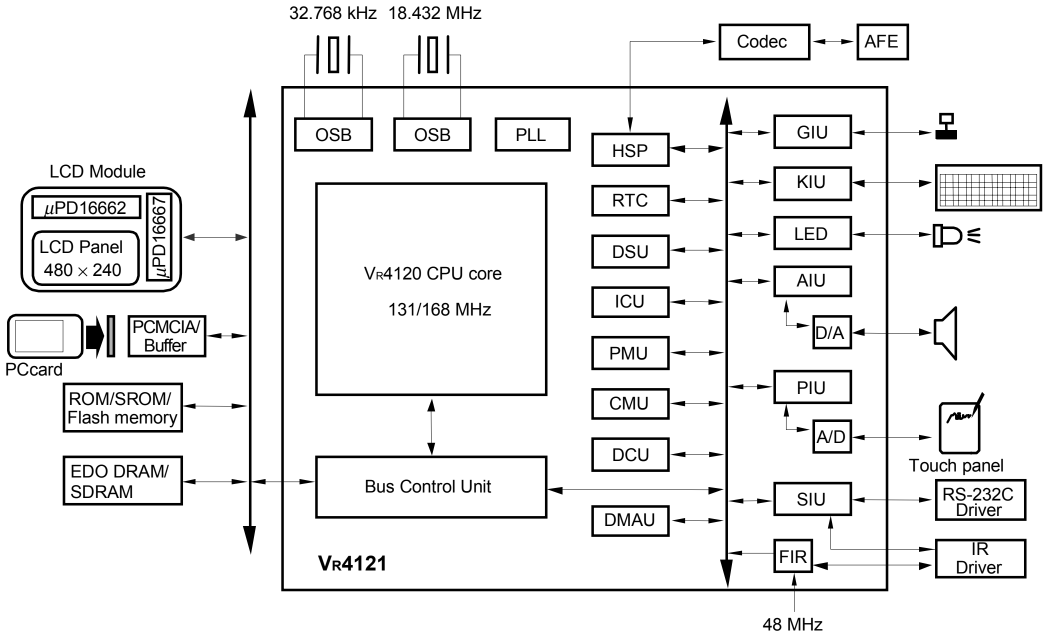
The VR4121 is a high-performance 64-bit microprocessor. However, it can also run 32-bit applications.

1.4 VR4121 Processor

The VR4121 consists of the VR4120 CPU core and seventeen peripheral units. It can connect external controllers directly.

Figure 1-1 describes an internal block diagram of the VR4121 processor and example of connection to external blocks.

Figure 1-1. VR4121 Internal Block Diagram and Example of Connection to External Blocks



1.4.1 Internal block structure

The following is an outline of the peripheral units.

For the CPU core, refer to **1.5 VR4120 CPU Core**.

(1) Bus control unit (BCU)

In the VR4121, the bus control unit (BCU) transfers data between the VR4120 CPU core and SysAD bus. It also controls external circuits, such as the LCD controller connected to the system bus, DRAM (EDO-type DRAM or synchronous DRAM), ROM (flash memory, masked ROM, or synchronous ROM), and PCMCIA controller, and transfers data between the VR4121 and these external devices, using the address and data buses.

(2) Real-time clock unit (RTC)

The real-time clock (RTC) is provided with an accurate counter that operates on a 32.768-kHz clock pulse supplied from the clock generator. It is also provided with several counters and compare registers for controlling various interrupts.

(3) Deadman's switch unit (DSU)

The Deadman's switch unit (DSU) is used to check whether the processor is running normally. If the register of this unit is not cleared by software within a specified period, the system is shut down.

(4) Interrupt control unit (ICU)

The interrupt control unit (ICU) controls interrupt requests that are caused by factors either internal or external to the VR4121, and informs the VR4120 CPU core when an interrupt request occurs.

(5) Power management unit (PMU)

The power management unit (PMU) outputs signals necessary to control the power of the entire system including the VR4121. The signals are used to control the PLL of the VR4120 CPU core and the internal clocks (Pclock, TClock, and MasterOut) in low-power modes.

(6) Direct memory access address unit (DMAAU)

The direct memory access address unit (DMAAU) controls the address of three different DMA transfers.

(7) Direct memory access control unit (DCU)

The direct memory access control unit (DCU) controls the arbitration of three different DMA transfers.

(8) Clock mask unit (CMU)

The clock mask unit (CMU) controls the way the clocks (TClock and MasterOut) are supplied from the VR4120 CPU core to internal peripheral units.

(9) General purpose I/O unit (GIU)

The general purpose I/O unit (GIU) controls 49 general-purpose I/O pins.

(10) Audio interface unit (AIU)

The audio interface unit (AIU) executes mic-input sampling and audio signal output by controlling the internal A/D converter and D/A converter.

(11) Keyboard interface unit (KIU)

The keyboard interface unit (KIU) has 12 scan lines and 8 detection lines. It can detect when any of 64/80/96 keys are entered. It supports key rollover for two to three continuous strokes.

(12) Touch panel interface unit (PIU)

The touch panel interface unit (PIU) detects when the touch panel is touched, by controlling the internal A/D converter.

(13) Debug serial interface unit (DSIU)

The debug serial interface unit (DSIU) is a serial interface for debugging. It supports a maximum transfer rate of 115 kbps.

(14) Serial interface unit (SIU)

The serial interface unit (SIU) conforms to the RS-232-C specification and is compatible with NS16550. It supports a maximum transfer rate of 1.15 Mbps. Also available is an IrDA serial interface supporting a maximum transfer rate of 4 Mbps, but this interface and the RS-232-C interface are mutually exclusive.

(15) Fast IrDA interface unit (FIR)

The FIR unit is a unit for performing 0.5- to 4-Mbps IrDA communication. This unit operates based on a dedicated 48-MHz clock input.

(16) Host signal processing unit (HSP)

The HSP unit is used to realize a software modem. It interfaces the CPU core with an external codec device, and controls them.

(17) Light emitting diode unit (LED)

The LED unit is used to control the lighting of external LED.

1.4.2 I/O registers

The I/O registers are used for peripheral unit control. Lists of registers for peripheral units are as follows.

Table 1-1. BCU Registers

Register Symbols	Function	Physical Address
BCUCNTREG1	BCU Control Register 1	0x0B00 0000
BCUCNTREG2	BCU Control Register 2	0x0B00 0002
ROMSIZEREG	ROM Size Register	0x0B00 0004
RAMSIZEREG	DRAM Size Register	0x0B00 0006
BCUSPEEDREG	BCU Access Cycle Change Register	0x0B00 000A
BCUERRSTREG	BCU BUS ERROR Status Register	0x0B00 000C
BCURFCNTREG	BCU Refresh Control Register	0x0B00 000E
REVIDREG	Peripheral Unit Revision ID Register	0x0B00 0010
BCURFCOUNTREG	BCU Refresh Cycle Count Register	0x0B00 0012
CLKSPEEDREG	Clock Setting Register	0x0B00 0014
BCUCNTREG3	BCU Control Register 3	0x0B00 0016
SDRAMMODEREG	SDRAM Mode Register	0x0B00 001A
SROMMODEREG	SROM Mode Register	0x0B00 001C
SDRAMCNTREG	SDRAM Control Register	0x0B00 001E
★ BCUTOUTCNTREG	BCU Timeout Control Register	0x0B00 0300
★ BCUTOUTCOUNTREG	BCU Timeout Count Register	0x0B00 0302

Table 1-2. DMAAU Registers

Register Symbols	Function	Physical Address
AIUIBALREG	AIU IN DMA Base Address Register Low	0x0B00 0020
AIUIBAHREG	AIU IN DMA Base Address Register High	0x0B00 0022
AIUIALREG	AIU IN DMA Address Register Low	0x0B00 0024
AIUIAHREG	AIU IN DMA Address Register High	0x0B00 0026
AIUOBALREG	AIU OUT DMA Base Address Register Low	0x0B00 0028
AIUOBAHREG	AIU OUT DMA Base Address Register High	0x0B00 002A
AIUOALREG	AIU OUT DMA Address Register Low	0x0B00 002C
AIUOAHREG	AIU OUT DMA Address Register High	0x0B00 002E
FIRBALREG	FIR DMA Base Address Register Low	0x0B00 0030
FIRBAHREG	FIR DMA Base Address Register High	0x0B00 0032
FIRALREG	FIR DMA Address Register Low	0x0B00 0034
FIRAHREG	FIR DMA Address Register High	0x0B00 0036

Table 1-3. DCU Registers

Register Symbols	Function	Physical Address
DMARSTREG	DMA Reset Register	0x0B00 0040
DMAIDLEREG	DMA Sequencer Status Register	0x0B00 0042
DMASENREG	DMA Sequencer Enable Register	0x0B00 0044
DMAMSKREG	DMA Mask Register	0x0B00 0046
DMAREQREG	DMA Request Register	0x0B00 0048
TDREG	Transfer Direction Setting Register	0x0B00 004A

Table 1-4. CMU Register

Register Symbol	Function	Physical Address
CMUCLKMSK	CMU Clock Mask Register	0x0B00 0060

Table 1-5. ICU Registers

Register Symbols	Function	Physical Address
SYSINT1REG	Level 1 System Interrupt Register 1	0x0B00 0080
PIUINTREG	Level 2 PIU Interrupt Register	0x0B00 0082
AIUINTREG	Level 2 AIU Interrupt Register	0x0B00 0084
KIUINTREG	Level 2 KIU Interrupt Register	0x0B00 0086
GIUINTLREG	Level 2 GIU Interrupt Register Low	0x0B00 0088
DSIUINTREG	Level 2 DSIU Interrupt Register	0x0B00 008A
MSYSINT1REG	Level 1 Mask System Interrupt Register 1	0x0B00 008C
MPIUINTREG	Level 2 Mask PIU Interrupt Register	0x0B00 008E
MAIUINTREG	Level 2 Mask AIU Interrupt Register	0x0B00 0090
MKIUINTREG	Level 2 Mask KIU Interrupt Register	0x0B00 0092
MGIUINTLREG	Level 2 Mask GIU Interrupt Register Low	0x0B00 0094
MDSIUINTREG	Level 2 Mask DSIU Interrupt Register	0x0B00 0096
NMIREG	Battery Interrupt Select Register	0x0B00 0098
SOFTINTREG	Software Interrupt Register	0x0B00 009A
SYSINT2REG	Level 1 System Interrupt Register 2	0x0B00 0200
GIUINTHREG	Level 2 GIU Interrupt Register High	0x0B00 0202
FIRINTREG	Level 2 FIR Interrupt Register	0x0B00 0204
MSYSINT2REG	Level 1 Mask System Interrupt Register 2	0x0B00 0206
MGIUINTHREG	Level 2 Mask GIU Interrupt Register High	0x0B00 0208
MFIRINTREG	Level 2 Mask FIR Interrupt Register	0x0B00 020A

Table 1-6. PMU Registers

Register Symbols	Function	Physical Address
PMUINTREG	PMU Interrupt/Status Register	0x0B00 00A0
PMUCNTREG	PMU Control Register	0x0B00 00A2
PMUINT2REG	PMU Interrupt/Status Register 2	0x0B00 00A4
PMUCNT2REG	PMU Control Register 2	0x0B00 00A6
PMUWAITREG	PMU Wait Count Register	0x0B00 00A8
PMUDIVREG	PMU Div Mode Register	0x0B00 00AC

Table 1-7. RTC Registers

Register Symbols	Function	Physical Address
ETIMELREG	Elapsed Time Timer Register Low	0x0B00 00C0
ETIMEMREG	Elapsed Time Timer Register Middle	0x0B00 00C2
ETIMEHREG	Elapsed Time Timer Register High	0x0B00 00C4
ECMPLREG	Elapsed Time Timer Compare Register Low	0x0B00 00C8
ECMPMREG	Elapsed Time Timer Compare Register Middle	0x0B00 00CA
ECMPHREG	Elapsed Time Timer Compare Register High	0X0B00 00CC
RTCL1LREG	RTC Long 1 Timer Register Low	0x0B00 00D0
RTCL1HREG	RTC Long 1 Timer Register High	0x0B00 00D2
RTCL1CNTLREG	RTC Long 1 Timer Count Register Low	0x0B00 00D4
RTCL1CNTHREG	RTC Long 1 Timer Count Register High	0x0B00 00D6
RTCL2LREG	RTC Long 2 Timer Register Low	0x0B00 00D8
RTCL2HREG	RTC Long 2 Timer Register High	0x0B00 00DA
RTCL2CNTLREG	RTC Long 2 Timer Count Register Low	0x0B00 00DC
RTCL2CNTHREG	RTC Long 2 Timer Count Register High	0x0B00 00DE
TCLKLREG	TClock Counter Register Low	0x0B00 01C0
TCLKHREG	TClock Counter Register High	0x0B00 01C2
TCLKCNTLREG	TClock Counter Count Register Low	0x0B00 01C4
TCLKCNTHREG	TClock Counter Count Register High	0x0B00 01C6
RTCINTREG	RTC Interrupt Register	0x0B00 01DE

Table 1-8. DSU Registers

Register Symbols	Function	Physical Address
DSUCNTREG	DSU Control Register	0x0B00 00E0
DSUSETREG	DSU Cycle Set Register	0x0B00 00E2
DSUCLRREG	DSU Clear Register	0x0B00 00E4
DSUTIMREG	DSU Elapsed Time Register	0x0B00 00E6

Table 1-9. GIU Registers

Register Symbols	Function	Physical Address
GIUIOSELL	GPIO Input/Output Select Register L	0x0B00 0100
GIUIOSELH	GPIO Input/Output Select Register H	0x0B00 0102
GIUPIODL	GPIO Port Input/Output Data Register L	0x0B00 0104
GIUPIODH	GPIO Port Input/Output Data Register H	0x0B00 0106
GIUINTSTATL	GPIO Interrupt Status Register L	0x0B00 0108
GIUINTSTATH	GPIO Interrupt Status Register H	0x0B00 010A
GIUINTENL	GPIO Interrupt Enable Register L	0x0B00 010C
GIUINTENH	GPIO Interrupt Enable Register H	0x0B00 010E
GIUINTTYPL	GPIO Interrupt Type Select Register L	0x0B00 0110
GIUINTTYPH	GPIO Interrupt Type Select Register H	0x0B00 0112
GIUINTALSELL	GPIO Interrupt Active Level Select Register L	0x0B00 0114
GIUINTALSELH	GPIO Interrupt Active Level Select Register H	0x0B00 0116
GIUINTHTSELL	GPIO Interrupt Hold/Through Select Register L	0x0B00 0118
GIUINTHTSELH	GPIO Interrupt Hold/Through Select Register H	0x0B00 011A
GIUPODATL	GPIO Port Output Data Register L	0x0B00 011C
GIUPODATH	GPIO Port Output Data Register H	0x0B00 011E
GIUUSEUPDN	GPIO Pull-up/Pull-down Enable Register	0x0B00 02E0
GIUTERMUPDN	GPIO Pull-up/Pull-down Set Register	0x0B00 02E2

Table 1-10. PIU Registers

Register Symbols	Function	Physical Address
PIUCNTREG	PIU Control Register	0x0B00 0122
PIUINTREG	PIU Interrupt Register	0x0B00 0124
PIUSIVLREG	PIU Data Sampling Interval Register	0x0B00 0126
PIUSTBLREG	PIU A/D Converter Start Delay Register	0x0B00 0128
PIUCMDREG	PIU A/D Command Register	0x0B00 012A
PIUASCNREG	PIU A/D Port Scan Register	0x0B00 0130
PIUAMSKREG	PIU A/D Scan Mask Register	0x0B00 0132
PIUCIVLREG	PIU Check Interval Register	0x0B00 013E
PIUPB00REG	PIU Page 0 Buffer 0 Register	0x0B00 02A0
PIUPB01REG	PIU Page 0 Buffer 1 Register	0x0B00 02A2
PIUPB02REG	PIU Page 0 Buffer 2 Register	0x0B00 02A4
PIUPB03REG	PIU Page 0 Buffer 3 Register	0x0B00 02A6
PIUPB10REG	PIU Page 1 Buffer 0 Register	0x0B00 02A8
PIUPB11REG	PIU Page 1 Buffer 1 Register	0x0B00 02AA
PIUPB12REG	PIU Page 1 Buffer 2 Register	0x0B00 02AC
PIUPB13REG	PIU Page 1 Buffer 3 Register	0x0B00 02AE
PIUAB0REG	PIU A/D Scan Buffer 0 Register	0x0B00 02B0
PIUAB1REG	PIU A/D Scan Buffer 1 Register	0x0B00 02B2
PIUAB2REG	PIU A/D Scan Buffer 2 Register	0x0B00 02B4
PIUAB3REG	PIU A/D Scan Buffer 3 Register	0x0B00 02B6
PIUPB04REG	PIU Page 0 Buffer 4 Register	0x0B00 02BC
PIUPB14REG	PIU Page 1 Buffer 4 Register	0x0B00 02BE

Table 1-11. AIU Registers

Register Symbols	Function	Physical Address
MDMADATREG	Mike Input DMA Data Register	0x0B00 0160
SDMADATREG	Speaker Output DMA Data Register	0x0B00 0162
SODATREG	Speaker Output Data Register	0x0B00 0166
SCNTREG	Speaker Output Control Register	0x0B00 0168
SCNVRREG	Speaker Conversion Rate Register	0x0B00 016A
MIDATREG	Mike Input Data Register	0x0B00 0170
MCNTREG	Mike Input Control Register	0x0B00 0172
MCNVRREG	Mike Conversion Rate Register	0x0B00 0174
DVALIDREG	Data Valid Register	0x0B00 0178
SEQREG	Sequential Operation Enable Register	0x0B00 017A
INTREG	AIU Interrupt Register	0x0B00 017C

Table 1-12. KIU Registers

Register Symbols	Function	Physical Address
KIUDAT0	KIU Data0 Register	0x0B00 0180
KIUDAT1	KIU Data1 Register	0x0B00 0182
KIUDAT2	KIU Data2 Register	0x0B00 0184
KIUDAT3	KIU Data3 Register	0x0B00 0186
KIUDAT4	KIU Data4 Register	0x0B00 0188
KIUDAT5	KIU Data5 Register	0x0B00 018A
KIUSCANREP	KIU Scan/Repeat Register	0x0B00 0190
KIUSCANS	KIU Scan Status Register	0x0B00 0192
KIUWKS	KIU Wait Keyscan Stable Register	0x0B00 0194
KIUWKI	KIU Wait Keyscan Interval Register	0x0B00 0196
KIUINT	KIU Interrupt Register	0x0B00 0198
KIURST	KIU Reset Register	0x0B00 019A
KIUGPEN	KIU General Purpose Output Enable Register	0x0B00 019C
SCANLINE	KIU Scan Line Register	0x0B00 019E

Table 1-13. DSIU Registers

Register Symbols	Function	Physical Address
PORTREG	Port Change Register	0x0B00 01A0
MODEMREG	Modem Control Register	0x0B00 01A2
ASIM00REG	Asynchronous Mode 0 Register	0x0B00 01A4
ASIM01REG	Asynchronous Mode 1 Register	0x0B00 01A6
RXB0RREG	Receive Buffer Register (Extended)	0x0B00 01A8
RXB0LREG	Receive Buffer Register	0x0B00 01AA
TXS0RREG	Transmit Data Register (Extended)	0x0B00 01AC
TXS0LREG	Transmit Data Register	0x0B00 01AE
ASIS0REG	Status Register	0x0B00 01B0
INTR0REG	Debug SIU Interrupt Register	0x0B00 01B2
BPRM0REG	Baud-rate Generator Prescaler Mode Register	0x0B00 01B6
DSIURESETREG	Debug SIU Reset Register	0x0B00 01B8

Table 1-14. LED Registers

Register Symbols	Function	Physical Address
LEDHTSREG	LED H Time Set Register	0x0B00 0240
LEDLTSREG	LED L Time Set Register	0x0B00 0242
LEDCNTREG	LED Control Register	0x0B00 0248
LEDASTCREG	LED Auto Stop Time Count Register	0x0B00 024A
LEDINTREG	LED Interrupt Register	0x0B00 024C

Table 1-15. SIU Registers

Register Symbols	Function	LCR[7]	Physical Address
SIURB	Receiver Buffer Register (Read)	0	0x0C00 0000
SIUTH	Transmitter Holding Register (Write)		
SIUDLL	Divisor Latch (Least Significant Byte) Register	1	
SIUIE	Interrupt Enable Register	0	0x0C00 0001
SIUDLM	Divisor Latch (Most Significant Byte) Register	1	
SIUIID	Interrupt Identification Register (Read)	–	0x0C00 0002
SIUFC	FIFO Control Register (Write)		
SIULC	Line Control Register	–	0x0C00 0003
SIUMC	MODEM Control Register	–	0x0C00 0004
SIULS	Line Status Register	–	0x0C00 0005
SIUMS	MODEM Status Register	–	0x0C00 0006
SIUSC	Scratch Register	–	0x0C00 0007
SIUIRSEL	SIU/FIR IrDA Selector	–	0x0C00 0008
SIURESET	SIU Reset Register	–	0x0C00 0009
SIUCSEL	SIU Echo Back Control Register	–	0x0C00 000A

Remark LCR7 is bit 7 of the SIULC register.

Table 1-16. HSP Registers

Register Symbols	Function	Physical Address
HSPINIT	HSP Initialize Register	0x0C00 0020
HSPDATA(7:0)	HSP Data Register (Low-order)	0x0C00 0022
HSPDATA(15:8)	HSP Data Register (High-order)	0x0C00 0023
HSPINDEX	HSP Index Register	0x0C00 0024
HSPID(7:0)	HSP ID Register	0x0C00 0028
HSPPCS(7:0)	HSP I/O Address Program Confirmation Register	0x0C00 0029
HSPPCTEL(7:0)	HSP Signature Checking Port	0x0C00 0029

Table 1-17. FIR Registers

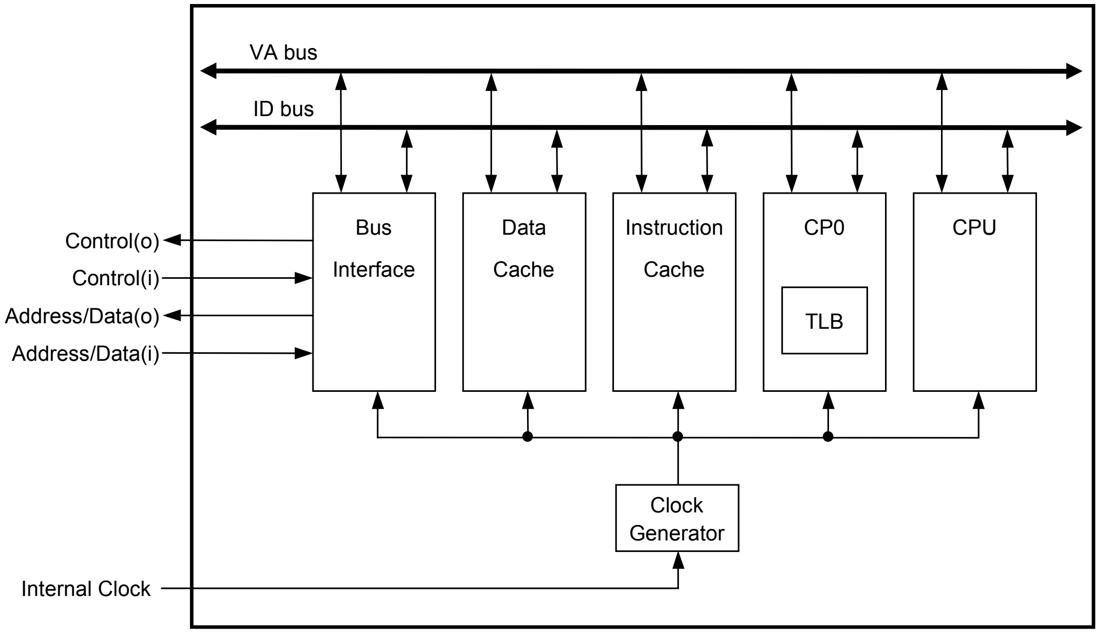
Register Symbols	Function	Physical Address
FRSTR	FIR Reset Register	0x0C00 0040
DPINTR	DMA Page Interrupt Register	0x0C00 0042
DPCNTR	DMA Page Control Register	0x0C00 0044
TDR	Transmit Data Register	0x0C00 0050
RDR	Receive Data Register	0x0C00 0052
IMR	Interrupt Mask Register	0x0C00 0054
FSR	FIFO Setup Register	0x0C00 0056
IRSR1	IR Setup Register 1	0x0C00 0058
CRCSR	CRC Setup Register	0x0C00 005C
FIRCR	FIR Control Register	0x0C00 005E
MIRCR	MIR Control Register	0x0C00 0060
DMACR	DMA Control Register	0x0C00 0062
DMAER	DMA Enable Register	0x0C00 0064
TXIR	Transmission Indicate Register	0x0C00 0066
RXIR	Reception Indicate Register	0x0C00 0068
IFR	Interrupt Flag Register	0x0C00 006A
RXSTS	Reception Status Register	0x0C00 006C
TXFL	Transmit Frame Length Register	0x0C00 006E
MRXF	Maximum Receive Frame Length Register	0x0C00 0070
RXFL	Receive Frame Length Register	0x0C00 0074

1.5 VR4120 CPU Core

Figure 1-2 shows the internal block diagram of the VR4120 CPU 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 a bus interface.

Figure 1-2. VR4120 CPU Core Internal Block Diagram



1.5.1 Internal block configuration

(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) 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.

(3) Instruction cache

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

(4) Data cache

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

(5) CPU bus interface

The CPU bus interface controls data transmission/reception between the VR4120 CPU core and the BCU, which is one of peripheral units. The VR4120 CPU 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.

(6) Clock generator

The following clock inputs are oscillated and supplied to internal units.

- 32.768-kHz clock for RTC unit:
  oscillating a 32.768-kHz crystal resonator input via an internal oscillator to supply to the RTC unit.
- 18.432-MHz clock for serial interface and the VR4121's reference operating clock:
  oscillating an 18.432-MHz crystal resonator input via an internal oscillator, and then multiplying it by phase-locked loop (PLL) to generate a pipeline clock (PClock). The internal bus clock (Tclock and VTClock) is generated from PClock and supplied to peripheral units.

1.5.2 CPU registers

The VR4120 CPU core has the following registers:

- general-purpose registers (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.

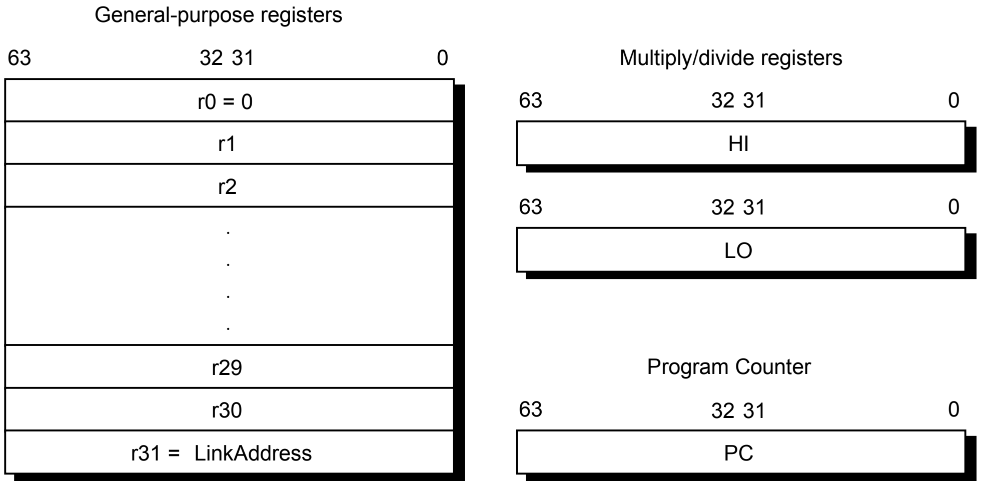
The register group is provided within the CP0 (system control coprocessor), to process exceptions and to manage addresses.

CPU registers can operate as either 32-bit or 64-bit registers, depending on the VR4121 processor operation mode.

The operation of the CPU register differs depending on what instructions are executed: 32-bit instructions or MIPS16 instructions. For details, refer to **CHAPTER 4 MIPS16 INSTRUCTION SET SUMMARY.**

Figure 1-3 shows the CPU registers.

Figure 1-3. VR4121 CPU Registers



The VR4121 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 **1.5.5 Coprocessors (CP0).**

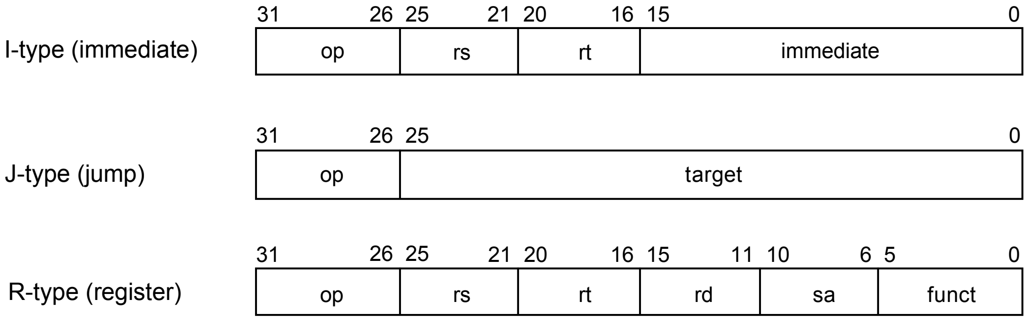
1.5.3 CPU instruction set overview

For CPU instructions, there are two types of instructions – 32-bit length instruction (MIPS III) and 16-bit length instruction (MIPS16).

(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 1-4: immediate (I-type), jump (J-type), and register (R-type). The field of each instruction format is described in **CHAPTER 3 MIPS III INSTRUCTION SET SUMMARY**.

Figure 1-4. 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.

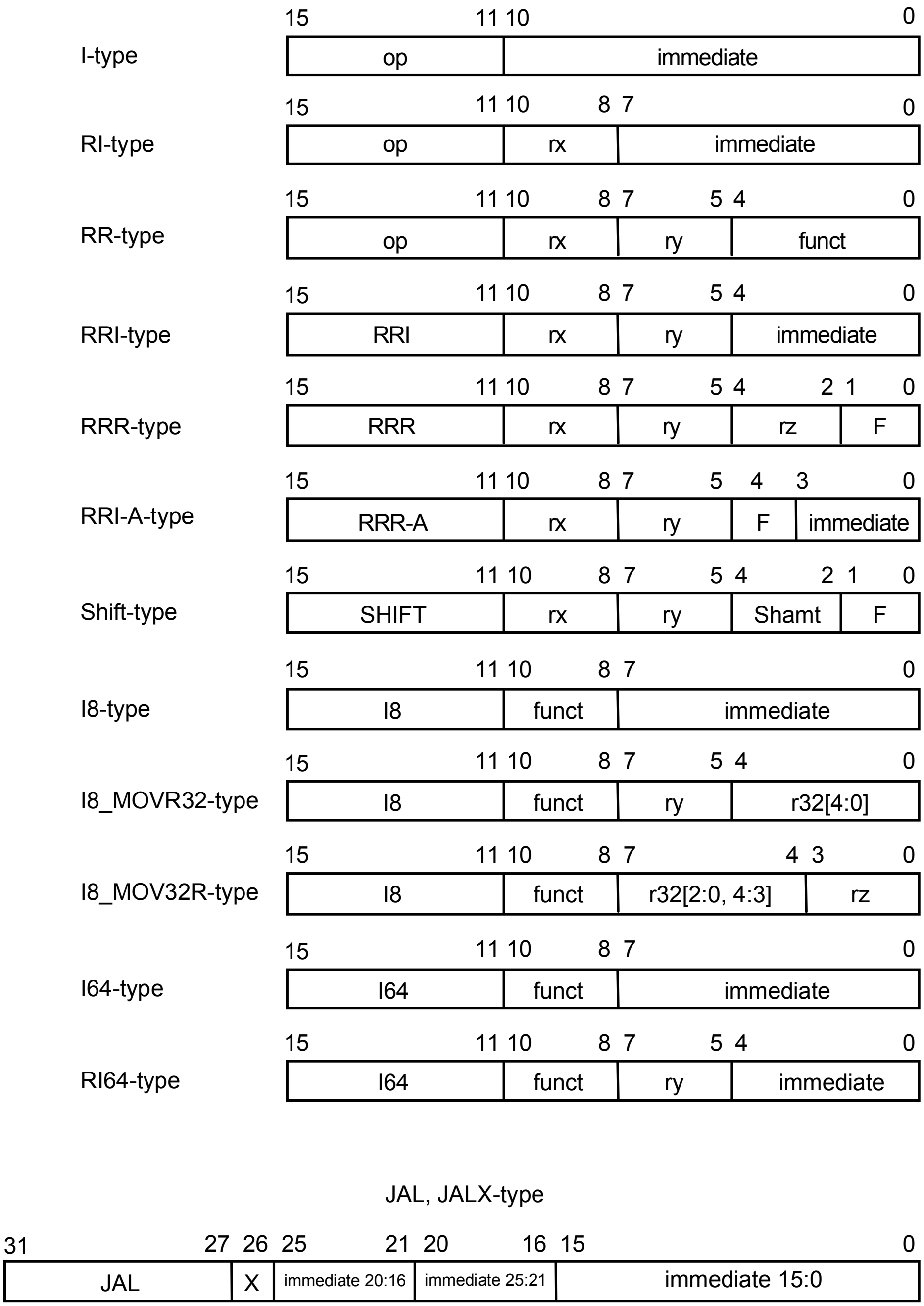
For the operation of each instruction, refer to **CHAPTER 3 MIPS III INSTRUCTION SET SUMMARY** and **CHAPTER 28 MIPS III INSTRUCTION SET DETAILS.**

(2) MIPS16 instruction

All the CPU instructions except for JAL and JALX are 16-bit length when executing MIPS16 instructions, and they are classified into thirteen instruction formats as shown in Figure 1-5.

The field of each instruction format is described in **CHAPTER 4 MIPS 16 INSTRUCTION SET.**

Figure 1-5. CPU Instruction Formats (16-bit length instruction)



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

(a) Load and store instructions move data between memory and general-purpose registers. They include RRI-, RI-, I8-, and RI64-types.
(b) Computational instructions perform arithmetic, logical, shift, and multiply and divide operations on values in registers. They include RI-, RRIA-, I8-, RI64-, I64-, RR-, RRR-, I8_MOVR32-, and I8_MOV32R-types.
(c) Jump and branch instructions change the control flow of a program. They include JAL-/JALX-, RR-, RI-, I8-, and I-types.
(d) Special instructions are break and extend instructions. The break instruction transfers control to an exception handler. The extend instruction extends the immediate field of the next instruction. They are RR- and I-types. When extending the immediate field of the next instruction by using the extend instruction, one cycle is needed for executing the extend instruction, and another cycle is needed for executing the next instruction.

For more details of each instruction's operation, refer to **CHAPTER 4 MIPS16 INSTRUCTION SET** and **CHAPTER 29 MIPS16 INSTRUCTION SET FORMAT**.

1.5.4 Data formats and addressing

The VR4121 uses following four data formats:

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

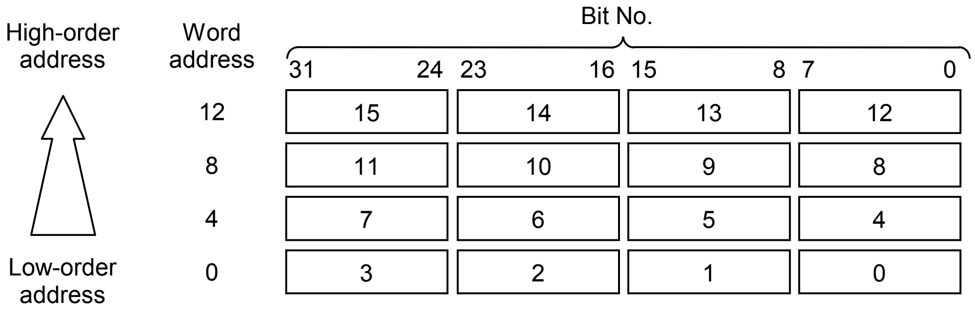
For the VR4120 CPU core, byte ordering within all of the larger data formats - halfword, word, doubleword - can be configured in either big-endian or little-endian order. **However, the VR4121 supports the little-endian order only.**

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 1-6 and 1-7 show this configuration.

In this manual, bit designations are always little endian.

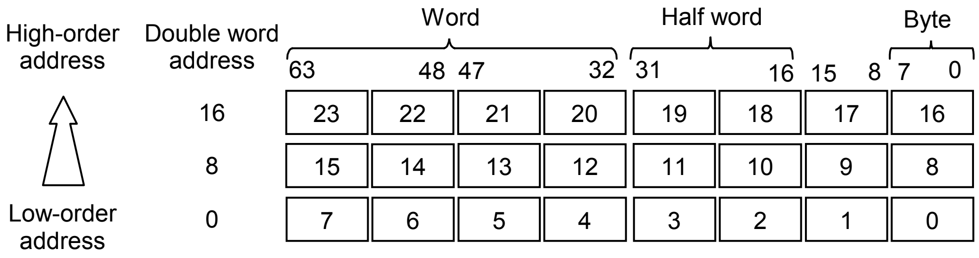
Figure 1-6. Little-Endian Byte Ordering in Word Data



Remarks 1. The lowest byte is the lowest address.

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

Figure 1-7. Little-Endian Byte Ordering in Double Word Data



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...)

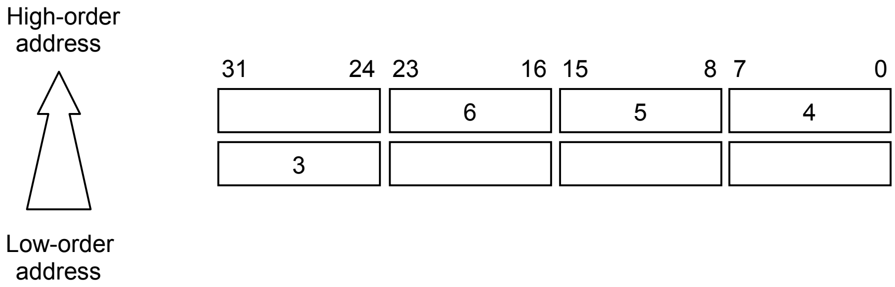
The following special instructions to load and store data that are not aligned on 4-byte (word) or 8-byte (doubleword) boundaries:

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 1-8 shows the access of a misaligned word that has byte address 3 for the little-endian conventions.

Figure 1-8. Misaligned Word Accessing (Little-Endian)



1.5.5 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 1-9 shows the definitions of the CP0 register, and Table 1-18 shows simple descriptions of each register. For the detailed descriptions of the registers related to the virtual system memory, refer to **CHAPTER 6 MEMORY MANAGEMENT SYSTEM**. For the detailed descriptions of the registers related to exception handling, refer to **CHAPTER 7 EXCEPTION PROCESSING**.

Figure 1-9. 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 1-18. 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 VR4121 hardware.

1.5.6 Floating-point unit (FPU)

The VR4121 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.

1.6 CPU Core Memory Management System (MMU)

The VR4121 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 VR4121 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 **CHAPTER 6 MEMORY MANAGEMENT SYSTEM**.

1.6.1 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 Kbyte to 256 Kbytes).

(1) Joint TLB (JTLB)

JTLB holds both an instruction address and data address.

For fast virtual-to-physical address decoding, the VR4121 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 Kbyte to 256 Kbytes. 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.

1.6.2 Operating modes

The VR4121 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 **CHAPTER 6 MEMORY MANAGEMENT SYSTEM** for details.

1.6.3 Cache

The VR4121 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 VR4121 has a storage capacity of 16 Kbytes, while the data cache has a capacity of 8 Kbyte.

A detailed description of caches is given in **CHAPETER 9 CACHE ORGANIZATION AND OPERATION**.

1.7 Instruction Pipeline

The VR4121 has a 6-stage instruction pipeline. Under normal circumstances, one instruction is issued each cycle.

A detailed description of pipeline is provided in **CHAPTER 5 VR4121 PIPELINE**.

1.8 Clock Interface

The VR4121 has the following 11 clocks.

- **CLKX1, CLKX2 (input)**
  These are oscillation inputs of 18.432 MHz, and used to generate operation clocks for the CPU core. They are used for CMU, PIU, AIU, DSIU, SIU, and HSP as well.
- **RTCX1, RTCX2 (input)**
  These are the 32.768-kHz resonator's input pins, which are used by the PMU, RTC, DSU, LED, touch panel interface, and keyboard interface. Some of the GPIO pins are used as the main operation clock. Only this clock continues to operate when the system is in Hibernate mode.
- **FIRCLK (input)**
  This is a 48-MHz clock input, and used for FIR.
- **PClock (internal)**
  This clock is used to control the pipeline used in the VR4120 CPU core, and for units relating to the pipeline. This clock is generated from the clock input of CLKX1 and CLKX2 pins. Its frequency is determined by CLKSEL(0:2) pins.
- **MasterOut (internal)**
  This is a bus clock of the VR4120 CPU core, and used for interrupt control. The contents of the CP0's count register are incremented synchronously with this clock. The initial value for the MasterOut frequency is 1/4 of the TClock frequency (about 8 MHz). This value is determined by the CLKSEL(2:0) pin during an RTC reset.

- **TClock (internal)**

  This is used as the clock for BCU operations, access from the BCU to on-chip peripheral functions, and system bus access. Its frequency is determined (as about 33 MHz) by the setting of the CLKSEL(2:0) pin during an RTC reset. In addition, some of the GPIO pins are used as the main operation clock.

  TClock can also be output from the BUSCLK pin when set to do so via the BCUCNTREG3 register.
- **BUSCLK (output)**

  This clock is supplied to the controller on the system bus. The initial value of this frequency is 1/4 of the TClock frequency (about 8 MHz). BUSCLK output can be changed to TClock output via BCUCNTREG3 register settings.
- **VTClock (internal)**

  This is used as the clock for the BCU's SDRAM and SROM access. Its frequency is the same as TClock (about 33 MHz). PMUDIVREG register settings and a processor restart can be used to change the division rate in relation to PClock to 1/1, 1/2, 1/2.5, 1/3, 1/4, 1/5, or 1/6. However, according to the division rate, it may not be possible to set all of these in relation to the maximum frequency of PClock.
- **SCLK (output)**

  This clock is supplied to the CLK pins in SDRAM and SROM. It has the same frequency as VTClock, but operates only when accessing SDRAM or SROM. After an RTC reset, this clock cannot operate if the SMODE(2:1) pin has been set to disable use of SDRAM and SROM. In such cases, it is used as the ADD25 signal.
- **HSPMCLK (output)**

  This clock is supplied to the external CODEC. Its frequency is determined by the HSPMCLKD register.
- **HSPSCLK (input)**

  This is an operation clock for the external CODEC and the modem interface.

Relationships between CLKSEL(2:0) pin settings and frequencies are indicated as follows.

Table 1-19. CLKSEL Pin Setting and Corresponding Clock Frequency

CLKSEL (2:0)	PClock	VTClock[Note 1]		BUSCLK[Note 2] (When Used for TClock Output)	BUSCLK[Note 2] (When 1/4 of TClock Frequency)	MasterOut
		MIN.	MAX.			
111	RFU	RFU	RFU	RFU	RFU	RFU
110	168.5 MHz	28.1 MHz	56.2 MHz	28.1 MHz	7.0 MHz	7.0 MHz
101	147.5 MHz	29.5 MHz	59.0 MHz	29.5 MHz	7.4 MHz	7.4 MHz
100	131.1 MHz	32.8 MHz	65.5 MHz	32.8 MHz	8.2 MHz	8.2 MHz
011	118.0 MHz	29.5 MHz	59.0 MHz	29.5 MHz	7.4 MHz	7.4 MHz
010	98.3 MHz	32.8 MHz	65.5 MHz	32.8 MHz	8.2 MHz	8.2 MHz
001	90.7 MHz	30.2 MHz	60.5 MHz	30.2 MHz	7.6 MHz	7.6 MHz
000	78.6 MHz	26.2 MHz	52.4 MHz	26.2 MHz	6.6 MHz	6.6 MHz

Notes 1. This pin is set to the MIN. value during an RTC reset. After the reset, its frequency can be changed.

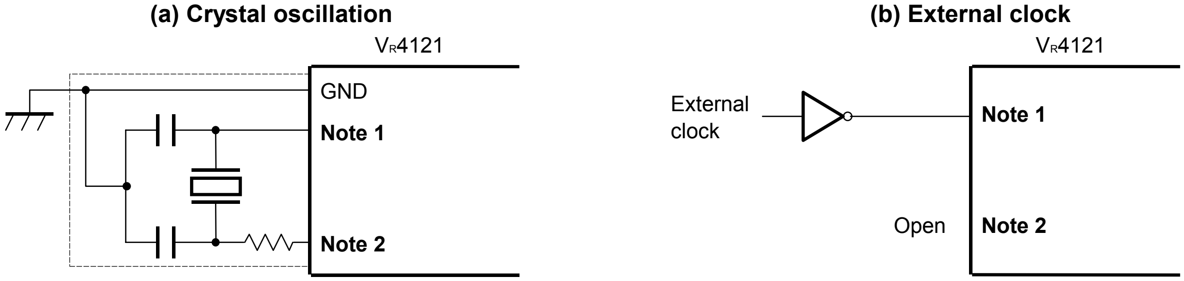
2. During an RTC reset, this pin is set to output at 1/4 the TClock frequency.

Table 1-20. CLKSEL Pin Setting and Frequency of VTClock and SCLK

CLKSEL (2:0)	PClock	VTClock, SCLK						
		Div1.5	Div2	Div2.5	Div3	Div4	Div5	Div6
111	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
110	168.5 MHz	RFU	RFU	RFU	56.2 MHz	42.1 MHz	33.7 MHz	28.1 MHz
101	147.5 MHz	RFU	RFU	59.0 MHz	49.2 MHz	36.9 MHz	29.5 MHz	RFU
100	131.1 MHz	RFU	65.5 MHz	52.4 MHz	43.7 MHz	32.8 MHz	RFU	RFU
011	118.0 MHz	RFU	59.0 MHz	47.2 MHz	39.3 MHz	29.5 MHz	RFU	RFU
010	98.3 MHz	65.5 MHz	49.2 MHz	39.3 MHz	32.8 MHz	RFU	RFU	RFU
001	90.7 MHz	60.5 MHz	45.4 MHz	36.3 MHz	30.2 MHz	RFU	RFU	RFU
000	78.6 MHz	52.4 MHz	39.3 MHz	31.5 MHz	26.2 MHz	RFU	RFU	RFU

Figure 1-10 shows an external circuit of the clock oscillator.

Figure 1-10. External Circuit of Clock Oscillator



Notes 1. CLKX1, RTCX1

2. CLKX2, RTCX2

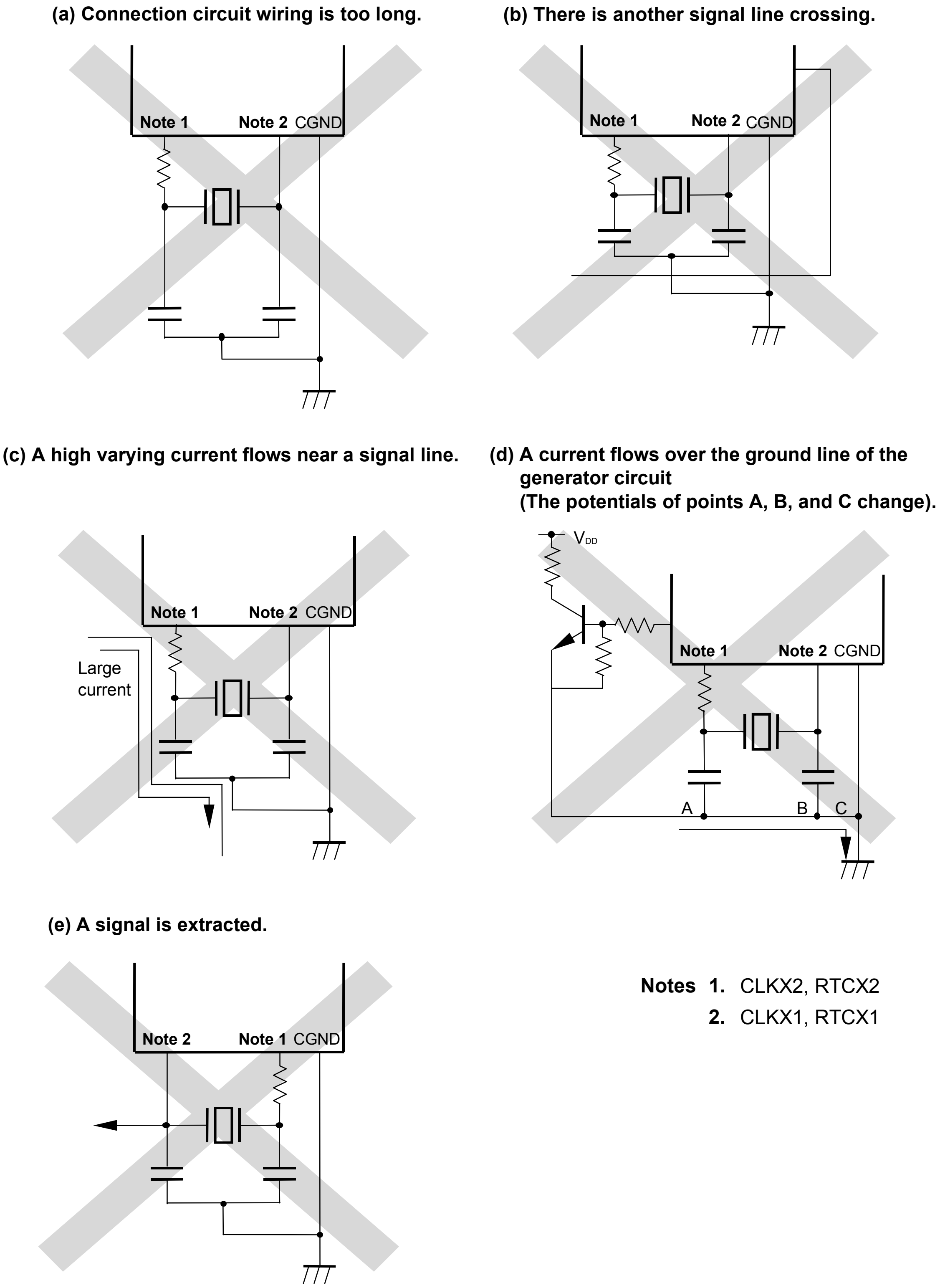
Cautions 1. When using the clock oscillator, wire as follows in the area enclosed by the broken line in the above figures to avoid an adverse effect from wiring capacitance.

- **Keep the wiring length as short as possible.**
- **Do not cross the wiring with the other signal lines. Do not route the wiring near a signal line through which a high fluctuating current flows.**
- **Always keep the ground point of the oscillator capacitor to the same potential as V_{SS}. Do not ground the capacitor to a ground pattern through which a high current flows.**
- **Do not fetch signals from the oscillator**

2. Ensure that no load such as wiring capacity is applied to the CLKX2 or RTCX2 pin when inputting an external clock.

Figure 1-11 shows examples of an oscillator having bad connections.

Figure 1-11. Examples of Oscillator with Bad Connection



(a) Connection circuit wiring is too long.

(b) There is another signal line crossing.

(c) A high varying current flows near a signal line.

(d) A current flows over the ground line of the generator circuit (The potentials of points A, B, and C change).

(e) A signal is extracted.

Notes 1. CLKX2, RTCX2

2. CLKX1, RTCX1

[MEMO]

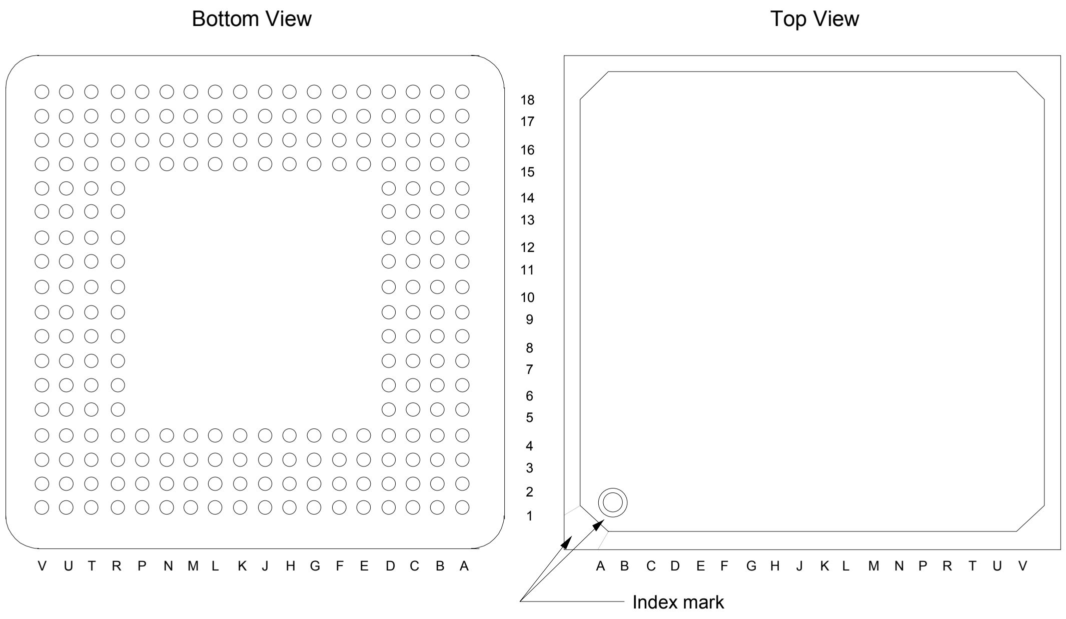
CHAPTER 2 PIN FUNCTIONS

2.1 Pin Configuration

- 224-pin fine-pitch BGA (16 × 16 mm)

  μPD30121F1-131-GA

  μPD30121F1-168-GA


Bottom View

Top View

18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
V U T R P N M L K J H G F E D C B A
A B C D E F G H J K L M N P R T U V
Index mark

Pin No.	Power System	Pin Name	Pin No.	Power System	Pin Name	Pin No.	Power System	Pin Name
A1	3.3 V	VDD3	C15	3.3 V	RTS#/CLKSEL1	H15	3.3 V	GND3
A2	3.3 V	SHB#	C16	3.3 V	GND3	H16	3.3 V	KPORT6
A3	3.3 V	BUSCLK	C17	3.3 V	ILCSENSE	H17	3.3 V	KPORT4
A4	3.3 V	HLDACK#	C18	3.3 V	AFERST#	H18	2.5 V	VDD2
A5	3.3 V	IOCHRDY	D1	3.3 V	DATA5	J1	3.3 V	DATA20/GPIO20
A6	3.3 V	MEMW#	D2	3.3 V	DATA3	J2	3.3 V	DATA17/GPIO17
A7	3.3 V	ADD23	D3	3.3 V	DATA6	J3	3.3 V	DATA22/GPIO22
A8	3.3 V	VDD3	D4	3.3 V	GND3	J4	3.3 V	DATA19/GPIO19
A9	3.3 V	ADD18	D5	3.3 V	MEMCS16#	J15	3.3 V	KSCAN9/GPIO41
A10	3.3 V	ADD15	D6	3.3 V	ADD25/SCLK	J16	3.3 V	VDD3
A11	3.3 V	ADD8	D7	3.3 V	GND3	J17	2.5 V	GND2
A12	3.3 V	ADD7	D8	3.3 V	ADD19	J18	3.3 V	KSCAN11/GPIO43
A13	2.5 V	VDD2	D9	3.3 V	ADD16	K1	3.3 V	DATA23/GPIO23
A14	3.3 V	DCD#/GPIO15	D10	3.3 V	ADD14	K2	3.3 V	DATA26/GPIO26
A15	3.3 V	TxD/CLKSEL2	D11	3.3 V	VDD3	K3	3.3 V	DATA25/GPIO25
A16	3.3 V	IRDOUT#	D12	3.3 V	GND3	K4	3.3 V	DATA21/GPIO21
A17	3.3 V	IRING	D13	3.3 V	ADD4	K15	3.3 V	KSCAN7/GPIO39
A18	3.3 V	VDD3	D14	3.3 V	CTS#	K16	3.3 V	KSCAN10/GPIO42
B1	3.3 V	DATA1	D15	3.3 V	GND3	K17	3.3 V	KSCAN5/GPIO37
B2	3.3 V	IOR#	D16	3.3 V	GND3	K18	3.3 V	KSCAN8/GPIO40
B3	3.3 V	IOW#	D17	3.3 V	SDI	L1	3.3 V	DATA27/GPIO27
B4	3.3 V	LEDOUT#	D18	3.3 V	SDO	L2	3.3 V	DATA31/GPIO31
B5	3.3 V	FIRCLK	E1	3.3 V	DATA9	L3	3.3 V	DATA29/GPIO29
B6	3.3 V	HLDRQ#	E2	3.3 V	DATA4	L4	3.3 V	DATA24/GPIO24
B7	3.3 V	ZWS#	E3	3.3 V	DATA7	L15	3.3 V	KSCAN3/GPIO35
B8	3.3 V	ADD24	E4	3.3 V	DATA10	L16	3.3 V	KSCAN6/GPIO38
B9	3.3 V	ADD21	E15	3.3 V	OPD#	L17	3.3 V	KSCAN0/GPIO32
B10	3.3 V	ADD12	E16	3.3 V	HSPSCLK	L18	3.3 V	KSCAN4/GPIO36
B11	3.3 V	ADD6	E17	3.3 V	FS	M1	3.3 V	DATA30/GPIO30
B12	2.5 V	GND2	E18	3.3 V	HC0	M2	3.3 V	VDD3
B13	3.3 V	DSR#	F1	3.3 V	DATA13	M3	3.3 V	GND3
B14	3.3 V	IRDIN	F2	3.3 V	DATA8	M4	3.3 V	DATA28/GPIO28
B15	3.3 V	FIRDIN#/SEL	F3	3.3 V	DATA11	M15	3.3 V	KSCAN2/GPIO34
B16	3.3 V	BATTINH/BATTINT#	F4	3.3 V	DATA14	M16	3.3 V	MIPS16EN
B17	3.3 V	OFFHOOK	F15	3.3 V	KPORT3	M17	3.3 V	GND3
B18	3.3 V	MUTE	F16	3.3 V	HSPMCLK	M18	3.3 V	KSCAN1/GPIO33
C1	3.3 V	DATA2	F17	3.3 V	TELCON	N1	2.5 V	VDD2
C2	3.3 V	DATA0	F18	3.3 V	KPORT1	N2	3.3 V	ADD3
C3	3.3 V	SMODE2	G1	2.5 V	VDD2	N3	3.3 V	ADD10
C4	3.3 V	CKE	G2	3.3 V	DATA12	N4	3.3 V	GND2
C5	3.3 V	GND3	G3	3.3 V	DATA15	N15	3.3 V	GND3
C6	3.3 V	IOCS16#	G4	3.3 V	GND3	N16	3.3 V	VDD3
C7	3.3 V	MEMR#	G15	3.3 V	KPORT7	N17	2.5 V	VDDP
C8	3.3 V	ADD22	G16	3.3 V	KPORT2	N18	3.3 V	GND3
C9	3.3 V	ADD20	G17	3.3 V	KPORT0	P1	3.3 V	ADD9
C10	3.3 V	ADD17	G18	3.3 V	KPORT5	P2	3.3 V	ADD0
C11	3.3 V	ADD13	H1	3.3 V	DATA16/GPIO16	P3	3.3 V	ADD2
C12	3.3 V	ADD5	H2	2.5 V	GND2	P4	3.3 V	ADD11
C13	3.3 V	RxD	H3	3.3 V	DATA18/GPIO18	P15	2.5 V	VDD2 (VDDPD)
C14	3.3 V	DTR#/CLKSEL0	H4	3.3 V	VDD3	P16	2.5 V	GNDP

Pin No.	Power System	Pin Name	Pin No.	Power System	Pin Name	Pin No.	Power System	Pin Name
P17	3.3 V	CLKX2	T6	3.3 V	AV$_{DD}$	U13	3.3 V	GPIO9
P18	2.5 V	GND2 (GNDPD)	T7	3.3 V	LCAS#	U14	3.3 V	SYSDIR/GPIO6
R1	3.3 V	ADD1	T8	3.3 V	ROMCS2#	U15	3.3 V	SCAS#/GPIO5
R2	3.3 V	POWER	T9	3.3 V	RD#	U16	3.3 V	GPIO1
R3	3.3 V	GND3	T10	3.3 V	WR#	U17	3.3 V	GPIO2
R4	3.3 V	GND3	T11	3.3 V	DBUS32/GPIO48	U18	3.3 V	CGND
R5	3.3 V	AUDIOIN	T12	3.3 V	DDOUT/GPIO44	V1	3.3 V	V$_{DD}$3
R6	3.3 V	DV$_{DD}$	T13	3.3 V	GPIO11	V2	3.3 V	PIUGND
R7	3.3 V	MRAS2#/ULCAS#	T14	3.3 V	GPIO8	V3	3.3 V	TPX0
R8	3.3 V	MRAS1#	T15	3.3 V	GND3	V4	3.3 V	TPY1
R9	3.3 V	ROMCS1#	T16	3.3 V	GND3	V5	3.3 V	ADIN2
R10	3.3 V	RSTOUT	T17	3.3 V	GPIO0	V6	3.3 V	AUDIOOUT
R11	3.3 V	GND3	T18	3.3 V	RTCX1	V7	3.3 V	MRAS3#/UUCAS#
R12	3.3 V	SMODE1/GPIO49	U1	3.3 V	MPOWER	V8	3.3 V	MRAS0#
R13	3.3 V	DDIN/GPIO45	U2	3.3 V	RTCRST#	V9	3.3 V	ROMCS0#
R14	3.3 V	GPIO12	U3	3.3 V	AGND	V10	3.3 V	V$_{DD}$3
R15	3.3 V	GND3	U4	3.3 V	TPX1	V11	3.3 V	LCDCS#
R16	3.3 V	CV$_{DD}$	U5	3.3 V	TPY0	V12	3.3 V	DCTS#/GPIO47
R17	3.3 V	RTCX2	U6	3.3 V	ADIN1	V13	3.3 V	GPIO14
R18	3.3 V	CLKX1	U7	3.3 V	DGND	V14	3.3 V	GPIO10
T1	3.3 V	POWERON	U8	3.3 V	UCAS#	V15	3.3 V	SPOWER/GPIO7
T2	3.3 V	RSTSW#	U9	3.3 V	ROMCS3#	V16	3.3 V	SRAS#/GPIO4
T3	3.3 V	GND3	U10	3.3 V	LDCRDY	V17	3.3 V	GPIO3
T4	3.3 V	PIUV$_{DD}$	U11	3.3 V	DRTS#/GPIO46	V18	3.3 V	V$_{DD}$3
T5	3.3 V	ADIN0	U12	3.3 V	GPIO13			

Pin Identification

ADD (0:25):	Address Bus
ADIN (0:2):	General Purpose Input for A/D
AFERST#:	AFE Reset
AGND:	GND for A/D
AUDIOIN:	Audio Input
AUDIOOUT:	Audio Output
AV$_{DD}$:	V$_{DD}$ for A/D
BATTINH:	Battery Inhibit
BATTINT:	Battery Interrupt Request
BUSCLK:	System Bus Clock
CGND:	GND for Oscillator
CKE:	Clock Enable
CLKSEL (0:2):	Clock Select
CLKX1:	Clock X1
CLKX2:	Clock X2
CTS#:	Clear to Send
CV$_{DD}$:	V$_{DD}$ for Oscillator
DATA (0:31):	Data Bus
DBUS32:	Data Bus 32
DCD#:	Data Carrier Detect
DCTS#:	Debug Serial Clear to Send
DDIN:	Debug Serial Data Input
DDOUT:	Debug Serial Data Output
DGND:	GND for D/A
DRTS#:	Debug Serial Request to Send
DSR#:	Data Set Ready
DTR#:	Data Terminal Ready
DV$_{DD}$:	V$_{DD}$ for D/A
FIRCLK:	FIR Clock
FIRDIN#:	FIR Data Input
FS:	Frame Synchronization
GND2, GND3:	Ground
GNDP, GNDPD:	Ground for PLL
GPIO (0:49):	General Purpose I/O
HC0:	Hardware Control 0
HLDACK#:	Hold Acknowledge
HLDRQ#:	Hold Request
HSPMCLK:	HSP Codec Master Clock
HSPSCLK:	HSP Codec Serial Clock
ILCSENSE:	Input Loop Current Sensing
IOCHRDY:	I/O Channel Ready
IOCS16#:	I/O Chip Select 16
IOR#:	I/O Read
IOW#:	I/O Write
IRDIN:	IrDA Data Input
IRDOUT#:	IrDA Data Output
IRING:	Input Ring
KPORT (0:7):	Key Code Data Input
KSCAN (0:11):	Key Scan Line
LCAS#:	Lower Column Address Strobe
LCDCS#:	LCD Chip Select
LCDRDY:	LCD Ready
LEDOUT#:	LED Output
MEMCS16#:	Memory Chip Select 16
MEMR#:	Memory Read
MEMW#:	Memory Write
MIPS16EN:	MIPS16 Enable
MPOWER:	Main Power
MRAS (0:3)#:	DRAM Row Address Strobe
MUTE:	Mute
OFFHOOK:	Off Hook
OPD#:	Output Power Down
PIUGND:	GND for Touch Panel Interface
PIUV$_{DD}$:	V$_{DD}$ for Touch Panel Interface
POWER:	Power Switch
POWERON:	Power On State
RD#:	Read
ROMCS (0:3)#:	ROM Chip Select
RSTOUT:	System Bus Reset Output
RSTSW#:	Reset Switch
RTCRST#:	Real-time Clock Reset
RTCX1:	Real-time Clock X1
RTCX2:	Real-time Clock X2
RTS#:	Request to Send
RxD:	Receive Data
SCAS#:	Column Address Strobe for SDRAM/SROM
SCLK:	SDRAM/SROM Clock
SDI:	HSP Serial Data Input
SDO:	HSP Serial Data Output
SEL:	IrDA Module Select
SHB#:	System Hi-Byte Enable
SMODE (1:2):	SDRAM Mode
SPOWER:	SDRAM Power Control
SRAS#:	Row Address Strobe for SDRAM/SROM
SYSDIR:	System Bus Buffer Direction
TELCON:	Telephone Control
TPX (0:1):	Touch Panel X I/O
TPY (0:1):	Touch Panel Y I/O
TxD:	Transmit Data
UCAS#:	Upper Column Address Strobe
ULCAS#:	Lower Byte of Upper Column Address Strobe
UUCAS#:	Upper Byte of Upper Column Address Strobe
V$_{DD}$2, V$_{DD}$3:	Power Supply Voltage
V$_{DD}$P, V$_{DD}$PD:	V$_{DD}$ for PLL
WR#:	Write
ZWS#:	Zero Wait State

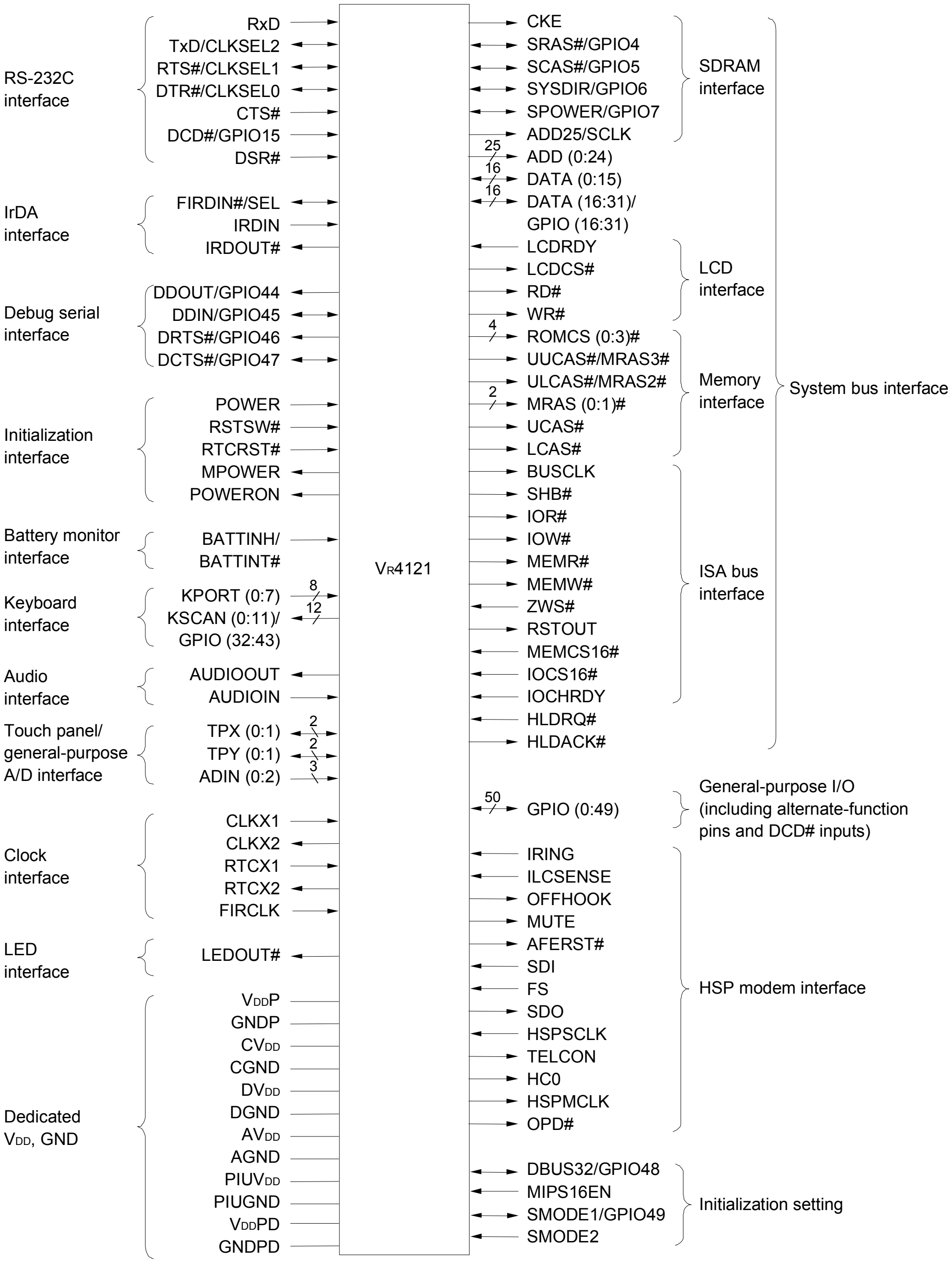
Remark # indicates active low

2.2 Pin Function Description

The functional classification of the VR4121 pins is listed below.

Remark # indicates active low.

Figure 2-1. VR4121 Signal Classification



★ **Caution The number of pins shown in this figure does not match the actual number of pins because the multiplexed pins are shown as interface pins.**

2.2.1 System bus interface signals

These signals are used when the VR4121 is connected to DRAM, SDRAM, ROM, SROM, or an LCD, or other devices in the system through the system bus.

Table 2-1. System Bus Interface Signals (1/3)

Signal	I/O	Function
ADD25/SCLK	O	This function differs depending on how the SMODE(1:2) pin is set. <When SMODE(1:2) = 00> This is a 25-bit address bus. <When SMODE(1:2) ≠ 00> This is the operating clock for SDRAM and SROM.
ADD(0:24)	O	This is a 25-bit address bus. The VR4121 uses this to specify addresses for the SDRAM, SROM, DRAM, ROM, LCD, or system bus (ISA).
DATA(0:15)	I/O	This is a 16-bit data bus. The VR4121 uses this to transmit and receive data with a SDRAM, SROM, DRAM, ROM, LCD, or system bus.
DATA(16:31)/ GPIO(16:31)	I/O	This function differs depending on how the DBUS32 pin is set. <When DBUS32 = 1> This is the high-order 16 bits of the 32-bit data bus. This bus is used for transmitting and receiving data between the VR4121 and the DRAM and ROM. <When DBUS32 = 0> This is a general-purpose I/O (GPIO) port.
★ LCDCS#	O	This is the LCD chip select signal. This signal is active when the VR4121 is performing LCD access and high-speed system bus access using the ADD/DATA bus.
RD#	O	This is active when the VR4121 is reading data from the LCD, SDRAM, SROM, DRAM, or ROM.
WR#	O	This is active when the VR4121 is writing data to the LCD, SDRAM, or DRAM.
LCDRDY	I	This is the LCD ready signal. Set this signal as active when the LCD controller is ready to receive access from the VR4121.
ROMCS(2:3)#	O	The function differs with the setting of the DBUS32 pin. <When DBUS32 = 1> This becomes the chip select signal for the extended ROM, SROM, DRAM, or SDRAM. <When DBUS32 = 0> This is the ROM or SROM chip select signal.
ROMCS(0:1)#	O	This is the ROM or SROM chip select signal.
CKE	O	This is the SDRAM or SROM clock enable signal. When using neither SDRAM nor SROM, connect to GND or leave open.
UUCAS#/ MRAS3#	O	This function differs depending on how the DBUS32 pin is set or types of memory to be accessed. <When DBUS32 = 1> When accessing DRAM (EDO type): This signal is active (UUCAS#) when a valid column address is output via the ADD bus during access of DATA(24:31) in the 32-bit data bus. When accessing SDRAM: This is the I/O buffer control signal (UUDQM#) that is used during access of DATA(24:31) in the 32 bit data bus. During 32-bit access of LCD/high-speed system memory: Byte enable signal that is used during access of DATA(24:31). <When DBUS32 = 0> When accessing DRAM (EDO type): This is the DRAM's RAS signal (MRAS3#). This signal is active when a valid row address is output via the ADD bus for the DRAM connected to the high-order address. When accessing SDRAM: This is the SDRAM's chip select signal (CS3#). This signal is active when a command is issued for the SDRAM connected to the high-order address.

Table 2-1. System Bus Interface Signals (2/3)

Signal	I/O	Function
ULCAS#/ MRAS2#	O	This function differs depending on how the DBUS32 pin is set and type of memory being accessed. <When DBUS32 = 1> When accessing DRAM (EDO type): This signal is active (ULCAS#) when a valid column address is output via the ADD bus during access of DATA(16:23) in the 32-bit data bus. When accessing SRAM: This is the I/O buffer control signal (ULDQM#) that is used during access of DATA(16:23) in the 32-bit data bus. During 32-bit access of LCD/high-speed system memory: Byte enable signal that is used during access of DATA(16:23). <When DBUS32 = 0> When accessing DRAM (EDO type): This is the DRAM's RAS signal (MRAS2#). This signal is active when a valid row address is output via the ADD bus for the DRAM connected to the next highest address after the highest high-order address. When accessing SDRAM: This is the SDRAM's chip select signal (CS2#). This signal is active when a command is issued for the SDRAM connected to the second highest high-order address.
MRAS(0:1)#	O	This function differs depending on the type of memory being accessed. <When accessing DRAM (EDO type)> This is the DRAM's RAS-only signal. <When accessing SDRAM> This is the SDRAM's chip select signal (CS(0:1)#).
UCAS#	O	This function differs depending on the type of memory being accessed. <When accessing DRAM (EDO type)> This is the DRAM's CAS signal. This signal is active when a valid column address is output via the ADD bus during access of DATA(8:15) in the DRAM. <When accessing SDRAM> This is the I/O buffer control signal (UDQM#) that is used during access of DATA(8:15). < During 32-bit access of LCD/high-speed system memory > This is the byte enable signal that is used during access of DATA(8:15). This signal is active when a valid address is output via the ADD bus for access to DATA(8:15) when the size of the access bus to the LCD is 32 bits.
LCAS#	O	This function differs depending on the type of memory being accessed. <When accessing DRAM (EDO type)> This is the DRAM's CAS signal. This signal is active when a valid column address is output via the ADD bus during access of DATA(0:7) in the DRAM. <When accessing SDRAM> This is the I/O buffer control signal (LDQM#) that is used during access of DATA(0:7). < During 32-bit access of LCD/high-speed system memory > This is the byte enable signal that is used during access of DATA(0:7).
BUSCLK	O	This is the system bus clock. It is used to output the clock that is supplied to the controller on the system bus. Its frequency is determined based on the status of the CLKSEL2/T×D, CLKSEL1/RTS#, and CLKSEL0/DTR# pins. Ordinarily, the frequency is 1/4 of the TClock frequency. (**See 2.2.5 RS-232C interface**). The frequency can be changed via the PMU register settings.
SHB#	O	This is the system bus high-byte enable signal. During 16-bit system bus access, this signal is active when the high-order byte is valid on the data bus.
IOR#	O	This is the system bus I/O read signal. It is active when the VR4121 accesses the system bus to read data from an I/O port.
IOW#	O	This is the system bus I/O write signal. It is active when the VR4121 accesses the system bus to write data to an I/O port.
MEMR#	O	This is the system bus memory read signal. It is active when the VR4121 accesses the system bus to read data from memory.

Table 2-1. System Bus Interface Signals (3/3)

Signal	I/O	Function
MEMW#	O	This is the system bus memory write signal. It is active when the VR4121 accesses the system bus to write data to memory.
ZWS#	I	This is the system bus zero wait state signal. Set this signal as active to enable the controller on the system bus to be accessed by the VR4121 without a wait interval.
RSTOUT	O	This is the system bus reset signal. It is active when the VR4121 resets the system bus controller (during bus timeout, manipulation of BCUCNTREG1, and power-down mode).
MEMCS16#	I	This is a dynamic bus sizing request signal. Set this signal as active when system bus memory accesses data in 16-bit width. This signal is invalid when 32-bit width is selected using LCD/high-speed system bus.
★ IOCS16#	I	This is a dynamic bus sizing request signal. Set this signal as active when system bus I/O accesses data in 16-bit width. This signal is invalid when 32-bit width is selected using LCD/high-speed system bus.
IOCHRDY	I	This is the system bus ready signal. Set this signal as active when the system bus controller is ready to be accessed by the VR4121.
HLDRQ#	I	This is a hold request signal for the system bus and DRAM bus that is sent from an external bus master.
HLDACK#	O	This is a hold acknowledge signal for the system bus and DRAM bus that is sent to an external bus master.
SRAS#/GPIO4	I/O	This function differs depending on the type of memory being accessed. <When accessing DRAM (EDO type)> This is a general-purpose I/O port. <When accessing SDRAM> This is the RAS signal for SDRAM and SROM only.
SCAS#/GPIO5	I/O	This function differs depending on the type of memory being accessed. <When accessing DRAM (EDO type)> This is a general-purpose I/O port. <When accessing SDRAM> This is the CAS signal for SDRAM and SROM only.
SYSDIR/GPIO6	I/O	This function differs depending on the type of memory being accessed. <When accessing DRAM (EDO type)> This is a general-purpose I/O port. <When accessing SDRAM> This is the direction control signal for the buffer used to reduce the DATA bus's load.
SPOWER/ GPIO7	I/O	This function differs depending on the type of memory being accessed. <When accessing DRAM (EDO type)> This is a general-purpose I/O port. <When accessing SDRAM> This is the SDRAM's power supply control signal.

2.2.2 Clock interface signals

These signals are used to supply clocks. Table 2-2 lists functions of these signals.

Table 2-2. Clock Interface Signals

Signal	I/O	Function
RTCX1	I	This is the 32.768-kHz oscillator’s input pin. It is connected to one side of a crystal resonator.
RTCX2	O	This is the 32.768-kHz oscillator’s output pin. It is connected to one side of a crystal resonator.
CLKX1	I	This is the 18.432-MHz oscillator’s input pin. It is connected to one side of a crystal resonator.
CLKX2	O	This is the 18.432-MHz oscillator’s output pin. It is connected to one side of a crystal resonator.
FIRCLK	I	This is the 48-MHz clock input pin. Fix this at high level when FIR is not used.

The operating frequency of CPUCORE can be set by the CLKSEL2/TxD, CLKSEL1/RTS#, and CLKSEL0/DTR# signals.

For details of these signals, refer to **2.2.5 RS-232C interface signals**.

2.2.3 Battery monitor interface signals

These signals indicate when an external agent is able to provide enough power for system operations. Table 2-3 describes the functions of these signals.

Table 2-3. Battery Monitor Interface Signals

Signal	I/O	Function
BATTINH/ BATTINT#	I	This function differs depending on how the MPOWER pin is set. <When MPOWER = 0> BATTINH function This signal enables/prohibits activation due to power-on. 1 : Enable activation 0 : Prohibit activation <When MPOWER = 1> BATTINT# function This is an interrupt signal that is output when remaining power is low during normal operations. The external agent checks the remaining battery power. Activate the signal at this pin if voltage sufficient for operations cannot be supplied.

2.2.4 Initialization interface signals

These signals are used when an external agent initializes the processor operation parameters. Table 2-4 describes the functions of these signals.

Table 2-4. Initialization Interface Signals

Signal	I/O	Function
MPOWER	O	This signal indicates the VR4121 is operating. This signal is inactive during Hibernate mode.
POWERON	O	This signal indicates the VR4121 is ready to operate. It becomes active when a power-on factor is detected and becomes inactive when the BATTINH/BATTINT# signal check operation is completed.
POWER	I	This is a VR4121 activation signal.
RSTSW#	I	This is a VR4121 reset signal.
RTCRST#	I	This signal resets RTC. When power is first supplied to a device, the external agent must assert the signal at this pin for about 2 s.

2.2.5 RS-232C interface signals

These signals control data transmission and reception between the VR4121 and an RS-232C controller.

Table 2-5. RS-232C Interface Signals

Signal	I/O	Function
RxD	I	This is a receive data signal. It is used when the RS-232C controller sends serial data to the VR4121.
CTS#	I	This is a transmit enable signal. Assert this signal when the RS-232C controller is ready to receive transmission of serial data.
DCD#/ GPIO15	I	This is a carrier detection signal. Assert this signal when valid serial data is being received. It is also used when detecting a power-on factor for the VR4121. When this pin is not used for DCD# signal, this pin can be used as an interrupt detection function for the GIU unit.
DSR#	I	This is the data set ready signal. Assert this signal when the RS-232C controller is ready to receive/transmit serial data between the controller and the VR4121.
TxD/ CLKSEL2, RTS#/ CLKSEL1, DTR#/ CLKSEL0	I/O	This function differs depending on the operating status. • During normal operation (output) Signals used for serial communication TxD signal : This is a transmit data signal. It is used when the VR4121 sends serial data to the RS-232C controller. RTS# signal : This is a transmit request signal. This signal is asserted when the VR4121 is ready to receive serial data from the RS-232C controller. DTR# signal : This is a terminal equipment ready signal. This signal is asserted when the VR4121 is ready to transmit or receive serial data. • When RTC reset (input) Signals (CLKSEL(2:0)) used to set the CPU core operation frequency, BUSCLK frequency, and internal bus clock frequency. These signals are sampled when the RTCRST# signal changes from low level to high level. The relationships between the CLKSEL pin setting and each clock frequency are shown below.

CLKSEL (2:0)	CPU core operation frequency (PClock)	SDRAM/SROM operation frequency (VTClock)		BUSCLK frequency (When TClock output)	BUSCLK frequency (When 1/4 of TClock)	Interrupt control clock frequency (MasterOut)
		MIN.	MAX.			
111	RFU	RFU	RFU	RFU	RFU	RFU
110	168.5 MHz	28.1 MHz	56.2 MHz	28.1 MHz	7.0 MHz	7.0 MHz
101	147.5 MHz	29.5 MHz	59.0 MHz	29.5 MHz	7.4 MHz	7.4 MHz
100	131.1 MHz	32.8 MHz	65.5 MHz	32.8 MHz	8.2 MHz	8.2 MHz
011	118.0 MHz	29.5 MHz	59.0 MHz	29.5 MHz	7.4 MHz	7.4 MHz
010	98.3 MHz	32.8 MHz	65.5 MHz	32.8 MHz	8.2 MHz	8.2 MHz
001	90.7 MHz	30.2 MHz	60.5 MHz	30.2 MHz	7.6 MHz	7.6 MHz
000	78.6 MHz	26.2 MHz	52.4 MHz	26.2 MHz	6.6 MHz	6.6 MHz

2.2.6 IrDA interface signals

These signals are used to control data transmission and reception between the VR4121 and an IrDA controller.

Table 2-6. IrDA Interface Signals

Signal	I/O	Function
IRDIN	I	This is an IrDA serial data input signal. It is used when the VR4121 sends serial data to the IrDA controller, for both FIR and SIR. If the IrDA controller used is an HP product, however, this signal should be used for only SIR.
FIRDIN#/SEL	I/O	This function differs according to the IrDA controller used (for how to switch a controller, refer to **25.2.13 SIUIRSEL**). • HP's controller FIRDIN#: It is an FIR receive data input signal. • TEMIC's controller SEL: It is an output port for external FIR/SIR switching. • SHARP's controller Use is prohibited.
IRDOUT#	O	This is the IrDA serial data output signal. It is used when the IrDA controller sends serial data from the VR4121.

2.2.7 Debug serial interface signals

These signals are used to control data transmission and reception between the VR4121 and an external debug serial controller.

Table 2-7. Debug Serial Interface Signals

Signal	I/O	Function
DDOUT/ GPIO44	O	This is the debug serial data output signal. It is used when the VR4121 sends serial data to an external debug serial controller. When this pin is not used for the DDOUT signal, it can be used as a general-purpose output port.
DDIN/ GPIO45	I/O	This is the debug serial data input signal. It is used when an external debug serial data controller sends serial data to the VR4121. When this pin is not used for the DDIN signal, it can be used as a general-purpose output port.
DRTS#/ GPIO46	O	This is a transmission request signal. The VR4121 asserts this signal before sending serial data. When this pin is not used for the DRTS# signal, it can be used as a general-purpose output port.
DCTS#/ GPIO47	I/O	This is a transmit acknowledge signal. The VR4121 asserts this signal when it is ready to receive transmitted serial data. When this pin is not used for the DCTS# signal, it can be used as a general-purpose output port.

2.2.8 Keyboard interface signals

These signals are used to control a keyboard circuit to the VR4121.

Table 2-8. Keyboard Interface Signals

Signal	I/O	Function
KPORT(0:7)	I	This is a keyboard scan data input signal. It is used to scan for pressed keys on the keyboard.
KSCAN(0:11)/ GPIO(32:43)	O	These signal are used as keyboard scan data output signals and a general-purpose output port. The scan line is set as active when scanning for pressed keys on the keyboard. Pins that are not used for KSCAN pins can be used as a general-purpose output port.

2.2.9 Audio interface signals

This signal is used to input/output audio signals.

Table 2-9. Audio Interface Signals

Signal	I/O	Function
AUDIOIN	I	This pin is the audio input pin.
AUDIOOUT	O	This is an audio output signal. Analog signals that have been converted via the on-chip 10-bit D/A converter are output.

2.2.10 Touch panel/general purpose A/D interface signals

These are the signals to the on-chip A/D converter of the VR4121. Four of these signals are used for a touch panel, and the remaining three are used as general-purpose input pins.

Table 2-10. Touch Panel/General Purpose A/D Interface Signals

Signal	I/O	Function
TPX(0:1)	I/O	This is an I/O signal that is used for the touch panel. It uses the voltage applied to the X coordinate and the voltage input to the Y coordinate to detect which coordinates on the touch panel are being pressed.
TPY(0:1)	I/O	This is an I/O signal that is used for the touch panel. It uses the voltage applied to the Y coordinate and the voltage input to the X coordinate to detect which coordinates on the touch panel are being pressed.
ADIN(0:2)	I	This is a general-purpose A/D input signal.

2.2.11 General-purpose I/O Signals

These are general-purpose I/O pins of the VR4121. Normally, 33 of the 49 general-purpose I/O pins are used as alternate-function pins.

Table 2-11. General-purpose I/O Signals

Signal	I/O	Function
GPIO(0:3)	I/O	These are maskable power-on factors. After start-up, they are used as ordinary general-purpose I/O pins.
GPIO4/SRAS#	I/O	See **2.2.1 System bus interface signals**.
GPIO5/SCAS#	I/O	See **2.2.1 System bus interface signals**.
GPIO6/SYSDIR	I/O	See **2.2.1 System bus interface signals**.
GPIO7/SPOWER	I/O	See **2.2.1 System bus interface signals**.
GPIO8	I/O	These are general-purpose I/O pins.
GPIO(9:12)	I/O	These are maskable power-on factors. After start-up, they are used as ordinary general-purpose I/O pins.
GPIO(13:14)	I/O	These are general-purpose I/O pins.
DATA(16:31)/GPIO(16:31)	I/O	See **2.2.1 System bus interface signals**.
KSCAN(0:11)/GPIO(32:43)	O	See **2.2.8 Keyboard interface signals**.
DDOUT/GPIO44	O	See **2.2.7 Debug serial interface signals**.
DDIN/GPIO45	I/O	See **2.2.7 Debug serial interface signals**.
DRTS#/GPIO46	O	See **2.2.7 Debug serial interface signals**.
DCTS#/GPIO47	I/O	See **2.2.7 Debug serial interface signals**.
DBUS32/GPIO48	I/O	See **2.2.14 Initial setting signals**.
GPIO49/SMODE1	O	See **2.2.14 Initial setting signals**.

2.2.12 HSP MODEM interface signals

Table 2-12. HSP MODEM Interface Signals

Signal	I/O	Function
IRING	I	RING signal detect signal. This pin becomes active when the RING signal is detected.
ILCSENSE	I	Handset detect signal
OFFHOOK	O	On-hook relay control signal
MUTE	O	Modem speaker mute control signal
AFERST#	O	CODEC reset signal
SDI	I	Serial input signal from CODEC
FS	I	Frame synchronization signal from CODEC
SDO	O	Serial output signal to CODEC
HSPSCLK	I	Operation clock input of modem interface block for CODEC
TELCON	O	Handset relay control signal
HC0	O	CODEC control signal
HSPMCLK	O	Clock output to CODEC
OPD#	O	Use this pin for controlling power of CODEC and DAA. This signal is set as active when the power supply of CODEC and DAA is ON.

2.2.13 LED interface signal

Table 2-13. LED Interface Signal

Signal	I/O	Function
LEDOUT#	O	This is an output signal for lighting LEDs.

2.2.14 Initial setting signals

Table 2-14. Initial Setting Signals

Signal Name	I/O	Function
DBUS32/ GPIO48	I/O	The function differs depending on the operating status. <During normal operation (output)> This can be used as a general-purpose output port. <After an RTC reset (input)> ★ This is the switching signal for the data bus width. This signal is sampled at 1RTC clock cycle after the RTCRST# signal changes from low level to high level. 1: The data bus has a 32-bit width. 0: The data bus has a 16-bit width.
SMODE1/ GPIO49	I/O	The function differs depending on the operating status. <During normal operation (output)> This can be used as a general-purpose output port. < After an RTC reset (input)> This is a switching signal for the memory being used. It is used in combination with the ★ SMODE2 signal. This signal is sampled at 1RTC clock cycle after the RTCRST# signal changes from low level to high level.
SMODE2	I	★ This a switching signal for the memory being used. It is used in combination with the SMODE1 signal. This signal is sampled when the RTCRST# signal changes from low level to high level. The relation between the SMODE pin and the memory being used is shown below. SMODE(2:1) — Used Memory 11 — ROM: SROM RAM: SDRAM 10 — ROM: Flash memory, PageROM, ordinary ROM RAM: SDRAM 01 — ROM (boot bank): Flash memory, PageROM, ordinary ROM ROM (except boot bank): SROM RAM: SDRAM 00 — ROM: Flash memory, PageROM, ordinary ROM RAM: DRAM (EDO type)
MIPS16EN	I	★ This pin enables the use of MIPS16 instructions. This signal is sampled at 1RTC clock cycle after the RTCRST# signal changes from low level to high level. 1: Enables the use of MIPS16 instructions. 0: Disables the use of MIPS16 instructions.

2.2.15 Dedicated V_{DD} and GND signals

Table 2-15. Dedicated V_{DD} and GND Signals

Signal Name	Power-Supply System	Function
V_{DD}P	2.5 V	Dedicated V_{DD} for the PLL analog unit
GNDP	2.5 V	Dedicated GND for the PLL analog unit
V_{DD}PD	2.5 V	Dedicated V_{DD} for the PLL digital unit. Its function is identical to V_{DD}2.
GNDPD	2.5 V	Dedicated GND for the PLL digital unit. Its function is identical to GND2.
CV_{DD}	3.3 V	Dedicated V_{DD} for the oscillator
CGND	3.3 V	Dedicated GND for the oscillator
DV_{DD}	3.3 V	Dedicated V_{DD} for the D/A converter. The voltage applied to this pin becomes the maximum of the analog output of AUDIOOUT.
DGND	3.3 V	Dedicated GND for D/A converter. The voltage applied to this pin becomes the minimum of the analog output of AUDIOOUT.
AV_{DD}	3.3 V	Dedicated V_{DD} for the A/D converter. The voltage applied to this pin becomes the maximum voltage that can be detected by the A/D interface signals (8 lines).
AGND	3.3 V	Dedicated GND for the A/D converter. The voltage applied to this pin becomes the minimum voltage that can be detected by the A/D interface signals (8 lines).
PIUV_{DD}	3.3 V	Dedicated V_{DD} for touch-sensitive panel interface
PIUGND	3.3 V	Dedicated GND for touch-sensitive panel interface
V_{DD}2	2.5 V	Normal 2.5-V system V_{DD}
GND2	2.5 V	Normal 2.5-V system GND
V_{DD}3	3.3 V	Normal 3.3-V system V_{DD}
GND3	3.3 V	Normal 3.3-V system GND

Caution The V_R4121 has two types of power supplies. There are no restrictions as to the sequence in which these power supplies are applied. However, do not apply one type of power for more than one second while the other power supply is not applied.

2.3 Pin Status

2.3.1 Pin status upon specific states

Table 2-16 lists the pin states after the VR4121 is reset or when it is in the power mode.

Table 2-16. Pin Status upon Specific States (1/4)

Signal Name	After Reset by the RTCRST	After Reset by the Deadman's Switch or RSTSW	In the Suspend Mode	In the Hibernate Mode or Shut Down by the HAL Timer	During a Bus Hold
ADD25/SCLK	0	**Note 1**	**Note 2**	0	Hi-Z
ADD(0:24)	0	0	**Note 2**	0	Hi-Z
DATA(0:15)	0	0	**Note 2**	0	Hi-Z
DATA(16:31)/ GPIO(16:31)	0/ Hi-Z	0/ Hi-Z	**Note 2**	0/ Hi-Z	Hi-Z/ **Note 2**
LCDCS#	Hi-Z	1	1	Hi-Z	1
RD#	Hi-Z	1	1	Hi-Z	Hi-Z
WR#	Hi-Z	1	1	Hi-Z	Hi-Z
LCDRDY	–	–	–	–	–
ROMCS#(2:3)	Hi-Z	**Note 3**	**Note 3**	**Note 3**	**Note 3**
ROMCS#(0:1)	Hi-Z	1	1	Hi-Z	1
UUCAS#/MRAS#3	**Note 4**	**Note 5**	**Note 6**	0	Hi-Z
ULCAS#/MRAS#2	**Note 4**	**Note 5**	**Note 6**	0	Hi-Z
MRAS#(0:1)	Hi-Z	1	1	1	Hi-Z
UCAS#	0	**Note 7**	0	0	Hi-Z
LCAS#	0	**Note 7**	0	0	Hi-Z
BUSCLK	0	0	**Note 2**	0	**Note 8**

Notes 1. This differs depending on the setting of the SCLK bit in the SDRAMMODREG register.
When SCLK bit has a value of "1": outputs clock.
When SCLK bit has a value of "0": low level is output.

2. Maintains the state of the previous Full-speed Mode.

3. When used as the chip select for the ROM or extended ROM, this is the same as ROMCS(0:1)#.
When used as the RAS for the extended DRAM, this is the same as MRAS(0:1)#.

4. When DBUS32 = 1, this becomes the high impedance state.
When DBUS32 = 0, the high level is output.

5. When DBUS32 = 1: See Note 7 below.
When DBUS32 = 0: high level is output.

6. When DBUS32 = 1: low level is output.
When DBUS32 = 0: high level is output.

7. Reset by the RSTSW# signal: The pin outputs a low level. (Self refresh)
Reset by the Deadman's switch: The pin outputs a high level.

8. Bus hold from the Suspend Mode: The state of the previous Full-speed Mode is maintained.
Bus hold from Full-speed Mode or Standby Mode: Outputs clocks.

Remark 0: low level, 1: high level, Hi-Z: high impedance

Table 2-16. Pin Status upon Specific States (2/4)

Signal Name	After Reset by the RTCRST	After Reset by the Deadman’s Switch or RSTSW	In the Suspend Mode	In the Hibernate Mode or Shut Down by the HAL Timer	During a Bus Hold
SHB#	Hi-Z	1	1	Hi-Z	Hi-Z
IOR#	Hi-Z	1	1	Hi-Z	Hi-Z
IOW#	Hi-Z	1	1	Hi-Z	Hi-Z
MEMR#	Hi-Z	1	1	Hi-Z	Hi-Z
MEMW#	Hi-Z	1	1	Hi-Z	Hi-Z
ZWS#	–	–	–	–	–
RSTOUT	Hi-Z	1	0	Hi-Z	**Note 1**
IOCS16#	–	–	–	–	–
MEMCS16#	–	–	–	–	–
IOCHRDY	–	–	–	–	–
HLDRQ#	–	–	–	–	–
HLDACK#	Hi-Z	1	**Note 1**	Hi-Z	**Note 1**
★ CKE	0	**Note 2**	**Note 3**	**Note 3**	Hi-Z
RTCX1	–	–	–	–	–
RTCX2	–	–	–	–	–
CLKX1	–	–	–	–	–
CLKX2	–	–	–	–	–
FIRCLK	–	–	–	–	–
BATTINH/ BATTINT#	–	–	–	–	–
MPOWER	0	1	1	0	1
POWERON	0	0	0	0	0
POWER	–	–	–	–	–
RSTSW#	–	–	–	–	–
RTCRST#	–	–	–	–	–
RxD	–	–	–	–	–
TxD/CLKSEL2	Hi-Z	1	1	1	**Note 1**
RTS#/CLKSEL1	Hi-Z	1	1	1	**Note 1**
CTS#	–	–	–	–	–
DCD#/GPIO15	–	–	–	–	–
DTR#/CLKSEL0	Hi-Z	1	1	1	**Note 1**
DSR#	–	–	–	–	–
IRDIN#	–	–	–	–	–
IRDOUT#	0	0	0	0	**Note 1**

Notes 1. Normal operation proceeds.

2. This differs depending on the setting of the SCLK bit in the SDRAMMODREG register.
When SCLK bit has a value of "1": outputs clock.
When SCLK bit has a value of "0": low level is output.

3. Maintains the state of the previous Full-speed Mode.

Remark 0: low level, 1: high level, Hi-Z: high impedance

Table 2-16. Pin Status upon Specific States (3/4)

Signal Name	After Reset by the RTCRST	After Reset by the Deadman's Switch or RSTSW	In the Suspend Mode	In the Hibernate Mode or Shut Down by the HAL Timer	During a Bus Hold
FIRDIN#/SEL	Hi-Z	Hi-Z	**Note 2**	Hi-Z	**Note 2**
DDIN#/ GPIO45[Note 1]	–/ Hi-Z	–/ **Note 2**	–/ **Note 2**	–/ **Note 2**	– / **Note 2**
DDOUT#/ GPIO44[Note 1]	1/ 1	1/ **Note 2**	1/ **Note 2**	1/ **Note 2**	1/ **Note 2**
DRTS#/ GPIO46[Note 1]	1/ 1	1/ **Note 2**	1/ **Note 2**	1/ **Note 2**	1/ **Note 2**
DCTS#/ GPIO47[Note 1]	–/ Hi-Z	–/ **Note 2**	– / **Note 2**	– / **Note 2**	–/ **Note 2**
KPORT(0:7)	–	–	–	–	–
KSCAN(0:11)/ GPIO(32:43)[Note 1]	Hi-Z/ Hi-Z	Hi-Z/ **Note 2**	**Note 2**/ **Note 2**	Hi-Z/ **Note 2**	**Note 3**
AUDIOOUT	0	0	**Note 2**	0	**Note 3**
TPX(0:1)	1	1	**Note 2**	1	**Note 3**
TPY(0:1)	Hi-Z	Hi-Z	**Note 2**	Hi-Z	**Note 3**
ADIN(0:2)	–	–	–	–	–
AUDIOIN	–	–	–	–	–
GPIO(0:3)	Hi-Z	Hi-Z	**Note 2**	Hi-Z[Note 4]	**Note 3**
★ SRAS#/GPIO4	Hi-Z	**Note 5**/ Hi-Z	0/ **Note 2**	0/ Hi-Z	Hi-Z/ **Note 3**
★ SCAS#/GPIO5	Hi-Z	**Note 5**/ Hi-Z	0/ **Note 2**	0/ Hi-Z	Hi-Z/ **Note 3**
★ SYSDIR/GPIO6	0/ Hi-Z	0/ Hi-Z	0/ **Note 2**	0/ Hi-Z	Hi-Z/ **Note 3**
★ SPOWER/GPIO7	0/ Hi-Z	1/ Hi-Z	1/ **Note 2**	1/ Hi-Z	1/ **Note 3**
GPIO(8:14)	Hi-Z	Hi-Z	**Note 2**	Hi-Z[Note 4]	**Note 3**

Notes 1. Software can switch the function pin and the output port.

2. The state of the previous Full-speed Mode is maintained.

3. Normal operation proceeds.

4. During hibernate mode, the pull-up/pull-down setting is retained.

5. When reset by RSTSW# signal: low level output (self refresh)
When reset by deadman's switch: high level output

Remark 0: low level, 1: high level, Hi-Z: high impedance

Table 2-16. Pin Status upon Specific States (4/4)

Signal Name	After Reset by the RTCRST	After Reset by the Deadman's Switch or RSTSW	In the Suspend Mode	In the Hibernate Mode or Shut Down by the HAL Timer	During a Bus Hold
IRING	–	–	–	–	–
ILCSENSE	–	–	–	–	–
OFFHOOK Note 1	Hi-Z	Hi-Z	**Note 2**	Hi-Z	**Note 2**
MUTE Note 1	Hi-Z	Hi-Z	**Note 2**	Hi-Z	**Note 2**
AFERST Note 1	0	0	**Note 2**	0	**Note 2**
SDI	–	–	–	–	–
FS	–	–	–	–	–
SDO	0	0	**Note 2**	0	**Note 2**
HSPSCLK	–	–	–	–	–
TELCON Note 1	Hi-Z	Hi-Z	**Note 2**	Hi-Z	**Note 2**
HC0 Note 1	0	0	**Note 2**	0	**Note 2**
HSPMCLK Note 1	0	0	**Note 2**	0	**Note 2**
OPD#	0	0	**Note 2**	0	**Note 2**
LEDOUT#	1	**Note 3**	**Note 3**	**Note 3**	**Note 3**
DBUS32/ GPIO48 Note 4	Hi-Z/ Hi-Z	Hi-Z/ **Note 2**	**Note 2**/ **Note 2**	Hi-Z/ **Note 2**	**Note 2**/ **Note 2**
MIPS16EN	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z
SMODE1/ GPIO49 Note 4	**Note 5**	**Note 2**	**Note 2**	**Note 2**	**Note 2**
SMODE2	–	–	–	–	–

Notes 1. When initializing, always set BSC bit to 1 in the HSPINT register (0x0C00 0020).

2. The state of the previous Full-speed Mode is maintained.

3. Normal operation proceeds.

4. After the RTC reset is released, this functions as an output port.

5. Input state. Input low level.

Remark 0: low level, 1: high level, Hi-Z: high impedance

2.3.2 Connection of unused pins and pin I/O circuits

Table 2-17. Connection of Unused Pins and Pin I/O Circuit Type (1/3)

Signal	Internal Processing	External Processing	Drive Capability	I/O Circuit Type
ADD25/SCLK	Slew rate buffer	–	120 pF	A
ADD(0:24)	Slew rate buffer	–	120 pF	A
DATA(0:15)	–	–	40 pF	A
DATA(16:31)/ GPIO(16:31)	–	**Note 1**	40 pF	A
LCDCS#	Slew rate buffer	–	40 pF	A
RD#	Slew rate buffer	**Note 2**	120 pF	A
WR#	Slew rate buffer	**Note 2**	120 pF	A
LCDRDY	–	**Note 3**	–	A
ROMCS(2:3)#	Slew rate buffer	**Note 4**	40 pF	A
ROMCS(0:1)#	Slew rate buffer	–	40 pF	A
UUCAS#/MRAS3#	Slew rate buffer	**Note 2**	120 pF	A
ULCAS#/MRAS2#	Slew rate buffer	**Note 2**	120 pF	A
MRAS(0:1)#	Slew rate buffer	**Note 2**	40 pF	A
UCAS#	Slew rate buffer	**Note 2**	120 pF	A
LCAS#	Slew rate buffer	**Note 2**	120 pF	A
BUSCLK	Slew rate buffer	–	40 pF	A
SHB#	Slew rate buffer	**Note 2**	40 pF	A
IOR#	Slew rate buffer	**Note 2**	40 pF	A
IOW#	Slew rate buffer	**Note 2**	40 pF	A
MEMR#	Slew rate buffer	**Note 2**	40 pF	A
MEMW#	Slew rate buffer	**Note 2**	40 pF	A
ZWS#	**Note 5**	**Note 3**	–	A
RSTOUT	Slew rate buffer	Pull up	40 pF	A
IOCS16#	**Note 5**	**Note 3**	–	A
MEMCS16#	**Note 5**	**Note 3**	–	A
IOCHRDY	**Note 5**	**Note 3**	–	A

Notes **1.** Pins DATA(16:31) in the VR4121 function as GPIO(16:31) when using the 16-bit data bus. When using these pins as GPIO(16:31), pull them up or pull down so as not to input an intermediate-level signal.

2. When the bus hold function is used, pull-ups are recommended outside the VR4121.

3. Do not input an intermediate-level signal.

4. When used as the RAS signal of extended DRAM, external pull-up is recommended for the VR4121.

5. When the MPOWER pin outputs the low-level, intermediate-level input is enabled.

Table 2-17. Connection of Unused Pins and Pin I/O Circuit Type (2/3)

Signal	Internal Processing	External Processing	Drive Capability	I/O Circuit Type
HLDRQ#	**Note 1**	**Note 2**	–	A
HLDACK#	Slew rate buffer	–	40 pF	A
CKE	–	–	120 pF	A
RTCX1	–	Resonator	–	–
RTCX2	–	Resonator	–	–
CLKX1	–	Resonator	–	–
CLKX2	–	Resonator	–	–
FIRCLK	–	**Note 3**	–	A
BATTINH/ BATTINT#	Schmitt	–	–	B
MPOWER	–	–	40 pF	A
POWERON	–	–	40 pF	A
POWER	Schmitt	–	–	B
RSTSW#	Schmitt	–	–	B
RTCRST#	Schmitt	–	–	B
RxD	–	–	–	A
TxD/CLKSEL2	–	Pull up/ Pull down	40 pF	A
RTS#/CLKSEL1	–	Pull up/ Pull down	40 pF	A
CTS#	–	–	–	A
DCD#/GPIO15	Schmitt	Pull up	–	B
DTR#/CLKSEL0	–	Pull up/ Pull down	40 pF	A
DSR#	–	–	–	A
IRDIN	–	Pull up	–	A
IRDOUT#	–	–	40 pF	A
FIRDIN#/SEL	–	Pull up/ Pull down	40 pF	A
DDIN#/ GPIO45	–	–	40 pF	A
DDOUT/ GPIO44	–	–	40 pF	A
DRTS#/ GPIO46	–	–	40 pF	A
DCTS#/ GPIO47	–	–	40 pF	A

Notes 1. Intermediate-level input is enabled when the MPOWER pin is set for low-level output.

2. When the bus hold function is used : Pull up.
When the bus hold function is not used : Connect to V_{DD}.

3. When FIR unit is used : Attach an oscillator.
When FIR unit is not used : Connect to V_{DD}.

Table 2-17. Connection of Unused Pins and Pin I/O Circuit Type (3/3)

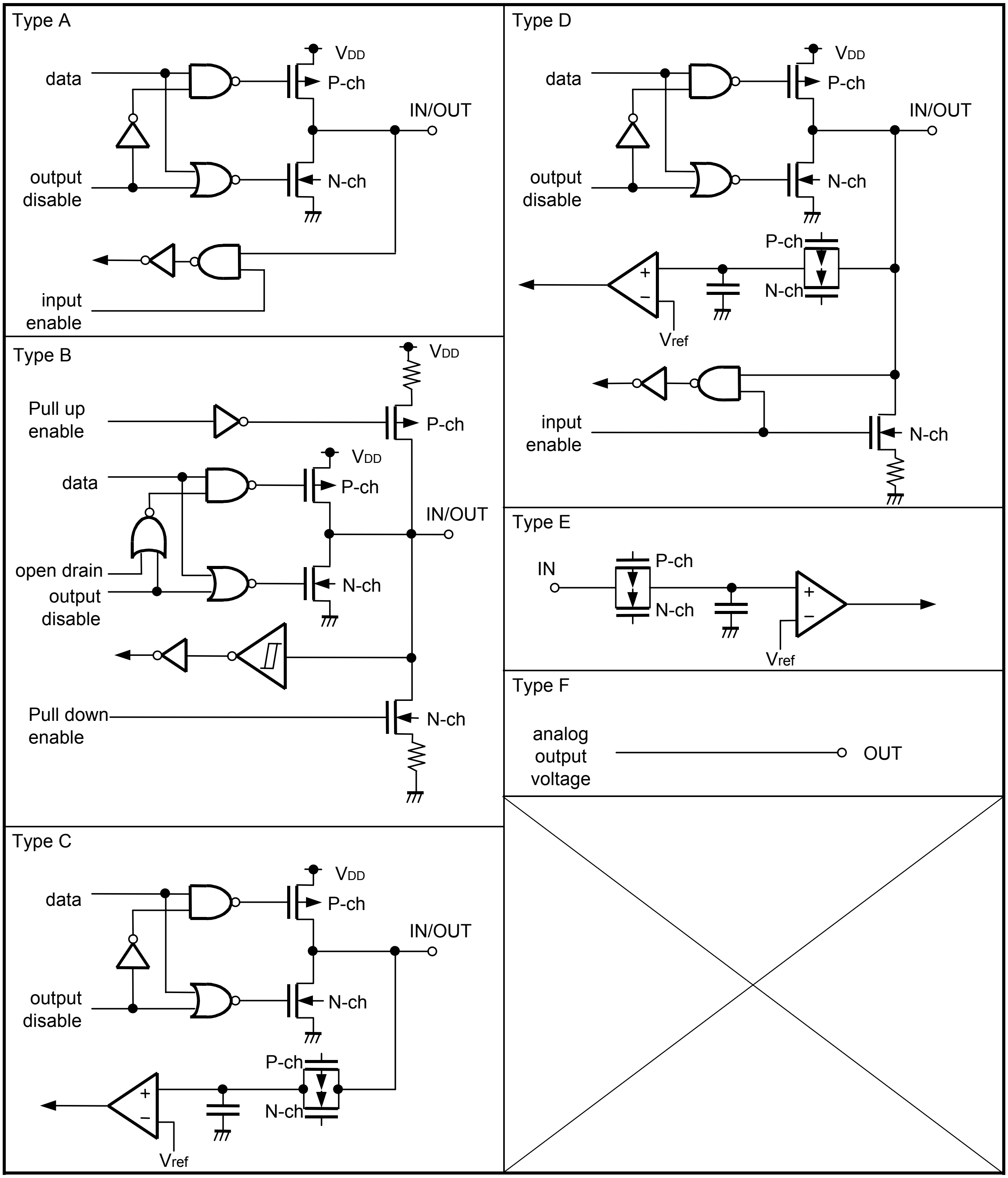
Signal	Internal Processing	External Processing	Drive Capability	I/O Circuit Type
KPORT(0:7)	Schmitt, Pull down	–	–	B
KSCAN(0:11)/ GPIO(32:43)	–	–	40 pF	A
AUDIOOUT	–	**Note 1**	–	F
TPX(0:1)	–	–	120 pF or more	C
TPY1	–	–	120 pF or more	D
TPY0	–	–	120 pF or more	C
ADIN(0:2)	–	–	–	E
AUDIOIN	–	–	–	E
GPIO(0:3)	Schmitt**Note 2**	**Note 2**	40 pF	B
SRAS#/GPIO4	Schmitt**Note 2**	**Note 2**	40 pF	B
SCAS#/GPIO5	Schmitt**Note 2**	**Note 2**	40 pF	B
SYSDIR/GPIO6	Schmitt**Note 2**	**Note 2**	40 pF	B
SPOWER/GPIO7	Schmitt**Note 2**	**Note 2**	40 pF	B
GPIO(8:14)	Schmitt**Note 2**	**Note 2**	40 pF	B
IRING	Schmitt	Pull down	–	B
ILCSENSE	–	Pull down	–	A
OFFHOOK	–	–	40 pF	A
MUTE	–	–	40 pF	A
AFERST#	–	–	40 pF	A
SDI	–	Pull up/Pull down	–	A
FS	–	Pull up/Pull down	–	A
SDO	–	–	40 pF	A
HSPSCLK	–	–	–	A
TELCON	–	–	40 pF	A
HC0	–	–	40 pF	A
HSPMCLK	–	–	40 pF	A
OPD#	–	–	40 pF	A
LEDOUT#	–	–	40 pF	A
DBUS32/ GPIO48	–	Pull up/Pull down	40 pF	A
MIPS16EN	–	Pull up/Pull down	40 pF	A
SMODE1/GPIO49	–	Pull up/Pull down	–	A
SMODE2	–	Pull up/Pull down	–	A

Notes 1. Connect an operation amplifier which has high-impedance input characteristics, since the output level of AUDIOOUT pin varies according to the external impedance.

2. If internal pull-up or pull-down resistors are used in GPIO(0:14), software can switch between pull up, pull down, and open.

If an internal pull-up or pull-down resistor is not used, then provide an external pull-up or pull-down resistor.

2.3.3 Pin I/O circuits



[MEMO]

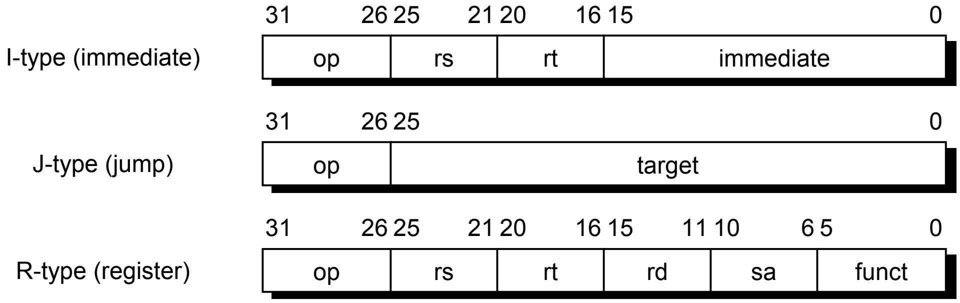
CHAPTER 3 MIPS III INSTRUCTION SET SUMMARY

This chapter is an overview of the MIPS III ISA central processing unit (CPU) instruction set; refer to **CHAPTER 28 MIPS III INSTRUCTION SET DETAILS** for detailed descriptions of individual CPU instructions.

3.1 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 3-1. 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 3-1. 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

(1) Support of the MIPS ISA

The VR4121 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.

3.2 Instruction Classes

The CPU instructions are classified into five classes.

3.2.1 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 **CHAPTER 5 VR4121 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 3-1. Number of Delay Slot Cycles Necessary for Load and Store Instructions

Instruction	Necessary Number of PCycles
Load	1
Store	1

(3) Defining Access Types

Access type indicates the size of a VR4121 processor data item to be loaded or stored, set by the load or store instruction opcode.

Access types and accessed byte are shown in Table 3-2.

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, define the bytes accessed within the addressed doubleword (shown in Table 3-2). Only the combinations shown in Table 3-2 are permissible; other combinations cause address error exceptions.

Tables 3-2 and 3-3 list the ISA-defined load/store instructions and extended-ISA instructions, respectively.

Figure 3-2. 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 3-2. 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.
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.

Table 3-3. Load/Store Instruction (Extended ISA)

Instruction	Format and Description op base rt offset
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.

3.2.2 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 3-4. 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 3-5. 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 3-6. 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 3-7. 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 3-8. 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 3-9. 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 3-10. 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 3-11. Multiply/Divide Instructions (Extended ISA) (1/2)

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.

Table 3-11. Multiply/Divide Instructions (Extended ISA) (2/2)

Instruction	Format and Description op rs rt rd sa funct
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. The same data as that stored in register LO 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 3-12 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 3-12. 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

3.2.3 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 3-13. 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 **CHAPTER 28 MIPS III INSTRUCTION SET DETAILS**.

(2) Overview of branch instructions

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

Tables 3-14 through 3-16 show the lists of Jump, Branch, and Expanded ISA instructions, respectively.

Table 3-14. 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 3-15 and 3-16.

(1) 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.

(2) Operation when unbranched (Table 3-16)

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 3-15 through 3-19.

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

Table 3-15. 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 3-16. 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.				

3.2.4 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 3-17. 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 3-18. 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 3-18. 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.

3.2.5 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 3-19. 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 3-19. 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.

CHAPTER 4 MIPS16 INSTRUCTION SET

4.1 Outline

If the MIPS16 ASE (Application-Specific Extension), which is an expanded function for MIPS ISA (Instruction Set Architecture), is used, system costs can be considerably reduced by lowering the memory capacity requirement of embedded hardware. MIPS16 is an instruction set that uses the 16-bit instruction length, and is compatible with MIPS I, II, III, IV, and V[Note] instruction sets in any combination. Moreover, 32-bit instruction length binary data can be executed with the VR4121.

Note The VR4100 Series currently supports the MIPS I, II, and III instruction sets.

4.2 Features

- 16-bit length instruction format
- Reduces memory capacity requirements to lower overall system cost
- MIPS16 instructions can be used with MIPS instruction binary
- Compatibility with MIPS I, II, III, IV, and V instruction sets
- Used with switching between MIPS16 instruction length mode and 32-bit MIPS instruction length mode.
- Supports 8-bit, 16-bit, 32-bit, and 64-bit data formats
- Provides 8 general registers and special registers
- Improved code generation efficiency using special 16-bit dedicated instructions

4.3 Register Set

Tables 4-1 and 4-2 show the MIPS16 register sets. These register sets form part of the register sets that can be accessed in 32-bit instruction length mode. MIPS16 ASE can directly access 8 of the 32 registers that can be used in the 32-bit instruction length mode.

In addition to these 8 general registers, the special instructions of MIPS16 ASE reference the stack pointer register (sp), return address register (ra), condition code register (t8), and program counter (pc). sp and ra are mapped by fixing to the general registers in the 32-bit instruction length mode.

MIPS16 has 2 move instructions that are used in addressing 32 general registers.

Table 4-1. General Registers

MIPS16 Register Encoding	32-Bit MIPS Register Encoding	Symbol	Comment
0	16	s0	General register
1	17	s1	General register
2	2	v0	General register
3	3	v1	General register
4	4	a0	General register
5	5	a1	General register
6	6	a2	General register
7	7	a3	General register
N/A	24	t8	MIPS16 condition code register. BTEQZ, BTNEZ, CMP, CMPI, SLT, SLTU, SLTI, and SLTIU instructions are implicitly referenced.
N/A	29	sp	Stack pointer register
N/A	31	ra	Return address register

Notes 1. The symbols are the general assembler symbols.

2. The MIPS register encoding numbers 0 to 7 correspond to the MIPS16 binary encoding of the registers, and are used to show the relationship between this encoding and the MIPS registers. The numbers 0 to 7 are not used to reference registers, except within binary MIPS16 instructions. Registers are referenced from the assembler using the MIPS name ($16, $17, $2, etc.) or the symbol name (s0, s1, v0, etc.). For example, when register number 17 is accessed with the register file, the programmer references either $17 or s1 even if the MIPS16 encoding of this register is 001.

3. The general registers not shown in this table cannot be accessed with a MIPS16 instruction set other than the Move instruction. The Move instruction of MIPS16 can access all 32 general registers.

4. To reference the MIPS16 condition code registers with this manual, either T, t8, or $24 has to be used, depending on the case. These three names reference the same physical register.

Table 4-2. Special Registers

Symbol	Description
PC	Program counter. The PC-relative Add instruction and Load instruction can access this register.
HI	The upper word of the multiply or divide result is inserted
LO	The lower word of the multiply or divide result is inserted

4.4 ISA Mode

MIPS16 ASE supports procedure calling, and returns from the MIPS16 instruction length mode or the 32-bit MIPS instruction length mode to the MIPS16 instruction length mode or the 32-bit MIPS instruction length mode.

- The JAL instruction supports calling to the same ISA.
- The JALX instruction supports calling that inverses ISA.
- The JALR instruction supports calling to either ISA.
- The JR instruction supports also returning to either ISA.

MIPS16 ASE also supports a return operation from exception processing.

- The ERET instruction, which is defined only in 32-bit instruction length mode, supports returning to ISA when an exception has not occurred.

The ISA mode bit defines the instruction length mode to be executed. If the ISA mode bit is 0, the processor executes only 32-bit MIPS instructions. If the ISA mode bit is 1, the processor executes only MIPS16 instructions.

4.4.1 Changing ISA mode bit by software

Only the JALX, JR, and JALR instructions change the ISA mode bit between the MIPS16 instruction mode and the 32-bit instruction length mode. The ISA mode bit cannot be directly overwritten by software. The JALX changes the ISA mode bit to select another ISA mode. The JR instruction and JALR instruction load the ISA mode bit from bit 0 of the general register that holds the target address. Bit 0 is not a part of the target address. Bit 0 of the target address is always 0, and no address exception is generated.

Moreover, the JAL, JALR, and JALX instructions save the ISA mode bit to bit 0 of the general register that acquires the return address. The contents of this general register are later used by the JR and JALR instruction for return and restoration of the ISA mode.

4.4.2 Changing ISA mode bit by exception

Even if an exception occurs, the ISA mode changes. When an exception occurs, the ISA mode bit is cleared to 0 so that the exception is serviced with 32-bit code. Then the ISA mode status before the exception occurred is saved to the least significant bit of the EPC register or the error EPC register. During return from an exception, the ISA mode before the exception occurred is returned to by executing the JR or ERET instruction with the contents of this register. Moreover, the ISA mode bit is cleared to 0 after cold reset and soft reset of the CPU core, and the 32-bit instruction length mode returns to its initial state.

4.4.3 Enabling change ISA mode bit

Changing the ISA mode bit is valid only MIPS16EN is set to active when the RTCRST is selected, and the MIPS16 instruction mode is enabled. The operation of the JALX, JALR, JR, and ERET instructions in the 32-bit instruction mode, differs depending on whether the MIPS16 instruction mode is enabled or prohibited. If the MIPS16 instruction mode is prohibited, the JALX instruction generates a reserved instruction exception. The JR and JALR instructions generate an address exception when bit 0 of the source register is 1. The ERET instruction generates an address exception when bit 0 of the EPC or error EPC register is 1. If the MIPS16 instruction mode is enabled, the JALX instruction executes JAL, and the ISA mode bit is inverted. The JR and JALR instructions load the ISA mode from bit 0 of the source register. The ERET instruction loads the ISA mode from bit 0 of the EPC or error EPC register. Bit 0 of the target address is always 0, and no address exception is generated even when bit 0 of the source register is 1.

4.5 Types of Instructions

This section describes the different types of instructions, and indicates the MIPS16 instructions included in each group.

Instructions are divided into the following types.

Load and Store instructions: Move data between memory and the general registers.

Computational instructions: Perform arithmetic operations, logical operations, and shift operations on values in registers.

Jump and Branch instructions: Change the control flow of a program.

Special instructions: Break instructions and Extend instructions. Break transfers control to an exception handler. Extend enlarges the immediate field of the next instruction. Instructions that can be extended with Extend are indicated as **Note 1** in **Table 4-3 MIPS16 Instruction Set Outline**.

Table 4-3. MIPS16 Instruction Set Outline

Op	Description	Op	Description
Load and Store instructions		Multiply/Divide instructions	
LB[Note 1]	Load Byte	MULT	Multiply
LBU[Note 1]	Load Byte Unsigned	MULTU	Multiply Unsigned
LH[Note 1]	Load Halfword	DIV	Divide
LHU[Note 1]	Load Halfword Unsigned	DIVU	Divide Unsigned
LW[Note 1]	Load Word	MFHI	Move From HI
LWU[Notes 1, 2]	Load Word Unsigned	MFLO	Move From LO
LD[Notes 1, 2]	Load Doubleword	DMULT[Note 2]	Doubleword Multiply
SB[Note 1]	Store Byte	DMULTU[Note 2]	Doubleword Multiply Unsigned
SH[Note 1]	Store Halfword	DDIV[Note 2]	Doubleword Divide
SW[Note 1]	Store Word	DDIVU[Note 2]	Doubleword Divide Unsigned
SD[Notes 1, 2]	Store Doubleword		
		Jump/Branch instructions	
Arithmetic instructions: ALU immediate instructions		JAL	Jump and Link
LI[Note 1]	Load Immediate	JALX	Jump and Link Exchange
ADDIU[Note 1]	Add Immediate Unsigned	JR	Jump Register
DADDIU[Notes 1, 2]	Doubleword Add Immediate Unsigned	JALR	Jump and Link Register
SLTI[Note 1]	Set on Less Than Immediate	BEQZ[Note 1]	Branch on Equal to Zero
SLTIU[Note 1]	Set on Less Than Immediate Unsigned	BNEZ[Note 1]	Branch on Not Equal to Zero
CMPI[Note 1]	Compare Immediate	BTEQZ[Note 1]	Branch on T Equal to Zero
		BTNEZ[Note 1]	Branch on T Not Equal to Zero
Arithmetic instructions: 2/3 operand register instructions		B[Note 1]	Branch Unconditional
ADDU	Add Unsigned		
SUBU	Subtract Unsigned	Shift instructions	
DADDU[Note 2]	Doubleword Add Unsigned	SLL[Note 1]	Shift Left Logical
DSUBU[Note 2]	Doubleword Subtract Unsigned	SRL[Note 1]	Shift Right Logical
SLT	Set on Less Than	SRA[Note 1]	Shift Right Arithmetic
SLTU	Set on Less Than Unsigned	SLLV	Shift Left Logical Variable
CMP	Compare	SRLV	Shift Right Logical Variable
NEG	Negate	SRAV	Shift Right Arithmetic Variable
AND	AND	DSLL[Notes 1, 2]	Doubleword Shift Left Logical
OR	OR	DSRL[Notes 1, 2]	Doubleword Shift Right Logical
XOR	Exclusive OR	DSRA[Notes 1, 2]	Doubleword Shift Right Arithmetic
NOT	Not	DSLLV[Note 2]	Doubleword Shift Left Logical Variable
MOVE	Move	DSRLV[Note 2]	Doubleword Shift Right Logical Variable
		DSRAV[Note 2]	Doubleword Shift Right Arithmetic Variable
Special instructions			
EXTEND	Extend		
BREAK	Breakpoint		

Notes 1. Extendable instruction. For details, see **4.8.2 Extend instructions**.

2. Can be used in 64-bit mode and 32-bit kernel mode.

4.6 Instruction Format

The MIPS16 instruction set has a length of 16 bits and is located at the halfword boundary. One part of Jump instructions and instructions for which the Extend instruction extends immediate become 32 bits in length, but crossing the word boundary does not represent a problem.

The instruction format is shown below. Variable subfields are indicated with lower case letters (rx, ry, rz, immediate, etc.).

In the case of special functions, constants are input to the two instruction subfields op and funct. These values are indicated by upper case mnemonics. For example, in the case of the Load Byte instruction, op is LB, and in the case of the Add instruction, op is SPECIAL, and function is ADD.

The constants of the fields used in the instruction formats are shown below.

Table 4-4. Field Definition

Field	Definition
op	5-bit major operation code
rx	3-bit source/destination register specification
ry	3-bit source/destination register specification
immediate or imm	4-bit, 5-bit, 8-bit, or 11-bit immediate value, branch displacement, or address displacement
rz	3-bit source/destination register specification
Funct or F	Function field

I-type (immediate) instruction format

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="5">op</td><td colspan="11">immediate</td></tr>
</table>

RI-type instruction format

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="5">op</td><td colspan="3">rx</td><td colspan="8">immediate</td></tr>
</table>

RR-type instruction format

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="5">op</td><td colspan="3">rx</td><td colspan="3">ry</td><td colspan="5">Funct</td></tr>
</table>

RRI-type instruction format

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="5">op</td><td colspan="3">rx</td><td colspan="3">ry</td><td colspan="5">immediate</td></tr>
</table>

RRR-type instruction format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RRR					rx			ry			rz			F	

RRI-A type instruction format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RRI-A					rx			ry			F	immediate			

SHIFT instruction format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHIFT					rx			ry			shamt^{Note}			F	

Note The 3-bit shamt field can encode shift count numbers from 0 to 7. 0-bit shift (NOP) cannot be executed. 0 is regarded as shift count 8.

I8-type instruction format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I8					Funct			immediate							

I8_MOVR32 instruction format (used only with MOVR32 instruction)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I8					Funct			ry			r32[4:0]				

I8_MOV32R instruction format (used only with MOV32R instruction)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I8					Funct			r32[2:0, 4:3]^{Note}					rz		

Note The r32 field uses special bit encoding. For example, encoding of $7 (00111) is 11100 in the r32 field.

I64-type instruction format

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="5">I64</td><td colspan="3">Funct</td><td colspan="8">immediate</td></tr>
</table>

RI64-type instruction format

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="5">I64</td><td colspan="3">Funct</td><td colspan="3">ry</td><td colspan="5">immediate</td></tr>
</table>

JAL and JALX instruction format

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="5">JAL</td><td>X</td><td colspan="5">immediate 20:16</td><td colspan="5">immediate 25:21</td><td colspan="16">immediate 15:0</td></tr>
</table>

Remark JAL in case of X = 0 instruction
JALX in case of X = 1 instruction

EXT-I instruction format

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="5">EXTEND</td><td colspan="6">immediate 10:5</td><td colspan="5">immediate 15:11</td><td colspan="5">MAJOR</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td colspan="5">immediate 4:0</td></tr>
</table>

EXT-RI instruction format

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="5">EXTEND</td><td colspan="6">immediate 10:5</td><td colspan="5">immediate 15:11</td><td colspan="5">MAJOR</td><td colspan="3">rx</td><td>0</td><td>0</td><td>0</td><td colspan="5">immediate 4:0</td></tr>
</table>

EXT-RRI instruction format

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="5">EXTEND</td><td colspan="6">immediate 10:5</td><td colspan="5">immediate 15:11</td><td colspan="5">MAJOR</td><td colspan="3">rx</td><td colspan="3">ry</td><td colspan="5">immediate 4:0</td></tr>
</table>

EXT-RRI-A instruction format

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="5">EXTEND</td><td colspan="7">immediate 10:4</td><td colspan="4">imm 14:11</td><td colspan="5">RRI-A</td><td colspan="3">rx</td><td colspan="3">ry</td><td>F</td><td colspan="4">imm 3:0</td></tr>
</table>

EXT-SHIFT instruction format

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="5">EXTEND</td><td colspan="5">shamt 4:0</td><td>S5 Note</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td colspan="5">SHIFT</td><td colspan="3">rx</td><td colspan="3">ry</td><td>0</td><td>0</td><td>0</td><td colspan="2">F</td></tr>
</table>

Note Only in the case of DSLL, the S5 bit is the most significant bit of the 6-bit shift count field (shamt). In the case of all 32-bit extended shifts, S5 must be 0. For a normal shift instruction, the display of shift count 0 is considered as shift count 8, but the extended shift instruction does not perform such mapping changes. Therefore, 0-bit shift using the extended format is possible.

EXT-I8 instruction format

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="5">EXTEND</td><td colspan="6">immediate 10:5</td><td colspan="5">immediate 15:11</td><td colspan="5">I8</td><td colspan="3">Funct</td><td>0</td><td>0</td><td>0</td><td colspan="5">immediate 4:0</td></tr>
</table>

EXT-I64 instruction format

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="5">EXTEND</td><td colspan="6">immediate 10:5</td><td colspan="5">immediate 15:11</td><td colspan="5">I64</td><td colspan="3">Funct</td><td>0</td><td>0</td><td>0</td><td colspan="5">immediate 4:0</td></tr>
</table>

EXT-RI64 instruction format

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="5">EXTEND</td><td colspan="6">immediate 10:5</td><td colspan="5">immediate 15:11</td><td colspan="5">I64</td><td colspan="3">Funct</td><td colspan="3">ry</td><td colspan="5">immediate 4:0</td></tr>
</table>

EXT-SHIFT64 instruction format

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="5">EXTEND</td><td colspan="5">shamt 4:0</td><td>S5 Note</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td colspan="5">RR</td><td>0</td><td>0</td><td>0</td><td colspan="3">ry</td><td colspan="5">Function</td></tr>
</table>

Note The S5 bit is the most significant bit of the 6-bit shift count field (shamt). In the case of a normal shift instruction, the display of shift count 0 is considered as shift count 8, but the extended shift instruction does not perform such mapping changes.
Therefore, 0-bit shift using the extended format is possible.

4.7 MIPS16 Operation Code Bit Encoding

This section describes encoding for major operation code and minor operation code. Table 4-5 shows bit encoding of the MIPS16 major operation code. Tables 4-6 to 4-11 show bit encoding of the minor operation code. The italic operation codes in the tables are instructions for the extended ISA.

Table 4-5. Bit Encoding of Major Operation Code (op)

Instruction Bits	Instruction Bits [13:11]							
[15:14]	000	001	010	011	100	101	110	111
00	addiusp[Note 1]	addiupc[Note 2]	b	jal(x)[Note 3]	beqz	bnez	SHIFT	*ld*
01	RRI-A	addiu8[Note 4]	slti	sltiu	I8	li	cmpi	*sd*
10	lb	lh	lwsp	lw	lbu	lhu	lwpc	*lwu*
11	sb	sh	swsp	sw	RRR	RR	extend	*I64*

Notes 1. addiusp: addiu rx, sp, immediate

2. addiupc: addiu rx, pc, immediate

3. jal(x): jal instruction and jalx instruction

4. addiu8: aadiu rx, immediate

Table 4-6. RR Minor Operation Code (RR-Type Instruction)

Instruction Bits	Instruction Bits [2:0]							
[4:3]	000	001	010	011	100	101	110	111
00	j(al)r[Note 1]	∗	slt	sltu	sllv	break	srlv	srav
01	*dsrl*[Note 2]	φ	cmp	neg	and	or	xor	not
10	mfhi	∗	mflo	*dsra*[Note 2]	*dsllv*	∗	*dsrlv*	*dsrav*
11	mult	multu	div	divu	*dmult*	*dmultu*	*ddiv*	*ddivu*

Notes 1. j(al)r: jr rx instruction (ry = 000)
jr ra instruction (ry = 001, rx = 000)
jalr ra, rx instruction (ry = 010)

2. dsrl and dsra use the rx register field to encode the shift count (8-digit shift for 0). In the case of the extended version of these two instructions, the EXT-SHIFT64 format is used. Only these two RR instructions can be extended.

Remarks The symbols in the figures have the following meaning.

∗: Execution of operation code with an asterisk on the current VR4121 causes a reserved instruction exception to be generated. This code is reserved for future extension.

φ: Operation code with φ is invalid, but no reserved instruction exception is generated in the VR4121.

Table 4-7. RRR Minor Operation Code (RRR-Type Instruction)

Instruction Bits [1:0]			
00	01	10	11
daddu	addu	*dsubu*	subu

Table 4-8. RRRI-A Minor Operation Code (RRI-Type ADD Instruction)

Instruction Bit [4]	
0	1
addiu[Note 1]	*daddiu*[Note 2]

Notes 1. addiu: addiu ry, rx, immediate

2. daddiu: daddiu ry, rx immediate

Table 4-9. SHIFT Minor Operation Code (SHIFT-Type Instruction)

Instruction Bits [1:0]			
00	01	10	11
sll	*dsll*	srl	sra

Table 4-10. I8 Minor Operation Code (I8-Type Instruction)

Instruction Bits [10:8]							
000	001	010	011	100	101	110	111
bteqz	btnez	swrasp[Note 1]	adjsp[Note 2]	∗	mov32r[Note 3]	∗	movr32[Note 4]

Notes 1. swrasp: sw ra, immediate(sp)

2. adjsp: addiu sp, immediate

3. mov32r: move r32, rz

4. movr32: move ry, r32

Remark The symbols used in the figures have the following meaning.

∗: Execution of operation code with an asterisk on the current VR4121 causes a reserved instruction exception to be generated. This code is reserved for future extension.

Table 4-11. I64 Minor Operation Code (64-Bit Only, I64-Type Instruction)

Instruction Bits [10:8]							
000	001	010	011	100	101	110	111
*ldsp*Note 1	*sdsp*Note 2	*sdrasp*Note 3	*dadjsp*Note 4	*ldpc*Note 5	*daddiu5*Note 6	*dadiupc*Note 7	*dadiusp*Note 8

Notes 1. ldsp: ld ry, immediate

2. sdsp: sd ry, immediate

3. sdrasp: sd ra, immediate

4. dadjsp: daddiu sp, immediate

5. ldpc: ld ry, immediate

6. daddiu5: daddiu ry, immediate

7. dadiupc: daddiu ry, pc, immediate

8. dadiusp: daddiu ry, sp, immediate

4.8 Outline of Instructions

This section describes the assembler syntax and defines each instruction. Instructions can be divided into the following four types.

- Load and store instructions
- Computational instructions
- Jump and branch instructions
- Special instructions

4.8.1 PC-Relative instructions

PC-relative instructions are the instruction format first defined among the MIPS16 instruction set. MIPS16 supports both extension and non-extension through the Extend instruction for four PC-relative instructions.

Load Word	LW rx, offset (pc)
Load Doubleword	LD ry, offset (pc)
Add Immediate Unsigned	ADDIU rx, pc, immediate
Doubleword Add Immediate Unsigned	DADDIU ry, pc, immediate

All these instructions calculate the PC value of a PC-relative instruction or the PC value of the instruction immediately preceding as the base address. The address calculation base using various function combinations is shown next.

Table 4-12. Base PC Address Setting

Instruction	Base PC Value
Non-extension PC-relative instructions not located in Jump delay slot	PC of instruction
Extension PC-relative instruction	PC of Extend instruction
Non-extension PC-relative instruction in Jump delay slot of JR or JALR	PC of JR instruction or JALR instruction
Non-extension PC-relative instruction in Jump delay slot of JAL or JALX	PC of initial halfword of JAL or JALX[Note]

Note Because the JAL and JALX instruction length is 32 bits.

The PC value used as the base for address calculation for the PC-relative instruction outlines shown in tables 4-14 and 4-15 is called base PC value. The base PC value is defined so as to be equivalent to the exception program counter (EPC) value related to the PC-relative instruction.

4.8.2 Extend instruction

The Extend instruction can extend the immediate fields of MIPS16 instructions, which have fewer immediate fields than equivalent 32-bit MIPS instructions. The Extend instruction must always precede (by one instruction) the instruction whose immediate field you want to extend. Every extended instruction consumes four bytes in program memory instead of two bytes (two bytes for Extend and two bytes for the instruction being extended), and it can cross a word boundary.

For example, the MIPS16 instruction contains a five-bit immediate.

LW ry, offset (rx)

The immediate expands to 16 bits (000000000 | |offset| |00) before execution in the pipeline. This allows 32 different offset values of 0, 4, 8, and up through 124. Once extended, this instruction can hold any of the normal 65,536 values in the range –32,768 through 32,767.

Shift instructions are extended to 5-bit unsigned immediate values. All other immediate instructions expand to either signed or unsigned 16-bit immediate values. The only exceptions are which can be extended only to a 15-bit signed immediate.

ADDIU ry, rx, immediate
DADDIU ry, rx, immediate

Extended instructions should not be placed in jump delay slots. Otherwise, the results are unpredictable because the pipeline would attempt to execute one half the instruction.

Table 4-13 lists the MIPS16 extendable instructions, the size of their immediate, and how much each immediate can be extended when preceded with the Extend instruction.

For the instruction format of the Extend instruction, see **4.6 Instruction Format**.

Table 4-13. Extendable MIPS16 Instructions

MIPS16 Instruction	MIPS16 Immediate	Instruction Format	Extended Immediate	Instruction Format
Load Byte	5	RRI	16	EXT-RRI
Load Byte Unsigned	5	RRI	16	EXT-RRI
Load Halfword	5	RRI	16	EXT-RRI
Load Halfword Unsigned	5	RRI	16	EXT-RRI
Load Word	5 8	RRI RI	16 16	EXT-RRI EXT-RI
Load Word Unsigned	5	RRI	16	EXT-RRI
Load Doubleword	5	RRI	16	EXT-RRI
Store Byte	5	RRI	16	EXT-RRI
Store Halfword	5	RRI	16	EXT-RRI
Store Word	5 (Other) 8 (SW rx, offset(sp)) 8 (SW ra, offset(sp))	RRI RI I8	16 16 16	EXT-RRI EXT-RI EXT-I8
Store Doubleword	5 (SD ry, offset(rx)) 8 (Other)	RRI I64	16 16	EXT-RRI EXT-I64
Load Immediate	8	RI	16	EXT-RI
Add Immediate Unsigned	4 (ADDIU ry, rx, imm) 8 (ADDIU sp, imm) 8 (Other)	RRI-A I8 RI	15 16 16	EXT-RRI-A EXT-I8 EXT-RI
Doubleword Add Immediate Unsigned	4 (DADDIU ry, rx, imm) 5 (DADDIU ry, pc, imm) 8 (Other)	RRI-A RI64 I64	15 16 16	EXT-RRI-A EXT-RI64 EXT-I64
Set on Less Than Immediate	8	RI	16	EXT-RI
Set on Less Than Immediate Unsigned	8	RI	16	EXT-RI
Compare Immediate	8	RI	16	EXT-RI
Shift Left Logical	3	SHIFT	5	EXT-SHIFT
Shift Right Logical	3	SHIFT	5	EXT-SHIFT
Shift Right Arithmetic	3	SHIFT	5	EXT-SHIFT
Doubleword Shift Left Logical	3	SHIFT	6	EXT-SHIFT
Doubleword Shift Right Logical	3	RR	6	EXT- SHIFT64
Doubleword Shift Right Arithmetic	3	RR	6	EXT- SHIFT64
Branch on Equal to Zero	8	RI	16	EXT-RI
Branch on Not Equal to Zero	8	RI	16	EXT-RI
Branch on T Equal to Zero	8	I8	16	EXT-I8
Branch on T Not Equal to Zero	8	I8	16	EXT-I8
Branch Unconditional	11	I	16	EXT-I

4.8.3 Delay slots

MIPS16 instructions normally execute in one cycle. However, some instructions have special requirements that must be met to assure optimum instruction flow. The instructions include all Load, Branch, and Multiply/Divide instructions.

(1) Load delay slots

MIPS16 operates with delayed loads. This is similar to the method used by 32-bit MIPS instruction sets. If another instruction references the load destination register before the load operation is completed, one cycle occurs automatically. To assure the best performance, the compiler should always schedule load delay slots as early as possible.

(2) Branch delay slots not supported

Unlike for 32-bit MIPS instructions, there are no branch delay slots for branch instructions in MIPS16. If a branch is taken, the instruction that immediately follows the branch (instruction corresponding to 32-bit MIPS delay slot) is cancelled. There are no restrictions on the instruction that follows a branch instruction, and such instruction is executed only when a branch is not taken. Branches, jumps, and extended instructions are permitted in the instruction slot after a branch.

(3) Jump delay slots

With MIPS16, there is a delay of one cycle after each jump instruction. The processor executes any instruction in the jump delay slot before it executes the jump target instruction. Two restrictions apply to any instruction placed in the jump delay slot:

1. Do not specify a branch or jump in the delay slot.
2. Do not specify an extended instruction (32-bits) in the delay slot. Doing so will make the results unpredictable.

(4) Multiply and divide scheduling

Multiply and divide latency depends on the hardware implementation. If an MFLO or MFHI instruction references the Multiply or Divide result registers before the result is ready, the pipeline stalls until the operation is complete and the result is available. However, to assure the best performance, the compiler should always schedule Multiply and Divide instructions as early as possible.

MIPS16 requires that all MFHI and MFLO instructions be followed by two instructions that do not write to the HI or LO registers. Otherwise, the data read by MFLO or MFHI will be undefined. The Extend instruction is counted singly as one instruction.

4.8.4 Instruction details

(1) Load and store instructions

Load and store instructions move data between memory and the general registers. The only addressing mode that is supported is the mode for adding immediate offset to the base register.

Table 4-14. Load and Store Instructions (1/3)

Instruction	Format and Description
Load Byte	LB ry, offset (rx) The 5-bit immediate is zero extended and then added to the contents of general register rx to form the virtual address. The bytes of the memory location specified by the address are sign extended and loaded into general register ry.
Load Byte Unsigned	LBU ry, offset (rx) The 5-bit immediate is zero extended and then added to the contents of general register rx to form the virtual address. The bytes of the memory location specified by the address are zero extended and loaded into general register ry.
Load Halfword	LH ry, offset (rx) The 5-bit immediate is shifted left one bit, zero extended, and then added to the contents of general register rx to form the virtual address. The halfword of the memory location specified by the address is sign extended and loaded to general register ry. If the least significant bit of the address is not 0, an address error exception is generated.
Load Halfword Unsigned	LHU ry, offset (rx) The 5-bit immediate is shifted left one bit, zero extended, and then added to the contents of general register rx to form the virtual address. The halfword of the memory location specified by the address is zero extended and loaded to general register ry. If the least significant bit of the address is not 0, an address error exception is generated.
Load Word	LW ry, offset (rx) The 5-bit immediate is shifted left two bits, zero extended, and then added to the contents of general register rx to form the virtual address. The word of the memory location specified by the address is loaded to general register ry. In the 64-bit mode, it is further sign extended to 64 bits. If either of the lower two bits is not 0, an address error exception is generated.
	LW rx, offset (pc) The two lower bits of the BasePC value associated with the instruction are cleared to form the masked BasePC value. The 8-bit immediate is shifted left two bits, zero extended, and then added to the masked BasePC to form the virtual address. The contents of the word at the memory location specified by the address are loaded to general register rx. In the 64-bit mode, it is further sign extended to 64 bits.
	LW rx, offset (sp) The 8-bit immediate is shifted left two bits, zero extended, and then added to the contents of general register sp to form the virtual address. The contents of the word at the memory location specified by the address are loaded to general register rx. In the 64-bit mode, it is further sign extended to 64 bits. If either of the two lower bits of the address is 0, an address error exception is generated.

Table 4-14. Load and Store Instructions (2/3)

Instruction	Format and Description
Load Word Unsigned	LWU ry, offset (rx) The 5-bit immediate is shifted left two bits, zero extended to 64 bits, and then added to the contents of general register rx to form the virtual address. The word of the memory location specified by the address is zero extended and loaded to general register ry. If either of the two lower bits of the address is not 0, an address error exception is generated. This operation is defined in the 64-bit mode and the 32-bit kernel mode. When this instruction is executed in the 32-bit user/supervisor mode, a reserved instruction exception is generated.
Load Doubleword	LD ry, offset (rx) The 5-bit immediate is shifted left three bits, zero extended to 64 bits, and then added to the contents of general register rx to form the virtual address. The 64-bit doubleword of the memory location specified by the address is loaded to general register ry. If any of the lower three bits of the address is not 0, an address error exception is generated. This operation is defined in the 64-bit mode and the 32-bit kernel mode. When this instruction is executed in the 32-bit user/supervisor mode, a reserved instruction exception is generated.
	LD ry, offset (pc) The lower three bits of the base PC value related to the instruction are cleared to form the masked BasePC value. The 5-bit immediate is shifted left three bits, zero extended to 64 bits, and then added to the masked BasePC to form the virtual address. The 64-bit doubleword at the memory location specified by the address is loaded to general register ry. This operation is defined in the 64-bit mode and the 32-bit kernel mode. When this instruction is executed in the 32-bit user/supervisor mode, a reserved instruction exception is generated.
	LD ry, offset (sp) The 5-bit immediate is shifted left three bits, zero extended to 64 bits, and added to the contents of general register sp to form the virtual address. The 64-bit doubleword at the memory location specified by the address is loaded to general register ry. If any of the three lower bits of the address is not 0, an address error exception is generated. This operation is defined in the 64-bit mode and the 32-bit kernel mode. When this instruction is executed in the 32-bit user/supervisor mode, a reserved instruction exception is generated.

Table 4-14. Load and Store Instructions (3/3)

Instruction	Format and Description
Store Byte	SB ry, offset (rx) The 5-bit immediate is zero extended and then added to the contents of general register rx to form the virtual address. The least significant byte of general register ry is stored to the memory location specified by the address.
Store Halfword	SH ry, offset (rx) The 5-bit immediate is shifted left one bit, zero extended, and then added to the contents of general register rx to form the virtual address. The lower halfword of general register ry is stored to the memory location specified by the address. If the least significant bit of the address is not 0, an address error exception is generated.
Store Word	SW ry, offset (rx) The 5-bit immediate is shifted left two bits, zero extended, and then added to the contents of general register rx to form the virtual address. The contents of general register ry are stored to the memory location specified by the address. If either of the two lower bits of the address is not 0, an address error exception is generated.
	SW rx, offset (sp) The 8-bit immediate is shifted left two bits, zero extended, and then added to the contents of general register sp to form the virtual address. The contents of general register rx are stored to the memory location specified by the address. If either of the two lower bits of the address is not 0, and address error exception is generated.
	SW ra, offset (sp) The 8-bit immediate is shifted left two bits, zero extended, and then added to the contents of general register sp to form the virtual address. The contents of general register ra are stored to the memory location specified by the address. If either of the two lower bits of the address is not 0, an address error exception is generated.
Store Doubleword	SD ry, offset (rx) The 5-bit immediate is shifted left three bits, zero extended to 64 bits, and then added to the contents of general register rx to form the virtual address. The 64 bits of general register ry are stored to the memory location specified by the address. If any of the lower three bits of the address is not 0, an address error exception is generated. This operation is defined in the 64-bit mode and the 32-bit kernel mode. When this instruction is executed in the 32-bit user/supervisor mode, a reserved instruction exception is generated.
	SD ry, offset (sp) The 5-bit immediate is shifted left three bits, zero extended to 64 bits, and then added to the contents of general register sp to form the virtual address. The 64 bits of general register ry are stored to the memory location specified by the address. If any of the lower three bits of the address is not 0, an address error exception is generated. This operation is defined in the 64-bit mode and the 32-bit kernel mode. When this instruction is executed in the 32-bit user/supervisor mode, a reserved instruction exception is generated.
	SD ra, offset (sp) The 8-bit immediate is shifted left three bits, zero extended to 64 bits, and then added to the contents of general register sp to form the virtual address. The 64 bits of general register ra are stored to the memory location specified by the memory. If any of the three lower bits of the address is not 0, an address error exception is generated. This operation is defined in the 64-bit mode and the 32-bit kernel mode. When this instruction is executed in the 32-bit user/supervisor mode, a reserved instruction exception is generated.

(2) Computational instructions

Computational instructions perform arithmetic, logical, and shift operations on values in registers. There are four categories of computational instructions: ALU immediate, two/three-operand register-type, shift, and multiply/divide.

Table 4-15. ALU Immediate Instructions (1/2)

Instruction	Format and Description
Load Immediate	LI rx, immediate The 8-bit immediate is zero extended and loaded to general register rx.
Add Immediate Unsigned	ADDIU ry, rx, immediate The 4-bit immediate is sign extended and then added to the contents of general register rx to form a 32-bit result. The result is placed into general register ry. No integer overflow exception occurs under any circumstances. In the 64-bit mode, the operand must be a 64-bit value formed by sign-extending a 32-bit value.
	ADDIU rx, immediate The 8-bit immediate is sign extended and then added to the contents of general register rx to form a 32-bit result. The result is placed into general register rx. No integer overflow exception occurs under any circumstances. In the 64-bit mode, the operand must be a 64-bit value formed by sign-extending a 32-bit value.
	ADDIU sp, immediate The 8-bit immediate is shifted left three bits, sign extended, and then added to the contents of general register sp to form a 32-bit result. The result is placed into general register sp. No integer overflow exception occurs under any circumstances. In the 64-bit mode, the operand must be a 64-bit value formed by sign-extending a 32-bit value.
	ADDIU rx, pc, immediate The two lower bits of the BasePC value associated with the instruction are cleared to form the masked BasePC value. The 8-bit immediate is shifted left two bits, zero extended, and then added to the masked BasePC value to form the virtual address. This address is placed into general register rx. No integer overflow exception occurs under any circumstances.
	ADDIU rx, sp, immediate The 8-bit immediate is shifted left two bits, zero extended, and then added to the contents of register sp to form a 32-bit result. The result is placed into general register rx. No integer overflow exception occurs under any circumstance. In the 64-bit mode, the operand must be a 64-bit value formed by sign-extending a 32-bit value.

Table 4-15. ALU Immediate Instructions (2/2)

Instruction	Format and Description
Doubleword Add Immediate Unsigned	DADDIU ry, rx, immediate The 4-bit immediate is sign extended to 64 bits, and then added to the contents of register rx to form a 64-bit result. The result is placed into general register ry. No integer overflow exception occurs under any circumstances. This operation is defined in the 64-bit mode and the 32-bit kernel mode. When this instruction is executed in the 32-bit user/supervisor mode, a reserved instruction exception is generated.
	DADDIU ry, immediate The 5-bit immediate is sign extended to 64 bits, and then added to the contents of register ry to form a 64-bit result. The result is placed into general register ry. No integer overflow exception occurs under any circumstances. This operation is defined in the 64-bit mode and the 32-bit kernel mode. When this instruction is executed in the 32-bit user/supervisor mode, a reserved instruction exception is generated.
	DADDIU sp, immediate The 8-bit immediate is shifted left three bits, sign extended to 64 bits, and then added to the contents of register sp to form a 64-bit result. The result is placed into general register sp. No integer overflow exception occurs under any circumstances. This operation is defined in the 64-bit mode and the 32-bit kernel mode. When this instruction is executed in the 32-bit user/supervisor mode, a reserved instruction exception is generated.
	DADDIU ry, pc, immediate The two lower bits of the BasePC value associated with the instruction are cleared to form the masked BasePC value. The 5-bit immediate is shifted left two bits, zero extended, and added to the masked BasePC value to form the virtual address. This address is placed into general register ry. No integer overflow exception occurs under any circumstances. This operation is defined in the 64-bit mode and the 32-bit kernel mode. When this instruction is executed in the 32-bit user/supervisor mode, a reserved instruction exception is generated.
	DADDIU ry, sp, immediate The 5-bit immediate is shifted left two bits, zero extended to 64 bits, and then added to the contents of register sp to form a 64-bit result. This result is placed into register ry. No integer overflow exception occurs under any circumstances. This operation is defined in the 64-bit mode and the 32-bit kernel mode. When this instruction is executed in the 32-bit user/supervisor mode, a reserved instruction exception is generated.
Set on Less Than Immediate	SLTI rx, immediate The 8-bit immediate is zero extended and subtracted from the contents of general register rx. Considering both quantities as signed integers, if rx is less than the zero-extended immediate, the result is set to 1; otherwise, the result is set to 0. The result is placed into register T ($24).
Set on Less Than Immediate Unsigned	SLTIU rx, immediate The 8-bit immediate is zero extended and subtracted from the contents of general register rx. Considering both quantities as signed integers, if rx is less than the zero-extended immediate, the result is set to 1; otherwise, the result is set to 0. The result is placed into register T ($24).
Compare Immediate	CMPI rx, immediate The 8-bit immediate is zero extended and exclusive ORed in 1-bit units with the contents of general register rx. The result is placed into register T ($24).

Table 4-16. Two-/Three-Operand Register Type (1/2)

Instruction	Format and Description
Add Unsigned	ADDU rz, rx, ry The contents of general registers rx and ry are added together to form a 32-bit result. The result is placed into general register rz. No integer overflow exception occurs under any circumstances. In the 64-bit mode, the operand must be a 64-bit value formed by sign-extending a 32-bit value.
Subtract Unsigned	SUBU rz, rx, ry The contents of general register ry are subtracted from the contents of general register rx. The 32-bit result is placed into general register rz. No integer overflow exception occurs under any circumstances. In the 64-bit mode, the operand must be a 64-bit value formed by sign-extending a 32-bit value.
Doubleword Add Unsigned	DADDU rz, rx, ry The contents of general register ry are added to the contents of general register rx. The 64-bit result is placed into register rz. No integer overflow exception occurs under any circumstances. This operation is defined in the 64-bit mode and the 32-bit kernel mode. When this instruction is executed in the 32-bit user/supervisor mode, a reserved instruction exception is generated.
Doubleword Subtract Unsigned	DSUBU rz, rx, ry The contents of general register ry are subtracted from the contents of general register rx. The 64-bit result is placed into general register rz. No integer overflow exception occurs under any circumstances. This operation is defined in the 64-bit mode and the 32-bit kernel mode. When this instruction is executed in the 32-bit user/supervisor mode, a reserved instruction exception is generated.
Set on Less Than	SLT rx, ry The contents of general register ry are subtracted from the contents of general register rx. Considering both quantities as signed integers, if the contents of rx are less than the contents of ry, the result is set to 1; otherwise, the result is set to 0. The result is placed into register T ($24). No integer overflow exception occurs. The comparison is valid even if the subtraction overflows.
Set on Less Than Unsigned	SLTU rx, ry The contents of general register ry are subtracted from the contents of general register rx. Considering both quantities as unsigned integers, if the contents of rx are less than the contents of ry, the result is set to 1; otherwise, the result it set to 0. The result is place in register T ($24). No integer overflow exception occurs. The comparison is valid even if the subtraction overflows.

Table 4-16. Two-/Three-Operand Register Type (2/2)

Instruction	Format and Description
Compare	CMP rx, ry The contents of general register ry are Exclusive-ORed with the contents of general register rx. The result is placed into register T ($24).
Negate	NEG rx, ry The contents of general register ry are subtracted from zero to form a 32-bit result. The result is placed in general register rx.
AND	AND rx, ry The contents of general register ry are logical ANDed with the contents of general register rx in 1-bit units. The result is placed in general register rx.
OR	OR rx, ry The contents of general register ry are logical ORed with the contents of general register ry. The result is placed in general register rx.
Exclusive OR	XOR rx, ry The contents of general register ry are Exclusive-ORed with the contents of general register rx in 1-bit units. The result is placed in general register rx.
NOT	NOT rx, ry The contents of general register ry are inverted in 1-bit units and placed in general register rx.
Move	MOVE ry, r32 The contents of general register r32 are moved to general register ry. r32 can specify any one of the 32 general registers.
Move	MOVE r32, rz The contents of general register rz are moved to general register r32. r32 can specify any one of the 32 general registers.

Table 4-17. Shift Instructions (1/2)

Instruction	Format and Description
Shift Left Logical	SLL rx, ry, immediate The 32-bit contents of general register ry are shifted left and zeros are inserted into the emptied low-order bits. The 3-bit immediate specifies the shift count. A shift count of 0 is interpreted as a shift count of 8. The result is placed in general register rx. In the 64-bit mode, the value that is formed by sign-extending shifted 32-bit value is stored as the result.
Shift Right Logical	SRL rx, ry, immediate The 32-bit contents of general register ry are shifted right, and zeros are inserted into the emptied high-order bits. The 3-bit immediate specifies the shift count. A shift count of 0 is interpreted as a shift count of 8. The result is placed in general register rx. In the 64-bit mode, the value that is formed by sign-extending shifted 32-bit value is stored as the result.
Shift Right Arithmetic	SRA rx, ry, immediate The 32-bit contents of general register ry are shifted right and the emptied high-order bits are sign extended. The 3-bit immediate specifies the shift count. A shift count of 0 is interpreted as a shift count of 8. In the 64-bit mode, the value that is formed by sign-extending shifted 32-bit value is stored as the result.
Shift Left Logical Variable	SLLV ry, rx The 32-bit contents of general register ry are shifted left, and zeros are inserted into the emptied low-order bits. The five low-order bits of general register rx specify the shift count. The result is placed in general register ry. In the 64-bit mode, the value that is formed by sign-extending shifted 32-bit value is stored as the result.
Shift Right Logical Variable	SRLV ry, rx The 32-bit contents of general register ry are shifted right, and the emptied high-order bits are sign extended. The five lower-order bits of general register rx specify the shift count. The register is placed in general register ry. In the 64-bit mode, the value that is formed by sign-extending shifted 32-bit value is stored as the result.
Shift Right Arithmetic Variable	SRAV ry, rx The 32-bit contents of general register ry are shifted right, and the emptied high-order bits are sign extended. The five low-order bits of general register rx specify the shift count. The result is placed in general register ry. In the 64-bit mode, the value that is formed by sign-extending shifted 32-bit value is stored as the result.

Table 4-17. Shift Instructions (2/2)

Instruction	Format and Description
Doubleword Shift Left Logical	DSLL rx, ry, immediate The 64-bit doubleword contents of general register ry are shifted left, and zeros are inserted into the emptied low-order bits. The 3-bit immediate specifies the shift count. A shift count of 0 is interpreted as a shift count of 8. The 64-bit result is placed in general register rx. This operation is defined in the 64-bit mode and the 32-bit kernel mode. When this instruction is executed in the 32-bit user/supervisor mode, a reserved instruction exception is generated.
Doubleword Shift Right Logical	DSRL ry, immediate The 64-bit doubleword contents of general register ry are shifted right, and zeros are inserted into the emptied high-order bits. The 3-bit immediate specifies the shift count. A shift count of 0 is interpreted as a shift count of 8. The result is placed in general register ry. This operation is defined in the 64-bit mode and the 32-bit kernel mode. When this instruction is executed in the 32-bit user/supervisor mode, a reserved instruction exception is generated.
Doubleword Shift Right Arithmetic	DSRA ry, immediate The 64-bit doubleword contents of general register ry are shifted right, and the emptied high-order bits are sign extended. The 3-bit immediate specifies the shift count. A shift count of 0 is interpreted as a shift count of 8. The result is placed in general register ry. This operation is defined in the 64-bit mode and the 32-bit kernel mode. When this instruction is executed in the 32-bit user/supervisor mode, a reserved instruction exception is generated.
Doubleword Shift Left Logical Variable	DSLLV ry, rx The 64-bit doubleword contents of general register ry are shifted left, and zeros are inserted into the emptied low-order bits. The six low-order bits of general register rx specify the shift count. The result is placed in general register ry. This operation is defined in the 64-bit mode and the 32-bit kernel mode. When this instruction is executed in the 32-bit user/supervisor mode, a reserved instruction exception is generated.
Doubleword Shift Right Logical Variable	DSRLV ry, rx The 64-bit doubleword contents of general register ry are shifted right, and zeros are inserted into the emptied high-order bits. The six low-order bits of general register rx specify the shift count. The result is placed in general register ry. This operation is defined in the 64-bit mode and the 32-bit kernel mode. When this instruction is executed in the 32-bit user/supervisor mode, a reserved instruction exception is generated.
Doubleword Shift Right Arithmetic Variable	DSRAV ry, rx The 64-bit doubleword contents of general register ry are shifted right, and the emptied high-order bits are sign extended. The six low-order bits of general register rx specify the shift count. The result is placed in general register ry. This operation is defined in the 64-bit mode and the 32-bit kernel mode. When this instruction is executed in the 32-bit user/supervisor mode, a reserved instruction exception is generated.

Table 4-18. Multiply/Divide Instructions (1/2)

Instruction	Format and Description
Multiply	MULT rx, ry The contents of general registers rx and ry are multiplied, treating both operands as 32-bit two's complement values. No integer overflow exception occurs. In the 64-bit mode, the operand must be a 64-bit value formed by sign-extending a 32-bit value. The low-order 32-bit word of the result are placed in special register LO, and the high-order 32-bit word is placed in special register HI. In the 64-bit mode, each result is sign extended and then stored. If either of the two immediately preceding instructions is MFHI or MFLO, their transfer instruction execution result becomes undefined. To obtain the correct result, insert two or more other instructions between the MFHI, MFLO instructions, and the MULT instruction.
Multiply Unsigned	MULTU rx, ry The contents of general registers rx and ry are multiplied, treating both operands as 32-bit unsigned values. No integer overflow exception occurs. In the 64-bit mode, the operand must be a 64-bit value formed by sign-extending a 32-bit value. The low-order 32-bit word of the result is placed in special register LO, and the high-order 32-bit word is placed in special register HI. In the 64-bit mode, each result is sign extended and stored. If either of the two immediately preceding instructions is MFHI or MFLO, the result of execution of these transfer instructions is undefined. To obtain the correct result, insert two or more other instructions between the MFHI, MFLO instructions and the MULTU instruction.
Divide	DIV rx, ry The contents of general register rx are divided by the contents of general register ry, treating both operands as 32-bit two's complement values. No integer overflow exception occurs. The result when the divisor is 0 is undefined. The 32-bit quotient is placed in special register LO, and the 32-bit remainder is placed in special register HI. In the 64-bit mode, the result is sign extended. Normally, this instruction is executed after instructions checking for division by zero and overflow. If either of the two immediately preceding instructions is MFHI or MFLO, the result of execution of these transfer instructions is undefined. To obtain the correct result, insert two or more other instructions between the MFHI, MFLO instructions and the DIV instruction.
Divide Unsigned	DIVU rx, ry The contents of general register rx are divided by the contents of general register ry, treating both operands as 32-bit unsigned values. No integer overflow exception occurs. The result when the divisor is 0 is undefined. The 32-bit quotient is placed in special register LO, and the 32-bit remainder is placed in special register HI. In the 64-bit mode, the result is sign extended. Normally, this instruction is executed after instructions checking for division by zero. If either of the two immediately preceding instructions is MFHI or MFLO, the result of execution of these transfer instructions is undefined. To obtain the correct result, insert two or more other instructions between the MFHI, MFLO instructions and the DIVU instruction.
Move From HI	MFHI rx The contents of special register HI are loaded into general register rx. To ensure correct operation when an interrupt occurs, do not use an instruction that changes the HI register (MULT, MULTU, DIV, DIVU, DMULT, DMULTU, DDIV, DDIVU) for the two instructions after the MFHI instruction.

Table 4-18. Multiply/Divide Instructions (2/2)

Instruction	Format and Description
Move From LO	MFLO rx The contents of special register LO are loaded into general register rx. To ensure correct operation when an interrupt occurs, do not use an instruction that changes the LO register (MULT, MULTU, DIV, DIVU, DMULT, DMULTU, DDIV, DDIVU) for the two instructions after the MFLO instruction.
Doubleword Multiply	DMULT rx, ry The 64-bit contents of general register rx and ry are multiplied, treating both operands as two's complement values. No integer overflow exception occurs. The low-order 64 bits of the result are placed in special register LO, and the high-order 64 bits are placed in special register HI. If either of the two immediately preceding instructions is MFHI or MFLO, the result of execution of these transfer instructions is undefined. To obtain the correct result, insert two or more other instructions between the MFHI, MFLO instructions and the DMULT instruction. This operation is defined in the 64-bit mode and the 32-bit kernel mode. When this instruction is executed in the 32-bit user/supervisor mode, a reserved instruction exception is generated.
Doubleword Multiply Unsigned	DMULTU rx, ry The 64-bit contents of general registers rx and ry are multiplied, treating both operands as unsigned values. No integer overflow exception occurs. The low-order 64 bits of the result are placed in special register LO, and the high-order 64 bits of the result are placed in special register HI. If either of the two immediately preceding instructions is MFHI or MFLO, the result of execution of these transfer instructions is undefined. To obtain the correct result, insert two or more other instructions between the MFHI, MFLO instructions and the DMULTU instruction. This operation is defined in the 64-bit mode and the 32-bit kernel mode. When this instruction is executed in the 32-bit user/supervisor mode, a reserved instruction exception is generated.
Doubleword Divide	DDIV rx, ry The 64-bit contents of general registers rx are divided by the contents of general register ry, treating both operands as two's complement values. No integer overflow exception occurs. The result when the divisor is 0 is undefined. The 64-bit quotient is placed in special register LO, and the 64-bit remainder is placed in special register HI. Normally, this instruction is executed after instructions checking for division by zero and overflow. If either of the two immediately preceding instructions is MFHI or MFLO, the result of execution of these transfer instructions is undefined. To obtain the correct result, insert two or more other instructions between the MFHI, MFLO instructions and the DDIV instruction. This operation is defined in the 64-bit mode and the 32-bit kernel mode. When this instruction is executed in the 32-bit user/supervisor mode, a reserved instruction exception is generated.
Doubleword Divide Unsigned	DDIVU rx, ry The 64-bit contents of general register rx are divided by the contents of general register ry, treating both operands as unsigned values. No integer overflow exception occurs. The result when the divisor is 0 is undefined. The 64-bit quotient is placed in special register LO, and the 64-bit remainder is placed in special register HI. Normally, this instruction is executed after an instruction checking for division by zero. If either of the two immediately preceding instructions is MFHI or MFLO, the result of execution of these transfer instructions is undefined. To obtain the correct result, insert two or more other instructions between the MFHI, MFLO instructions and the DDIVU instruction. This operation is defined in the 64-bit mode and the 32-bit kernel mode. When this instruction is executed in the 32-bit user/supervisor mode, a reserved instruction exception is generated.

(3) Jump and branch instructions

Jump and branch instructions change the control flow of a program.

All Jump instructions occur with a one-instruction delay. That is, the instruction immediately following the jump is always executed.

Branch instructions do not have a delay slot. If a branch is taken, the instruction immediately following the branch is never executed. If the branch is not taken, the instruction immediately following the branch is always executed.

Table 4-19 shows the MIPS16 Jump and branch instructions.

Table 4-19. Jump and Branch Instructions (1/2)

Instruction	Format and Description
Jump and Link	JAL target 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 immediately following the delay slot is placed in register ra. The ISA Mode bit is left unchanged. The value stored in ra bit 0 will reflect the current ISA Mode bit.
Jump and Link Exchange	JALX target 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 immediately following the delay slot is placed in register ra. The ISA Mode bit is inverted with a delay of one instruction. The value stored in ra bit 0 will reflect the ISA Mode bit before execution of the Jump execution.
Jump Register	JR rx The program unconditionally jumps to the address specified in general register rx, with a delay of one instruction. The instruction sets the ISA Mode bit to the value in rx bit 0. If the Jump target address is in the MIPS16 instruction length mode, no address exception occurs when bit 0 of the source register is 1 because bit 0 of the target address is 0 so that the instruction is located at the halfword boundary. If the 32-bit length instruction mode is changed, an address exception occurs when the jump target address is fetched if the two low-order bits of the target address are not 0.
	JR ra The program unconditionally jumps to the address specified in register ra, with a delay of one instruction. The instruction sets the ISA Mode bit to the value in ra bit 0. If the Jump target address is in the MIPS16 instruction length mode, no address exception occurs when bit 0 of the source register is 1 because bit 0 of the target address is 0 so that the instruction is located at the halfword boundary. If the 32-bit length instruction mode is changed, an address exception occurs when the jump target address is fetched if the two low-order bits of the target address are not 0.
Jump and Link Register	JALR ra, rx The program unconditionally jumps to the address contained in register rx, with a delay of one instruction. This instruction sets the ISA Mode bit to the value in rx bit 0. The address of the instruction immediately following the delay slot is placed in register ra. The value stored in ra bit 0 will reflect the ISA mode bit before the jump execution is executed. If the Jump target address is in the MIPS16 instruction length mode, no address exception occurs when bit 0 of the source register is 1 because bit 0 of the target address is 0 so that the instruction is located at the halfword boundary. If the 32-bit length instruction mode is changed, an address exception occurs when the jump target address is fetched if the two low-order bits of the target address are not 0.

Table 4-19. Jump and Branch Instructions (2/2)

Instruction	Format and Description
Branch on Equal to Zero	BEQZ rx, immediate The 8-bit immediate is shifted left one bit, sign extended, and then added to the address of the instruction after the branch to form the target address. If the contents of general register rx are equal to zero, the program branches to the target address. No delay slot is generated.
Branch on Not Equal to Zero	BNEZ rx, immediate The 8-bit immediate is shifted left one bit, sign extended, and then added to the address of the instruction after the branch to form the target address. If the contents of general register rx are not equal to zero, the program branches to the target address. No delay slot is generated.
Branch on T Equal to Zero	BTEQZ immediate The 8-bit immediate is shifted left one bit, sign extended, and then added to the address of the instruction after the branch to form the target address. If the contents of special register T ($24) are equal to zero, the program branches to the target address. No delay slot is generated.
Branch on T Not Equal to Zero	BTNEZ immediate The 8-bit immediate is shifted left one bit, sign extended, and then added to the address of the instruction after the branch to form the target address. If the contents of special register T ($24) are not equal to zero, the program branches to the target address. No delay slot is generated.
Branch Unconditional	B immediate The 11-bit immediate is shifted left one bit, sign extended, and then added to the address of the instruction after the branch to form the target address. The program branches to the target address unconditionally. No delay slot is generated.

(4) Special Instructions

Special instructions unconditionally perform branching to general exception vectors. Special instructions are of the R type. Table 4-20 shows two special instructions.

Table 4-20. Special Instructions

Instruction	Format and Description
Breakpoint	BREAK immediate A breakpoint trap occurs, unconditionally transferring control to the exception handler. By using a 6-bit code area, parameters can be sent to the exception handler. If the exception handler uses this parameter, the contents of memory including instructions must be loaded as data.
Extend	EXTEND immediate The 11-bit immediate is combined with the immediate in the next instruction to form a larger immediate equivalent to 32-bit MIPS. The Extend instruction must always precede (by one instruction) the instruction whose immediate field you want to extend. Every extended instruction consumes four bytes in program memory instead of two bytes (two bytes for Extend and two bytes for the instruction being extended), and it can cross a word boundary. (For details, see **4.8.2 Extend Instruction.**)

[MEMO]

CHAPTER 5 VR4121 PIPELINE

This chapter describes the basic operation of the VR4121 processor pipeline.

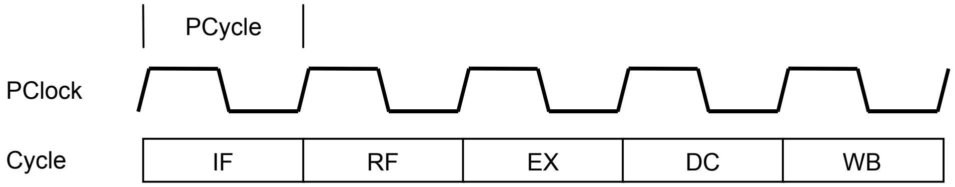
5.1 Pipeline Stages

The pipeline is controlled by PClock, and one cycle of this PClock is called PCycle. Each pipeline stage takes one PCycle.

5.1.1 Pipeline in MIPS III (32-bit length) instruction mode

The VR4121 has a five-stage instruction pipeline when in MIPS III instruction mode. 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. Once the pipeline has been filled, five instructions are executed simultaneously.

Figure 5-1. Pipeline Stages (MIPS III Instruction Mode)

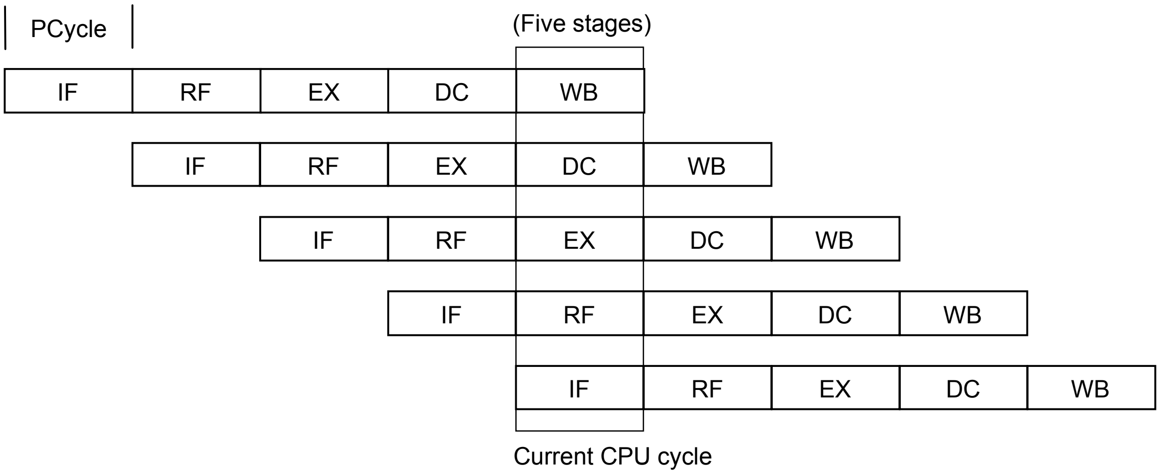


The five pipeline stages are:

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

Figure 5-2 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.

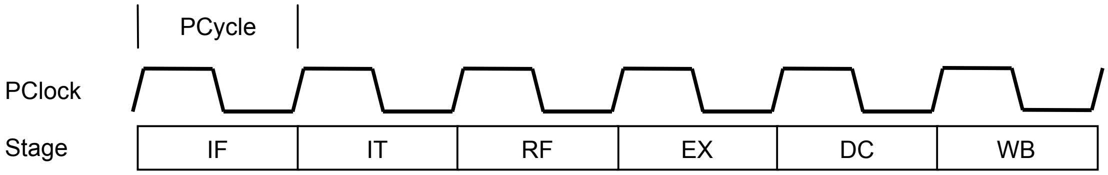
Figure 5-2. Instruction Execution in Pipeline



5.1.2 Pipeline in MIPS16 (16-bit length) instruction mode

The VR4121, which has internal stages for converting MIPS16 instructions into 32-bit instructions, uses a 6-stage pipeline in the MIPS16 mode. As a result, the execution of each instruction requires at least 6 PCycles. If the required data is not cached and must be retrieved from the main memory, a larger number of cycles will be required. If the pipeline flow is smooth, six instructions are executed simultaneously.

Figure 5-3. Pipeline Stages (MIPS16 Instruction Mode)

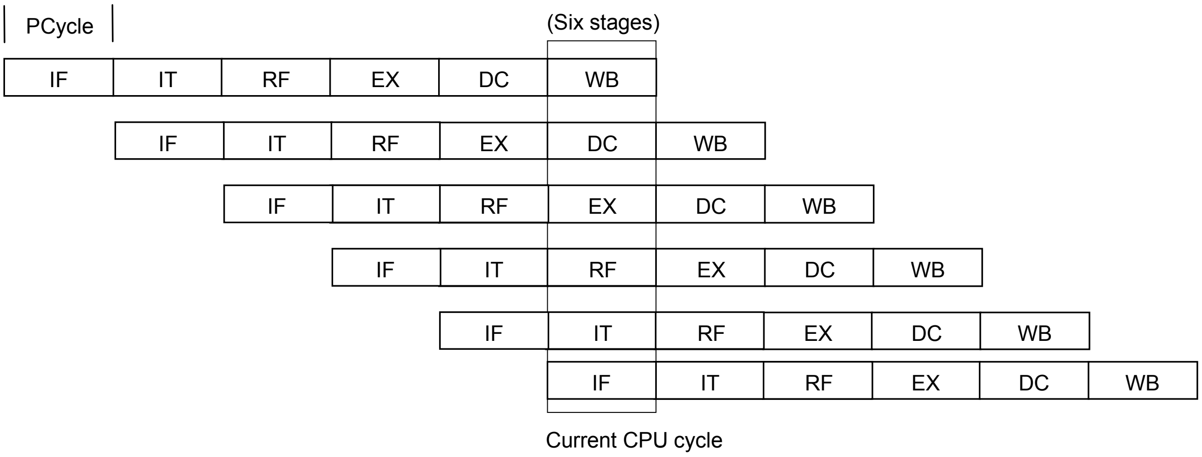


The pipeline stage names and their meaning are described below.

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

Figure 5-4 shows the outline of the pipeline. Each row in this figure indicates the execution process of an instruction, and the column indicates the 6 processes that are executed simultaneously.

Figure 5-4. Instruction Execution in Pipeline (MIPS16 Instruction Mode)



5.1.3 Pipeline activities

Figure 5-5 shows the activities that can occur during each pipeline stage; Table 5-1 describes these pipeline activities. Note that the IT stage occurs during MIPS16 instruction mode.

Figure 5-5. Pipeline Activities

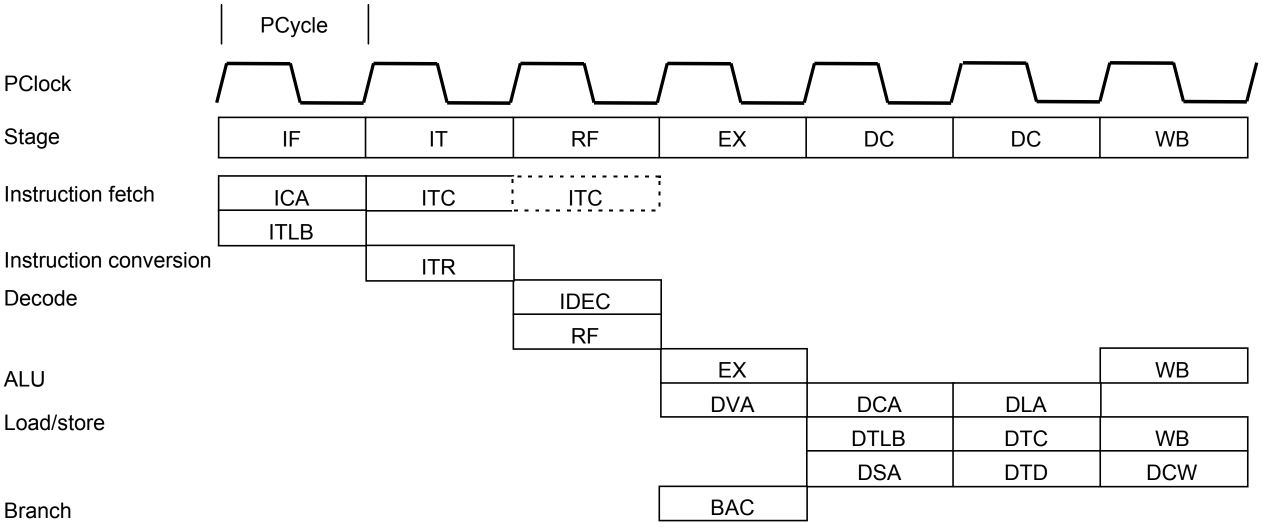


Table 5-1. Operation in Each Stage of Pipeline

Cycle	Mnemonic	Description
IF	ITLB	Instruction address conversion
	ICA	Instruction cache array access
IT	ITR	Instruction conversion
	ITC	Instruction tag fetch
RF	IDEC	Instruction decode
	RF	Register operand fetch
	ITC	Instruction tag check (in MIPS III instruction mode)
EX	EX	Execution stage
	BAC	Branch address calculation
	DVA	Data virtual address calculation
DC	DLA	Data cache load align
	DSA	Store align
	DCA	Data cache address decode/array access
	DTLB	Data address conversion
	DTC	Data tag check
	DTD	Data transfer to data cache
WB	DCW	Write to data cache
	WB	Write back to register file

5.2 Branch Delay

During a VR4121's pipeline operation, a one-cycle branch delay occurs when:

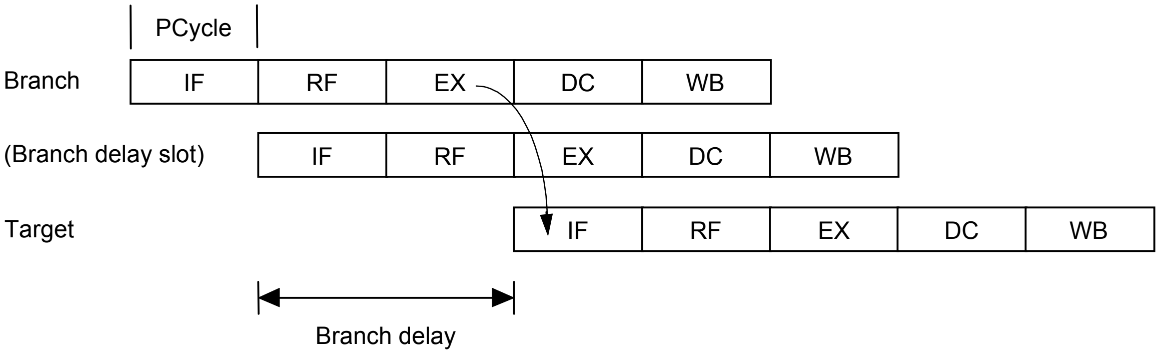
- Target address is calculated by a jump instruction
- Branch condition of branch instruction is met and then logical operation starts for branch-destination comparison

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 5-6 illustrates the branch delay and the location of the branch delay slot during MIPS III instruction mode.

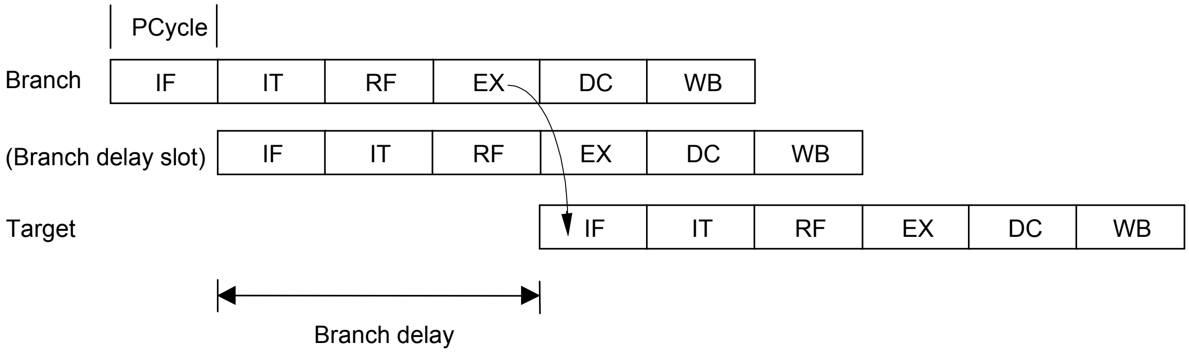
Figure 5-6. Branch Delay (In MIPS III Instruction Mode)



★ In the MIPS16 instruction mode, 3 branch delay cycles occur. If the branch condition of the branch instruction is satisfied, the instruction in the delay slot is discarded. However, in a jump instruction, one instruction in the delay slot is executed.

Figure 5-7 shows the image of branch delay in the MIPS16 instruction mode and the location of the branch delay slot.

Figure 5-7. Branch Delay (In MIPS16 Instruction Mode)



5.3 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 VR4121, 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.

5.4 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.

★ (1) Add instruction (ADD rd, rs, rt)

IF stage The eleven least-significant bits of the virtual address are used to access the instruction cache and read cache data. The virtual PC is incremented by 4 so that the next instruction can be fetched.

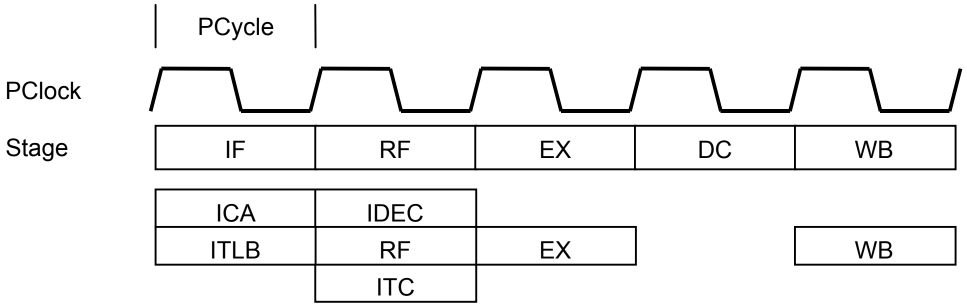
RF stage The 2-port register file is addressed with the rs and rt fields and the register data is valid at the register file output. At the same time, bypass multiplexers select inputs from either the EX-stage, DC-stage, or register file output, depending on the need for an operand bypass.

EX stage The ALU performs the operation.

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 The data is written into the register file.

Figure 5-8. ADD Instruction Pipeline Activities (In MIPS III Instruction Mode)



(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 and 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 (PC + 4 during MIPS16 instruction mode) 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 (PC + 4 during MIPS16 instruction mode) value is moved from the link output latch to the output latch of the EX stage.

DC stage The PC + 8 (PC + 4 during MIPS16 instruction mode) 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 used the same register addressed by rs; if it is, the result of executing such an instruction is undefined.

Figure 5-9. JALR Instruction Pipeline Activities (In MIPS III Instruction Mode)

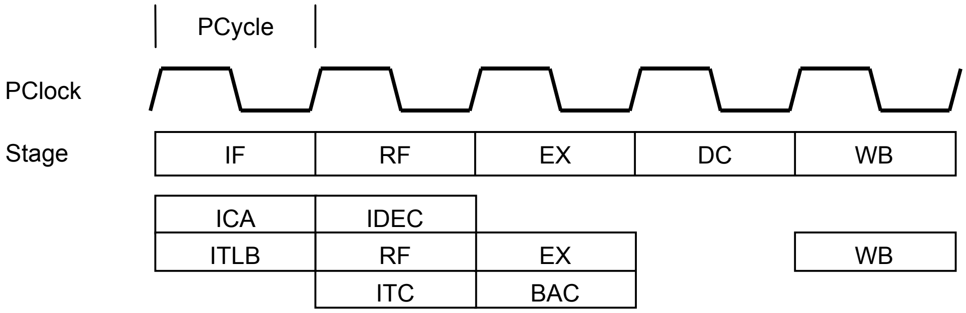
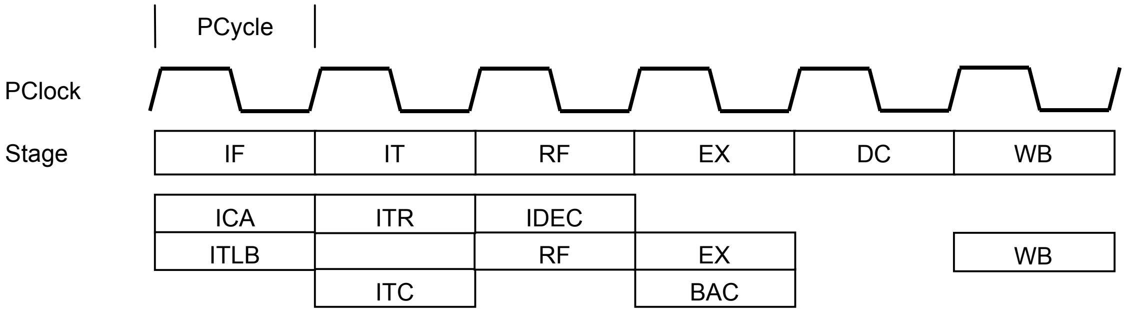


Figure 5-10. JALR Instruction Pipeline Activities (In MIPS16 Instruction Mode)



★ (3) Branch on Equal instruction (BEQ rs, rt, offset)

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

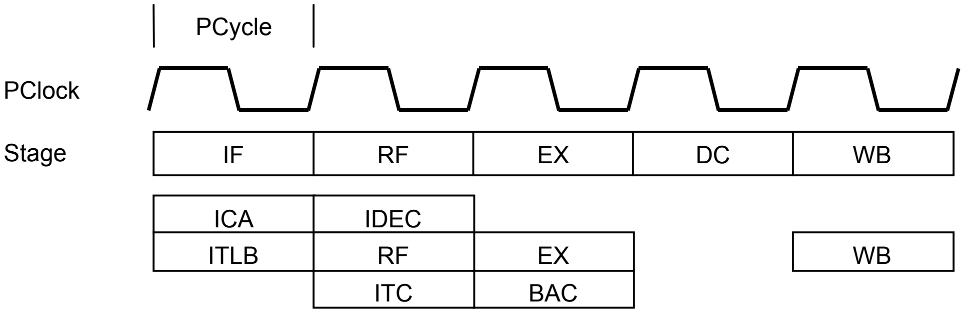
RF stage The register file is addressed with the rs and rt fields. A check is performed to determine if each corresponding bit position of these two operands has equal values. If they are equal, the PC is set to PC + target, where target is the sign-extended offset field. If they are not equal, the PC is set to PC + 4.

EX stage The next PC resulting from the branch comparison is valid at the beginning of instruction fetch.

DC stage This stage is a NOP for this instruction.

WB stage This stage is a NOP for this instruction.

Figure 5-11. BEQ Instruction Pipeline Activities (In MIPS III Instruction Mode)



(4) Trap if Less Than instruction (TLT rs, rt)

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

IT stage MIPS16 instruction set does not have TLT instruction, therefore this stage does not occur.

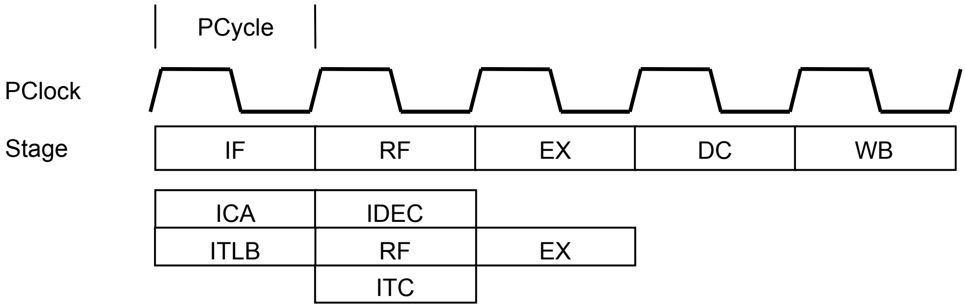
RF stage Same as the RF stage for the ADD instruction.

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. The signed 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 is performed for this instruction.

WB stage The EPC register is loaded with the value of the PC 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 5-12. TLT Instruction Pipeline Activities



(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 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.

DC2 stage The data is aligned and placed in the DC2 output latch.

WB stage The cache read data is written into the register file addressed by the rt field.

Figure 5-13. LW Instruction Pipeline Activities (In MIPS III Instruction Mode)

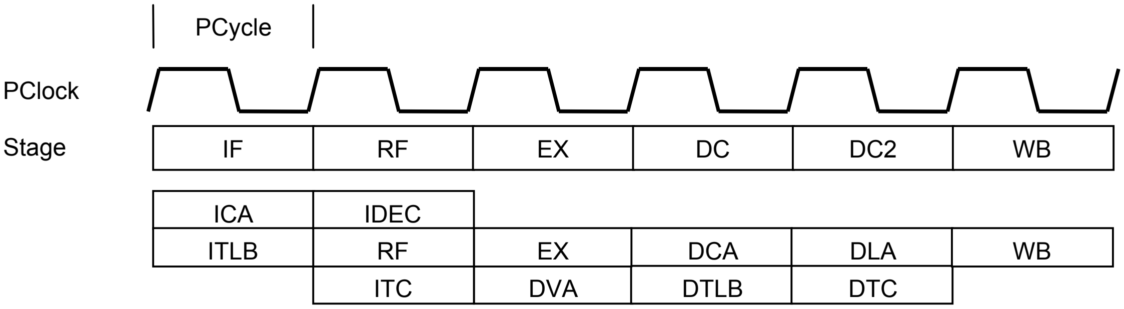
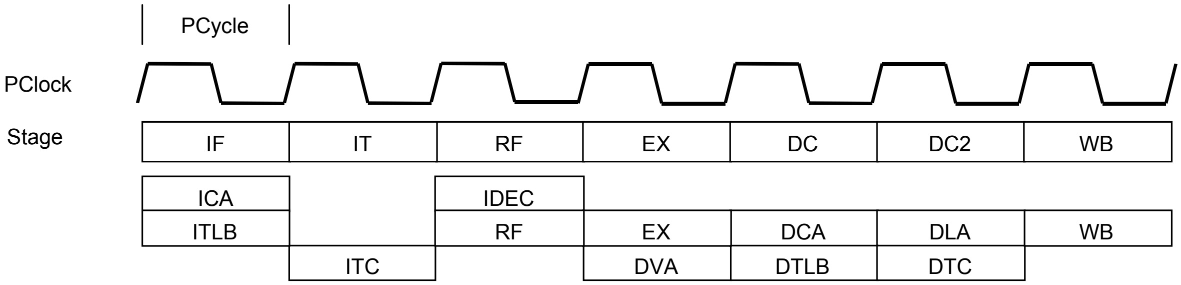


Figure 5-14. LW Instruction Pipeline Activities (In MIPS16 Instruction Mode)



(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] (store data) is sent through the bypass multiplexer and into the main shifter.

DC stage Refer to the LW instruction for a description of the cache access. Store data is aligned.

DC2 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 5-15. SW Instruction Pipeline Activities (In MIPS III Instruction Mode)

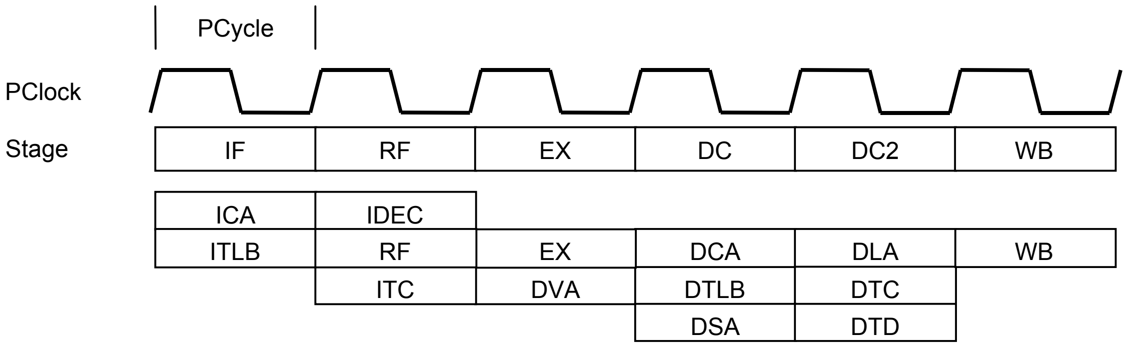
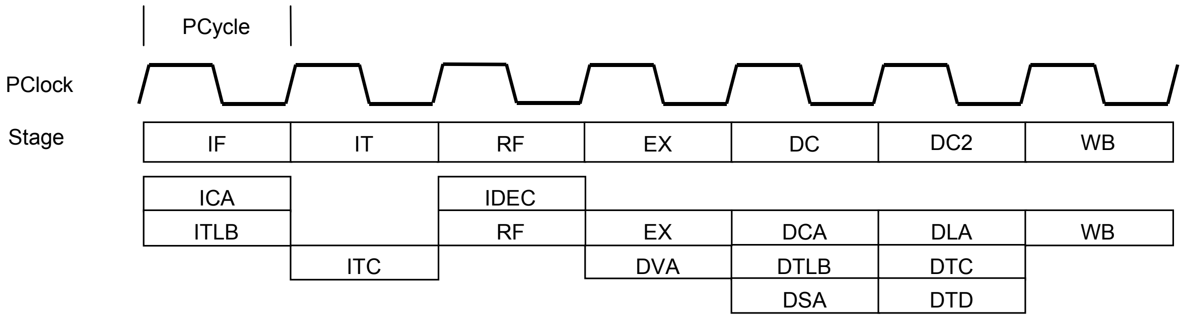


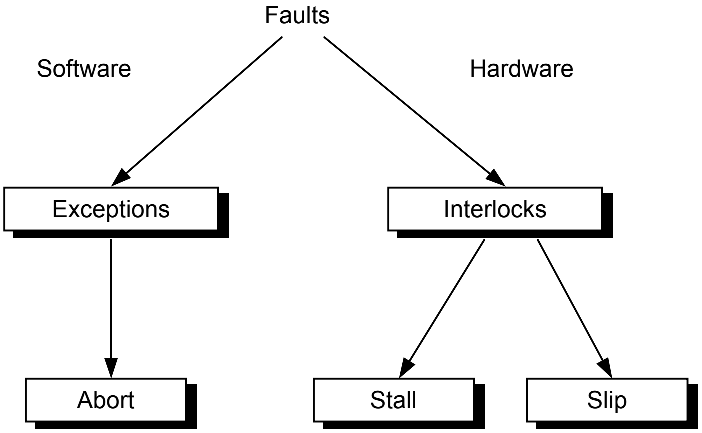
Figure 5-16. SW Instruction Pipeline Activities (In MIPS16 Instruction Mode)



5.5 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 5-17, all interlock and exception conditions are collectively referred to as faults.

Figure 5-17. Relationship between 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 5-2. For instance, an LDI Interlock is raised in the Register Fetch (RF) stage.

Tables 5-3 and 5-4 describe the pipeline interlocks and exceptions listed in Table 5-2.

Table 5-2. 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		–	NMI IAErr ITLB INTr IBE	SYSC BP CUn RSVD	Reset Trap OVF DAErr DTLB TMod WAT DBE	–

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

Table 5-3. 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 5-4. Description of Pipeline Exception

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

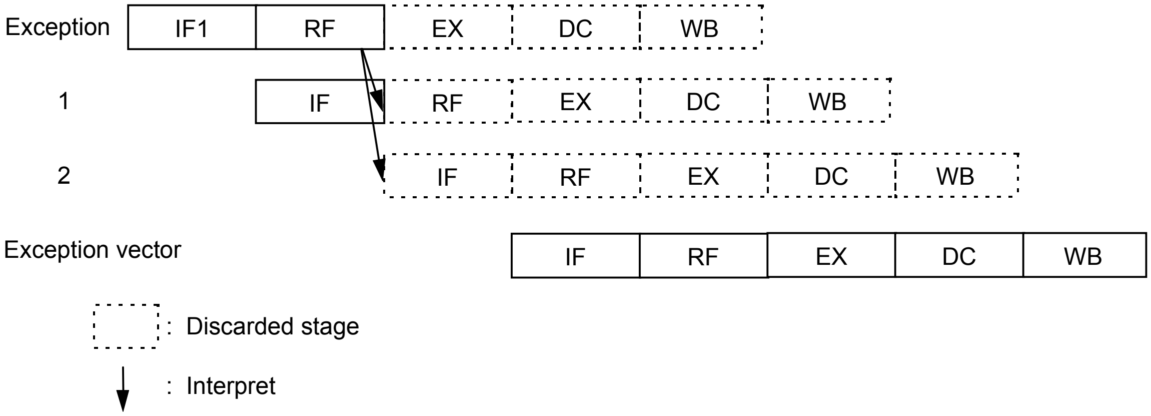
5.5.1 Exception conditions

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 VR4121 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.

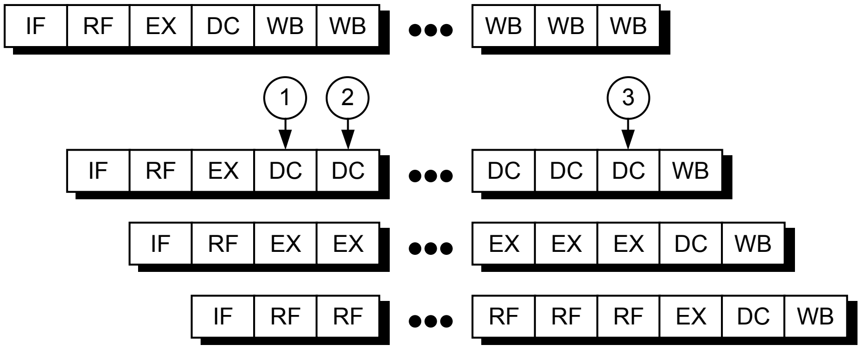
Figure 5-18. Exception Detection



5.5.2 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 5-19 shows a data cache miss stall, and Figure 5-20 shows a CACHE instruction stall.

Figure 5-19. 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

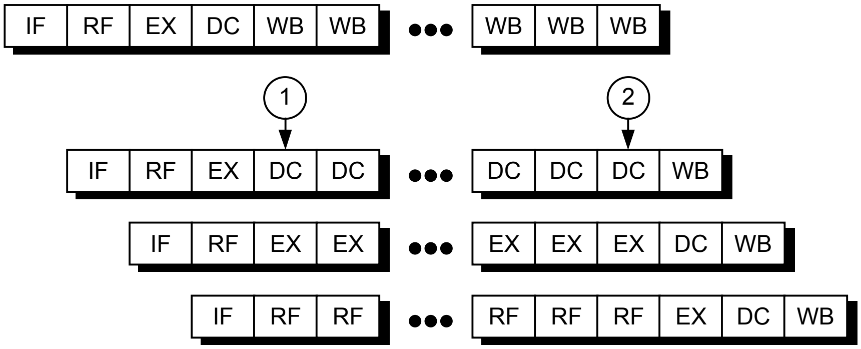
(1) Detect data cache miss

(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 5-20. 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

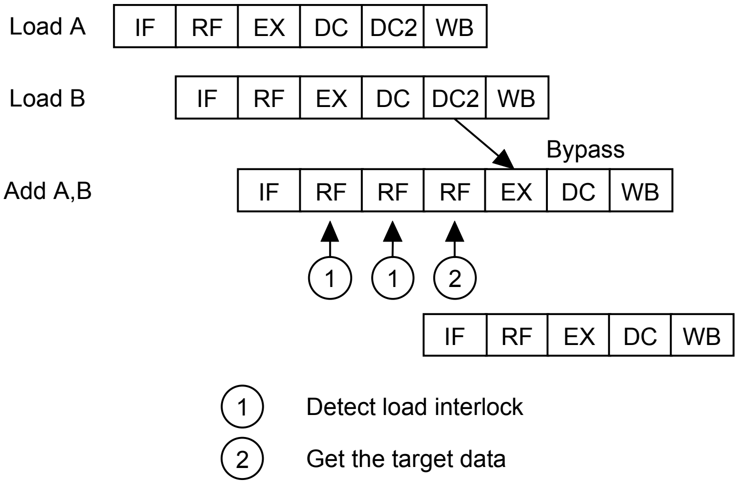
(2) CACHE instruction complete

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.

5.5.3 Slip conditions

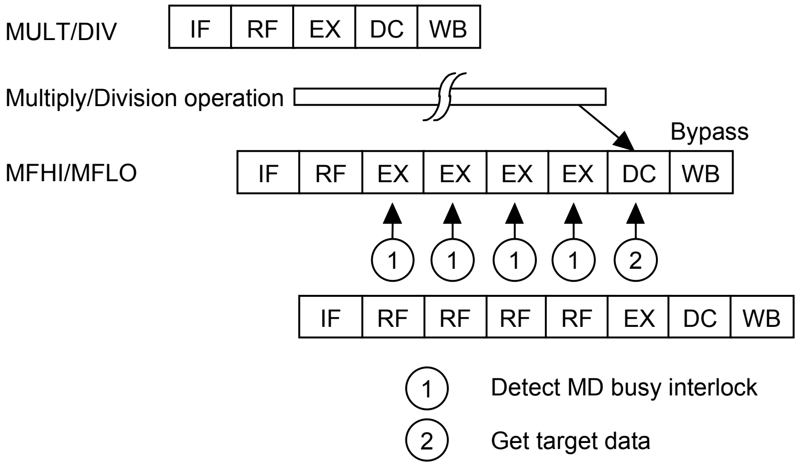
Between the RF stage and the EX stage, internal logic will determine whether it is possible to start the current instruction in this cycle. If all of the source operands are available (either from the register file or via the internal bypass logic) and all the hardware resources necessary to complete 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 space in the pipeline. Instructions suspended by branch likely instructions, ERET or exceptions will not cause slips.

Figure 5-21. Load Data Interlock



Load Data Interlock is detected in the RF stage shown in as Figure 5-21 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 5-22. MD Busy Interlock



MD Busy Interlock is detected in the RF stage as shown in Figure 5-22 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.

5.5.4 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.

5.6 Program Compatibility

The VR4120 CPU core is designed taking into consideration program compatibility with other VR-Series processors. However, because the VR4120 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 VR4120 will not necessarily run on other processors. Matters which should be paid attention to when porting programs between the VR4120 CPU core and other VR-Series processors are listed below.

- The VR4120 CPU core does not support floating-point instructions since it has no Floating-Point Unit (FPU).
- ★ 32-bit multiply-add instructions (DMACC, MACC) are added in the VR4120 CPU core. 16-bit multiply-add instructions (DMADD16, MADD16), which are supported in VR4100 or VR4110 cores, are upwardly compatible the 32-bit multiply-add instructions.
- Instructions for power modes (HIBERNATE, STANDBY, SUSPEND) are added in the VR4120 CPU core to support power modes.
- The VR4120 CPU core 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 VR4120 CPU core.
- The CP0 hazards of the VR4120 CPU core are equally or less stringent than those of other processors (for details, see **CHAPTER 30 VR4121 COPROCESSOR 0 HAZARDS**).
- An instruction for debug has been added for the VR4120 CPU core. However, this instruction cannot be used for the VR4121 CPU core.

For more information, refer to **CHAPTER 4 MIPS16 INSTRUTION SET**, **CHAPTER 28 MIPS III INSTRUCTION SET DETAILS**, the VR4100, VR4111 User's Manual, or the VR4300$^{\text{TM}}$ User's Manual.

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

Table 5-5. VR Series Supported Instructions

Product Instruction		VR4100 VR4102$^{\text{TM}}$	VR4111	VR4121	VR4300 VR4305$^{\text{TM}}$ VR4310$^{\text{TM}}$	VR5000 VR10000$^{\text{TM}}$
MIPS I instruction set		○	○	○	○	○
MIPS II instruction set		○	○	○	○	○
MIPS III instruction set		○	○	○	○	○
	LL bit operation	×	×	×	○	○
MIPS IV instruction set		×	×	×	×	○
MIPS16 instruction set		×	○	○	×	×
16-bit multiply-add operation		○	○	○ (Use of 32-bit multiply-add operation	×	×
32-bit multiply-add operation		×	×	○	×	×
Floating-point operation		×	×	×	○	○
Power mode transfer		○	○	○	×	×

CHAPTER 6 MEMORY MANAGEMENT SYSTEM

The VR4121 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.

6.1 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.

6.2 Virtual Address Space

The address space of the CPU is extended in memory management system, by converting (translating) huge virtual memory addresses into physical addresses.

The physical address space of the VR4121 is 4 Gbytes and 32-bit width addresses are used.

For the virtual address space, up to 2 Gbytes (2^{31}) are provided as a user's area and 32-bit width addresses are used in the 32-bit mode. In the 64-bit mode, up to 1 Tbyte (2^{40}) is provided as a user's area and 64-bit width addresses are used. For the format of the TLB entry in each mode, refer to **6.4.1 Format of a TLB entry**.

As shown in **Figures 6-2** and **6-3**, 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, and the Global (G) bit is in the EntryLo0 and EntryLo1 registers, described later in this chapter.

6.2.1 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 **6.5.11 Virtual-to-Physical Address Translation**.

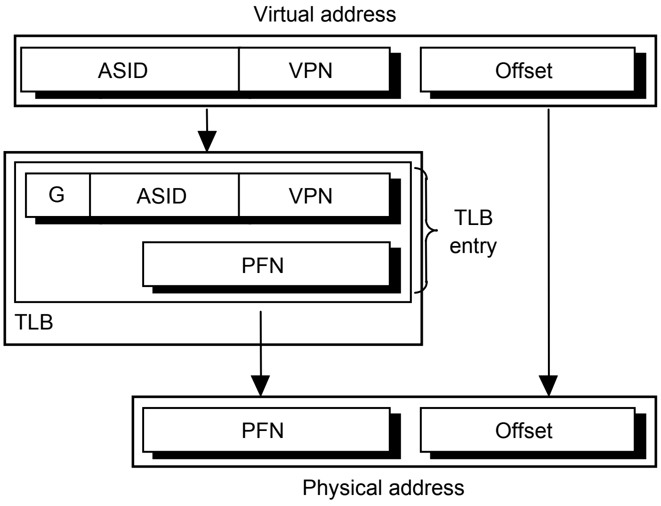
The next two sections describe the 32-bit and 64-bit mode address translations.

Figure 6-1. 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.

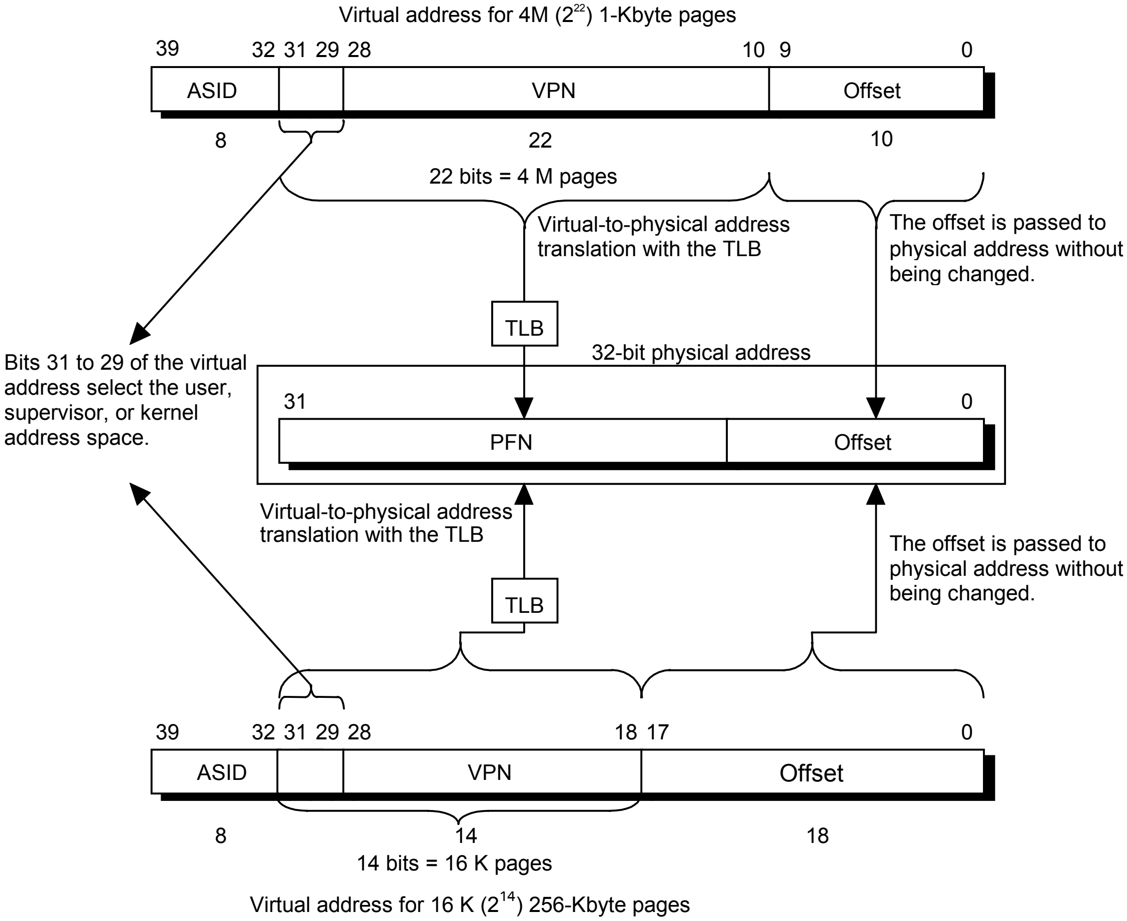


6.2.2 32-bit mode address translation

Figure 6-2 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 6-2 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 6-2 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 6-2. 32-Bit Mode Virtual Address Translation

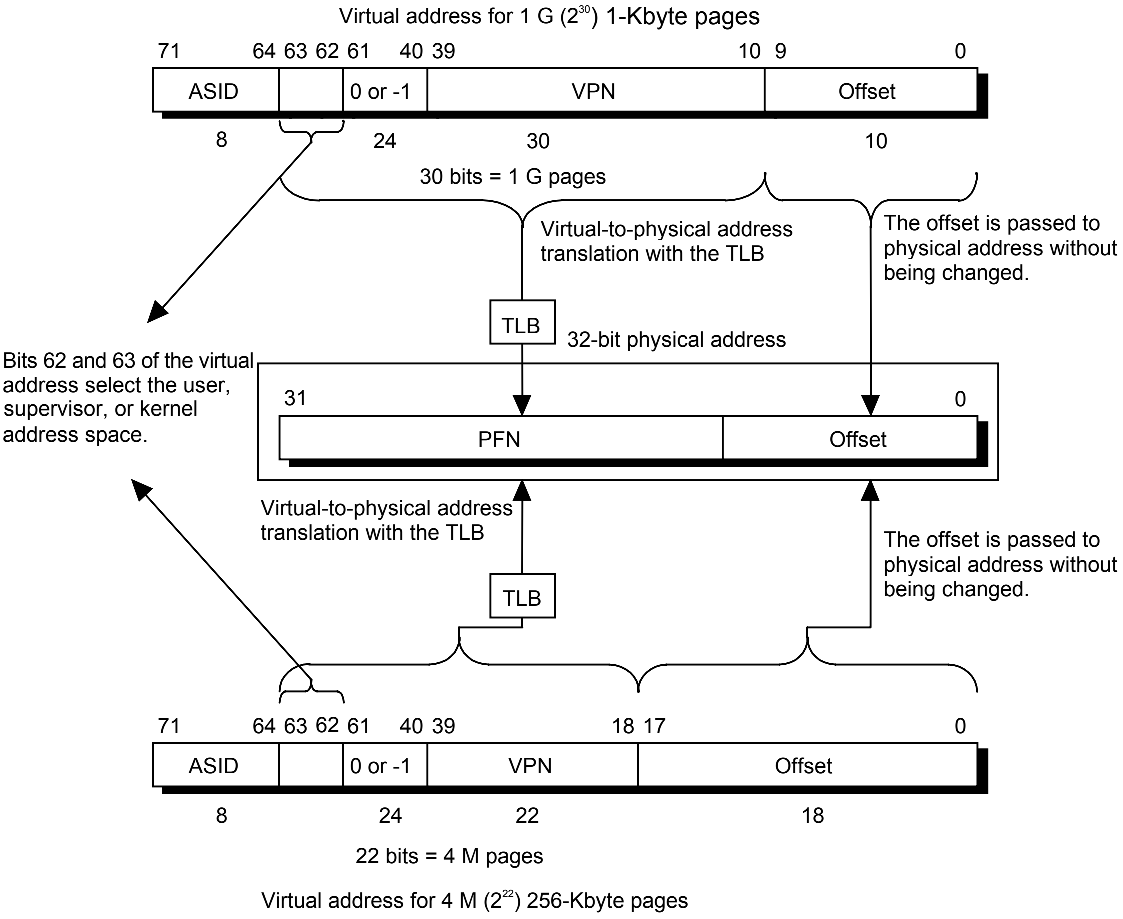

Virtual address for 4M (2^{22}) 1-Kbyte pages
39
32 31 29 28
10 9
0
ASID
VPN
Offset
8
22
10
22 bits = 4 M pages
Virtual-to-physical address translation with the TLB
The offset is passed to physical address without being changed.
TLB
Bits 31 to 29 of the virtual address select the user, supervisor, or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

6.2.3 64-bit mode address translation

Figure 6-3 shows the virtual-to-physical-address translation of a 64-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. This figure illustrates the two possible page sizes: a 1-Kbyte page (10 bits) and a 256-Kbyte page (18 bits).

- Shown at the top of Figure 6-3 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 6-3 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 6-3. 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

6.2.4 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.

6.2.5 User mode virtual addressing

During the single user mode, a 2-Gbyte (2^{31} bytes) virtual address space (useg) can be used in the 32-bit mode. In the 64-bit mode, a 1-Tbyte (2^{40} bytes) virtual address space (xuseg) can be used.

As shown in Tables 6-2 and 6-3, each virtual address is extended independently as another virtual address by setting an 8-bit address space ID area (ASID), to support user processes of up to 256. The contents of TLB can be retained after context switching by allocating each process by ASID. useg and xuseg can be referenced via TLB. Whether a cache is used or not is determined for each page by the TLB entry (depending on the C bit setting in the TLB entry).

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:

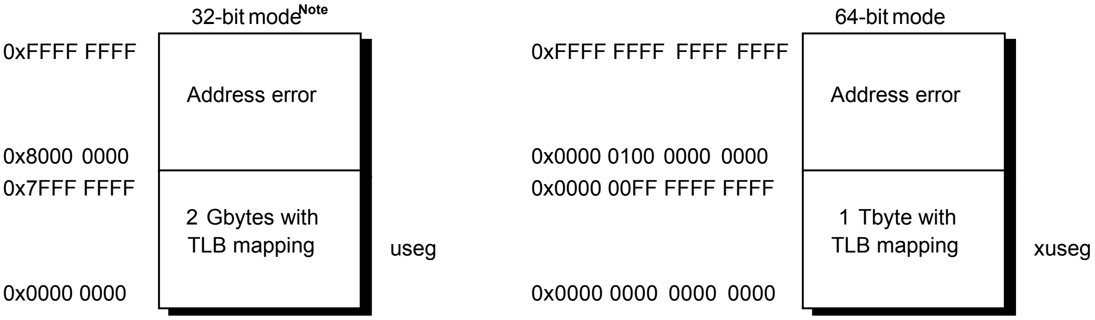
- KSU = 10
- EXL = 0
- ERL = 0

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 6-1 lists the characteristics of each user segment (useg and xuseg).

Figure 6-4. User Mode Address Space



Note The VR4121 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.

Table 6-1. Comparison of useg and xuseg

Address Bit Value	Status Register Bit Value				Segment Name	Address Range	Size
	KSU	EXL	ERL	UX			
32-bit A31 = 0	10	0	0	0	useg	0x0000 0000 to 0x7FFF FFFF	2 Gbytes (2^{31} bytes)
64-bit A(63:40) = 0	10	0	0	1	xuseg	0x0000 0000 0000 0000 to 0x0000 00FF FFFF FFFF	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 **CHAPTER 7 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 **CHAPTER 7 EXCEPTION PROCESSING**).

The XTLB Mismatch exception vector is used for TLB misses.

6.2.6 Supervisor-mode virtual addressing

Supervisor mode shown in Figure 6-5 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.

All of the suseg, sseg, xsuseg, xsseg, and csseg spaces are referenced via TLB. Whether cache can be used or not is determined by bit C of each page's TLB entry.

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

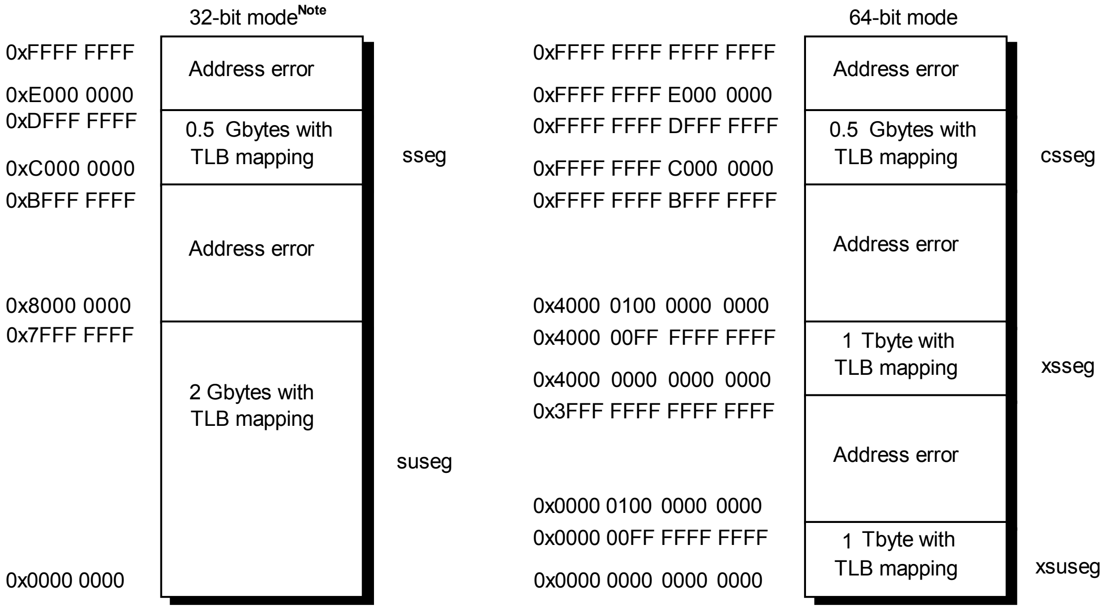
- KSU = 01
- EXL = 0
- ERL = 0

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 6-5 shows the supervisor mode address mapping, and Table 6-2 lists the characteristics of the Supervisor mode segments.

Figure 6-5. Supervisor Mode Address Space



Note The VR4121 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 6-2. 32-Bit and 64-Bit Supervisor Mode Segments

Address Bit	Status Register Bit Value				Segment	Address Range	Size
Value	KSU	EXL	ERL	SX	Name		
32-bit A31 = 0	01	0	0	0	suseg	0x0000 0000 to 0x7FFF FFFF	2 Gbytes (2^{31} bytes)
32-bit A(31:29) = 110	01	0	0	0	sseg	0xC000 0000 to 0xDFFF FFFF	512 Mbytes (2^{29} bytes)
64-bit A(63:62) = 00	01	0	0	1	xsuseg	0x0000 0000 0000 0000 to 0x0000 00FF FFFF FFFF	1 Tbyte (2^{40} bytes)
64-bit A(63:62) = 01	01	0	0	1	xsseg	0x4000 0000 0000 0000 to 0x4000 00FF FFFF FFFF	1 Tbyte (2^{40} bytes)
64-bit A(63:62) = 11	01	0	0	1	csseg	0xFFFF FFFF C000 0000 to 0xFFFF FFFF DFFF FFFF	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 0x0000 0000 and runs through 0x7FFF FFFF.

(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 0xC000 0000 and runs through 0xDFFF FFFF.

(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 0x0000 0000 0000 0000 and runs through 0x0000 00FF FFFF FFFF.

(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 0x4000 0000 0000 0000 and runs through 0x4000 00FF FFFF FFFF.

(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 0xFFFF FFFF C000 0000 and runs through 0xFFFF FFFF DFFF FFFF.

6.2.7 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

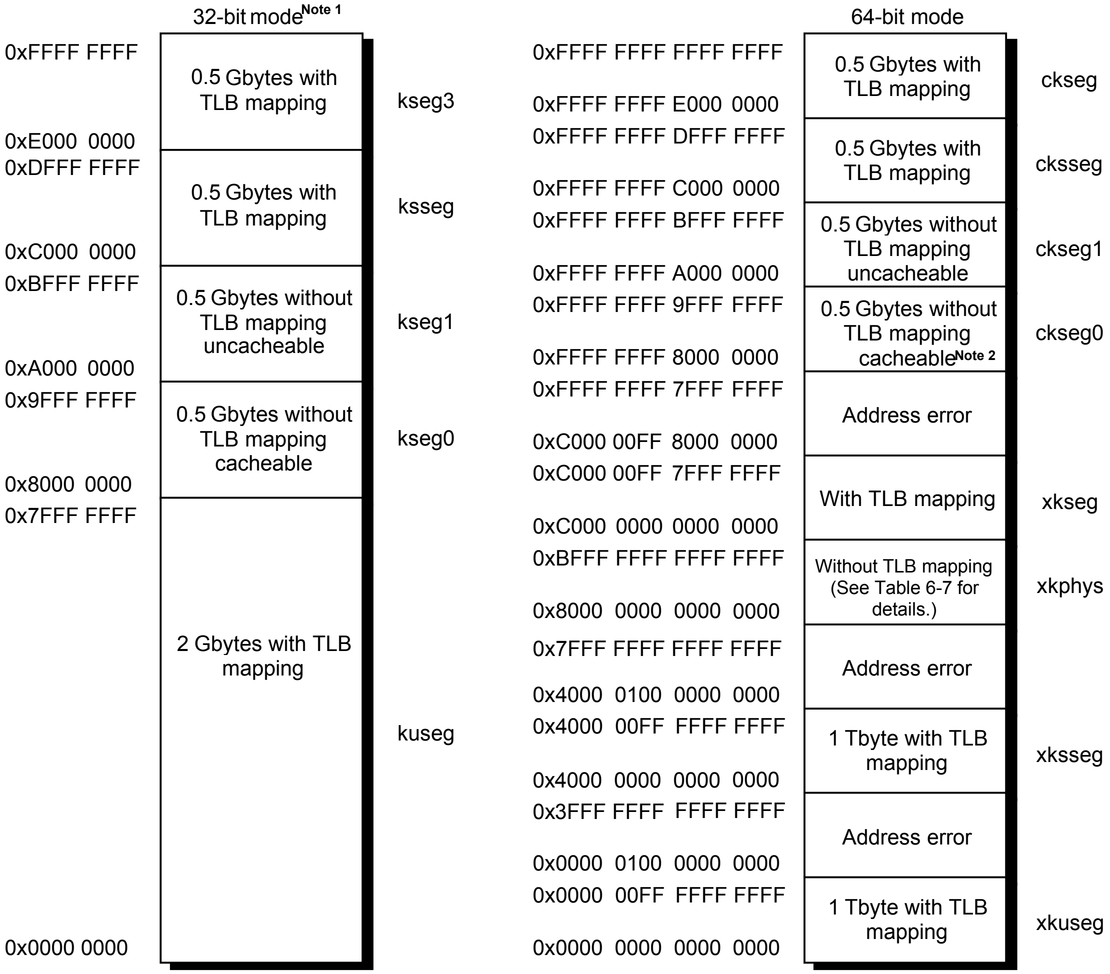
The addressing width in Kernel mode varies according to the state of the KX bit of the Status register, as follows:

- 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 6-6. Table 6-3 lists the characteristics of the 32-bit Kernel mode segments, and Table 6-4 lists the characteristics of the 64-bit Kernel mode segments.

Figure 6-6. Kernel Mode Address Space



Notes 1. The VR4121 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. 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

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

Figure 6-7. xkphys Area Address Space

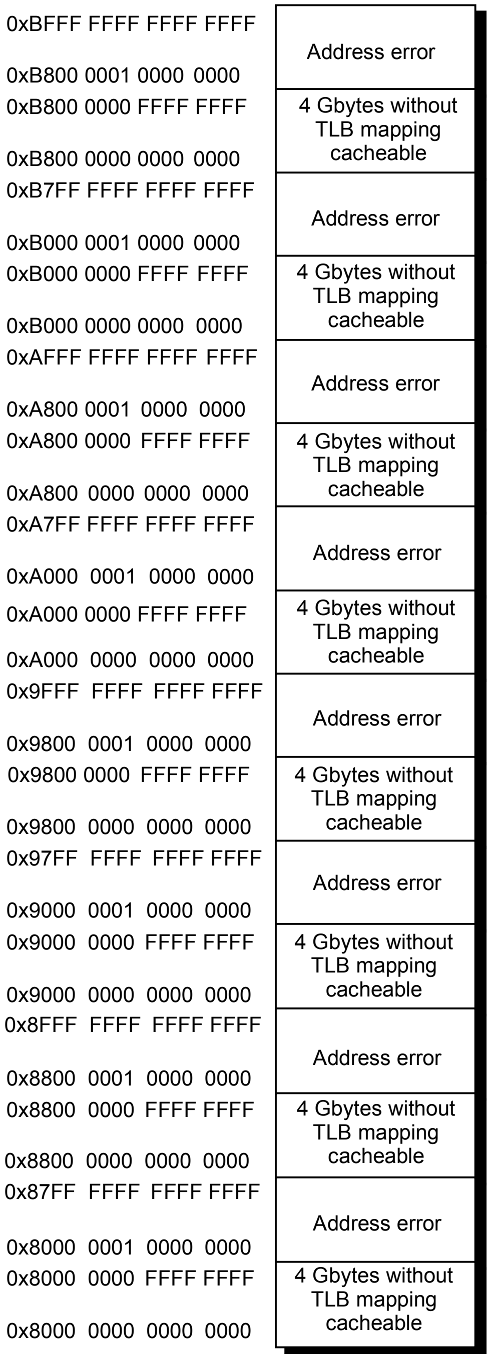


Table 6-3. 32-Bit Kernel Mode Segments

Address Bit Value	Status Register Bit Value				Segment	Virtual Address	Physical	Size
	KSU	EXL	ERL	KX	Name		Address	
A31 = 0	KSU = 00 or EXL = 1 or ERL = 1			0	kuseg	0x0000 0000 to 0x7FFF FFFF	TLB map	2 Gbytes (2^{31} bytes)
A(31:29) = 100				0	kseg0	0x8000 0000 to 0x9FFF FFFF	0x0000 0000 to 0x1FFF FFFF	512 Mbytes (2^{29} bytes)
A(31:29) = 101				0	kseg1	0xA000 0000 to 0xBFFF FFFF	0x0000 0000 to 0x1FFF FFFF	512 Mbytes (2^{29} bytes)
A(31:29) = 110				0	ksseg	0xC000 0000 to 0xDFFF FFFF	TLB map	512 Mbytes (2^{29} bytes)
A(31:29) = 111				0	kseg3	0xE000 0000 to 0xFFFF FFFF	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.

References to kuseg are mapped through TLB. Whether cache can be used or not is determined by bit C of each page's TLB entry.

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 0x8000 0000 from the virtual address.

The K0 field of the Config register controls cacheability (see **CHAPTER 7 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 0xA000 0000 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.

References to ksseg are mapped through TLB. Whether cache can be used or not is determined by bit C of each page's TLB entry.

(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.

References to kseg3 are mapped through TLB. Whether cache can be used or not is determined by bit C of each page's TLB entry.

Table 6-4. 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	0x0000 0000 0000 0000 to 0x0000 00FF FFFF FFFF	TLB map	1 Tbyte (2^{40} bytes)
A(63:62) = 01				1	xksseg	0x4000 0000 0000 0000 to 0x4000 00FF FFFF FFFF	TLB map	1 Tbyte (2^{40} bytes)
A(63:62) = 10				1	xkphys	0x8000 0000 0000 0000 to 0xBFFF FFFF FFFF FFFF	0x0000 0000 to 0xFFFF FFFF	4 Gbytes (2^{32} bytes)
A(63:62) = 11				1	xkseg	0xC000 0000 0000 0000 to 0xC000 00FF 7FFF FFFF	TLB map	2^{40} to 2^{31} bytes
A(63:62) = 11 A(63:31) = -1				1	ckseg0	0xFFFF FFFF 8000 0000 to 0xFFFF FFFF 9FFF FFFF	0x0000 0000 to 0x1FFF FFFF	512 Mbytes (2^{29} bytes)
A(63:62) = 11 A(63:31) = -1				1	ckseg1	0xFFFF FFFF A000 0000 to 0xFFFF FFFF BFFF FFFF	0x0000 0000 to 0x1FFF FFFF	512 Mbytes (2^{29} bytes)
A(63:62) = 11 A(63:31) = -1				1	cksseg	0xFFFF FFFF C000 0000 to 0xFFFF FFFF DFFF FFFF	TLB map	512 Mbytes (2^{29} bytes)
A(63:62) = 11 A(63:31) = -1				1	ckseg3	0xFFFF FFFF E000 0000 to 0xFFFF FFFF FFFF FFFF	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.

References to xkuseg are mapped through TLB. Whether cache can be used or not is determined by bit C of each page's TLB entry.

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.

References to xksseg are mapped through TLB. Whether cache can be used or not is determined by bit C of each page's TLB entry.

(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.

Whether cache can be used or not is determined by bits 59 to 61 of the virtual address. Table 6-5 shows cacheability corresponding to 8 address spaces.

Table 6-5. Cacheability and xkphys Address Space

Bits 61 to 59	Cacheability	Start Address
0	Cached	0x8000 0000 0000 0000 to 0x8000 0000 FFFF FFFF
1	Cached	0x8800 0000 0000 0000 to 0x8800 0000 FFFF FFFF
2	Uncached	0x9000 0000 0000 0000 to 0x9000 0000 FFFF FFFF
3	Cached	0x9800 0000 0000 0000 to 0x9800 0000 FFFF FFFF
4	Cached	0xA000 0000 0000 0000 to 0xA000 0000 FFFF FFFF
5	Cached	0xA800 0000 0000 0000 to 0xA800 0000 FFFF FFFF
6	Cached	0xB000 0000 0000 0000 to 0xB000 0000 FFFF FFFF
7	Cached	0xB800 0000 0000 0000 to 0xB800 0000 FFFF FFFF

(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
  References to xkseg are mapped through 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.

6.3 Physical Address Space

So VR4120 core uses a 32-bit address, that the processor physical address space encompasses 4 Gbytes. The VR4121 uses this 4-Gbyte physical address space as shown in Figure 6-8.

Figure 6-8. VR4121 Physical Address Space

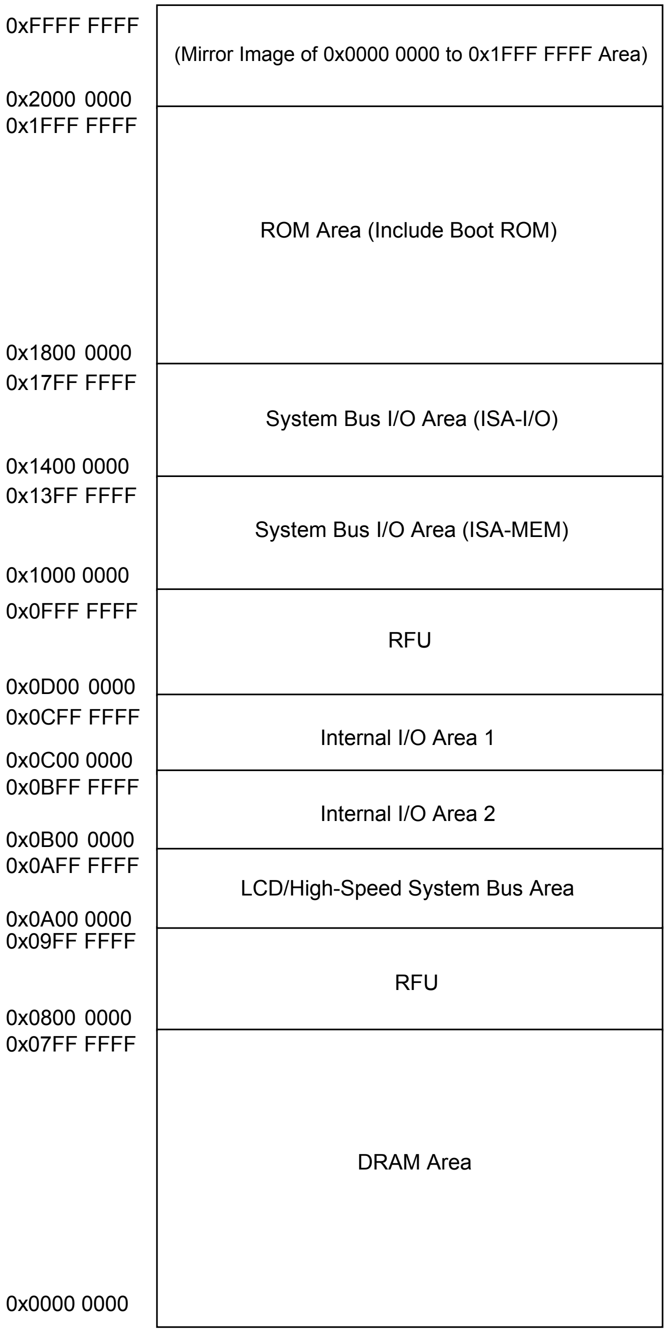


Table 6-6. VR4121 Physical Address Space

Physical Address	Space	Capacity (bytes)
0xFFFF FFFF to 0x2000 0000	Mirror image of 0x1FFF FFFF to 0x0000 0000	3.5 G
0x1FFF FFFF to 0x1800 0000	ROM space	128 M
0x17FF FFFF to 0x1400 0000	System bus I/O space (ISA-IO)Note	64 M
0x13FF FFFF to 0x1000 0000	System bus memory space (ISA-MEM)Note	64 M
0x0FFF FFFF to 0x0D00 0000	RFU	48 M
0x0CFF FFFF to 0x0C00 0000	Internal I/O space 1	16 M
0x0BFF FFFF to 0x0B00 0000	Internal I/O space 2	16 M
0x0AFF FFFF to 0x0A00 0000	LCD/high-speed system bus memory space	16 M
0x09FF FFFF to 0x0800 0000	RFU	32 M
0x07FF FFFF to 0x0000 0000	DRAM space	128 M

Note During an RTC reset, if usage of SDRAM and SROM has been set, the ADD25/SCLK pin becomes a clock output pin for memory and the address bus is a 25-bit bus, ADD(0:24). Accordingly, when specifying an address space for the system bus, caution is required since the address's low-order 32 Mbytes and high-order 32 Mbytes may become identical.

6.3.1 ROM address space

The VR4121's ROM space is divided into four banks. If the BCUCNTREG3's EXT_MEM bit has been set to enable use of expanded memory, the usable space is called expansion space. The following describes how banks are allocated according to the data bus width when using ordinary memory space and expansion space.

- During 16-bit bus mode: All banks are allocated in ordinary memory space.
- During 32-bit bus mode: Banks 0 and 1 are allocated in ordinary memory space and banks 2 and 3 are allocated in expansion space.

(1) Capacity of ROM space

The ROM address space varies in size depending on the data bus's bit width and the capacity of the ROM being used.

- The data bus bit width is set via the DBUS32 pin.
- The ROM capacity can be set in either of the following two ways.
  By setting the BCUCNTREG1 register's ROM64 bit or the BCUCNTREG3 register's EXT_ROM64 bit
  By setting the ROMSIZEREG register's SIZE bit

(a) Capacity setting via BCUCNTREG register

The capacity can be selected as 32 Mbits or 64 Mbits for either the ordinary memory space or the expansion space. Use the BCUCNTREG1 register's ROM64 bit when selecting the ordinary memory space and use the BCUCNTREG3 register's EXT_ROM64 bit when selecting the expansion space. However, the selectable range for SROM is limited.

(b) Capacity setting via ROMSIZEREG register

The capacity can be selected for each bank as 16, 64, or 128 Mbits. However, the selectable range for SROM is limited.

(2) Mapping of physical address

The area for each bank is determined based on the data bus bit width and the bank capacity, as described below.

(a) During 16-bit bus mode (DBUS32 = 0)

Bank 3 (ROMCS3#): 0x1FFF FFFF to 0x1FFF FFFF − (Bank 3 capacity)
Bank 2 (ROMCS2#): 0x1FFF FFFF − (Bank 3 capacity + 1 byte) to
0x1FFF FFFF − (Bank 3 capacity + Bank 2 capacity)
Bank 1 (ROMCS1#): 0x1FFF FFFF − (Bank 3 capacity + Bank 2 capacity + 1 byte) to
0x1FFF FFFF − (Bank 3 capacity + Bank 2 capacity + Bank 1 capacity)
Bank 0 (ROMCS0#): 0x1FFF FFFF − (Bank 3 capacity + Bank 2 capacity + Bank 1 capacity + 1 byte) to
0x1FFF FFFF − (Bank 3 capacity + Bank 2 capacity + Bank 1 capacity + Bank 0 capacity)

Physical addresses in the ROM address space are indicated below for both memory spaces, when using 32-Mbit ROM or 64-Mbit ROM.

Table 6-7. ROM Address Example (When Using 16-Bit Data Bus)

Physical Address	ADD(25:0) Pin	When Using 32-Mbit ROM (ROM64 = 0 or SIZE3 = SIZE2 = SIZE1 = SIZE0 = 1)	When Using 64-Mbit ROM (ROM64 = 1 or SIZE3 = SIZE2 = SIZE1 = SIZE0 = 2)
0x1FFF FFFF to 0x1FC0 0000	0x3FF FFFF to 0x3C0 0000	Bank 3 (ROMCS3#)	Bank 3 (ROMCS3#)
0x1FBF FFFF to 0x1F80 0000	0x3BF FFFF to 0x380 0000	Bank 2 (ROMCS2#)	
0x1F7F FFFF to 0x1F40 0000	0x37F FFFF to 0x340 0000	Bank 1 (ROMCS1#)	Bank 2 (ROMCS2#)
0x1F3F FFFF to 0x1F00 0000	0x33F FFFF to 0x300 0000	Bank 0 (ROMCS0#)	
0x1EFF FFFF to 0x1E80 0000	0x2FF FFFF to 0x280 0000	RFU	Bank 1 (ROMCS1#)
0x1E7F FFFF to 0x1E00 0000	0x27F FFFF to 0x200 0000		Bank 0 (ROMCS0#)
0x1DFF FFFF to 0x1800 0000	0x1FF FFFF to 0x000 0000		RFU

(b) During 32-bit bus mode (DBUS32 = 1)

Bank 1 (ROMCS1#): 0x1FFF FFFF to 0x1FFF FFFF – (Bank 1 capacity)
Bank 0 (ROMCS0#): 0x1FFF FFFF – (Bank 1 capacity + 1 byte) to
0x1FFF FFFF – (Bank 1 capacity + Bank 0 capacity)
Bank 3 (ROMCS3#): 0x1FFF FFFF – (Bank 1 capacity + Bank 0 capacity + 1 byte) to
0x1FFF FFFF – (Bank 1 capacity + Bank 0 capacity + Bank 3 capacity)
Bank 2 (ROMCS2#): 0x1FFF FFFF – (Bank 1 capacity + Bank 0 capacity + Bank 3 capacity + 1 byte) to
0x1FFF FFFF – (Bank 1 capacity + Bank 0 capacity + Bank 3 capacity + Bank 2 capacity)

Physical addresses in the ROM address space are indicated below for when 32-Mbit ROM (banks 3 and 2), 32-Mbit ROM (banks 1 and 0), or 64-Mbit ROM is used.

Table 6-8. Example of ROM Addresses When Using 32-Mbit Expansion ROM (32-Bit Data Bus)

Physical Address	ADD(25:0) Pin	When Using 32-Mbit ROM (ROM64 = 0, EXT_ROM64 = 0, or SIZE3 = SIZE2 = SIZE1 = SIZE0 = 2)	When Using 64-Mbit ROM (ROM64 = 1, EXT_ROM64 = 0, or SIZE1 = SIZE0 = 3, SIZE3 = SIZE2 = 2)
0x1FFF FFFF to 0x1F80 0000	0x3FF FFFF to 0x380 0000	Bank 1 (ROMCS1#)	Bank 1 (ROMCS1#)
0x1F7F FFFF to 0x1F00 0000	0x37F FFFF to 0x300 0000	Bank 0 (ROMCS0#)	
0x1EFF FFFF to 0x1E80 0000	0x2FF FFFF to 0x280 0000	Bank 3 (ROMCS3#)	Bank 0 (ROMCS0#)
0x1E7F FFFF to 0x1E00 0000	0x27F FFFF to 0x200 0000	Bank 2 (ROMCS2#)	
0x1DFF FFFF to 0x1D80 0000	0x1FF FFFF to 0x180 0000	RFU	Bank 3 (ROMCS3#)Note
0x1D7F FFFF to 0x1D00 0000	0x17F FFFF to 0x100 0000		Bank 2 (ROMCS2#)Note
0x1CFF FFFF to 0x1800 0000	0x0FF FFFF to 0x000 0000		RFU

Note Can be used exclusively from the expansion DRAM.

Physical addresses in the ROM address space are indicated below for when 64-Mbit ROM (banks 3 and 2), 32-Mbit ROM (banks 1 and 0), or 64-Mbit ROM is used.

Table 6-9. Example of ROM Addresses When Using 64-Mbit Expansion ROM (32-Bit Data Bus)

Physical Address	ADD(25:0) Pin	When Using 32-Mbit ROM (ROM64 = 0, EXT_ROM64 = 1, or SIZE1 = SIZE0 = 2, SIZE3 = SIZE2 = 3)	When Using 64-Mbit ROM (ROM64 = 1, EXT_ROM64 = 1, or SIZE3 = SIZE2 = SIZE1 = SIZE0 = 3)
0x1FFF FFFF to 0x1F80 0000	0x3FF FFFF to 0x380 0000	Bank 1 (ROMCS1#)	Bank 1 (ROMCS1#)
0x1F7F FFFF to 0x1F00 0000	0x37F FFFF to 0x300 0000	Bank 0 (ROMCS0#)	
0x1EFF FFFF to 0x1E00 0000	0x2FF FFFF to 0x200 0000	Bank 3 (ROMCS3#)	Bank 0 (ROMCS0#)
0x1DFF FFFF to 0x1D00 0000	0x1FF FFFF to 0x100 0000	Bank 2 (ROMCS2#)	Bank 3 (ROMCS3#)[Note]
0x1CFF FFFF to 0x1C00 0000	0x0FF FFFF to 0x000 0000	RFU	Bank 2 (ROMCS2#)[Note]
0x1BFF FFFF to 0x1800 0000	–		RFU

Note Can be used exclusively from the expansion DRAM.

Physical addresses in the ROM address space are indicated below for when 64-Mbit ROM or 128-Mbit ROM (all banks) is used.

★ **Table 6-10. Example of ROM Addresses When Using 128-Mbit Expansion ROM (32-Bit Data Bus)**

Physical Address	ADD(25:0) Pin	When Using 64-Mbit ROM (ROM64 = 1, EXT_ROM64 = 1, or SIZE3 = SIZE2 = SIZE1 = SIZE0 = 3)	When Using 128-Mbit ROM (SIZE3 = SIZE2 = SIZE1 = SIZE0 = 4)
0x1FFF FFFF to 0x1F00 0000	0x3FF FFFF to 0x300 0000	Bank 1 (ROMCS1#)	Bank 1 (ROMCS1#)
0x1EFF FFFF to 0x1E00 0000	0x2FF FFFF to 0x200 0000	Bank 0 (ROMCS0#)	
0x1DFF FFFF to 0x1D00 0000	0x1FF FFFF to 0x100 0000	Bank 3 (ROMCS3#)[Note]	Bank 0 (ROMCS0#)
0x1CFF FFFF to 0x1C00 0000	0xFF FFFF to 0x000 0000	Bank 2 (ROMCS2#)[Note]	
0x1BFF FFFF to 0x1A00 0000	0x3FF FFFF to 0x200 0000	RFU	Bank 3 (ROMCS3#)[Note]
0x19FF FFFF to 0x1800 0000	0x1FF FFFF to 0x000 0000		Bank 2 (ROMCS2#)[Note]

Note Can be used exclusively from the expansion DRAM.

6.3.2 System bus address space

The following three types of system bus address space are available.

- System bus I/O space[Note]
  This corresponds to the ISA's I/O space.

- System bus memory space[Note]
  This corresponds to the ISA's memory space.

Note During an RTC reset, if usage of SDRAM and SROM has been set, the ADD25/SCLK pin becomes a clock output pin for memory and the address bus is a 25-bit bus, ADD(0:24). Accordingly, when specifying an address space for the system bus, caution is required since the address's low-order 32 Mbytes and high-order 32 Mbytes may become identical.

- High-speed system bus memory space
  The access speed can be set independently of the system bus memory space.
  There are 16 Mbytes of high-speed system bus memory space. Therefore, the ADD(25:24) pin is fixed as 10.
  When system bus memory has been accessed from the high-speed system bus memory space, the LCDCS# pin becomes active.
  The high-speed system bus memory space is used exclusively from the LCD space. To switch between these two types of space, set the ISAM/LCD bit in BCUCNTREG1.
  ★ Data bus bit width is set with the DBUS32 pin and the LCD32/ISA32 bit in BCUCNTREG3.

  - DBUS32 pin = 0:16 bits
  - DBUS32 pin = 1
    LCD32/ISA32 in BCUCNTREG3 = 0:16 bits
    LCD32/ISA32 in BCUCNTREG3 = 1:32 bits

6.3.3 Internal I/O space

The VR4121 has two internal I/O spaces. Each of these spaces are described below.

Table 6-11. Internal I/O Space 1

Physical Address	Internal I/O
0x0CFF FFFF to 0x0C00 0080	RFU
0x0C00 007F to 0x0C00 0060	FIR2
0x0C00 005F to 0x0C00 0040	FIR
0x0C00 003F to 0x0C00 0020	HSP (Software modem interface)
0x0C00 001F to 0x0C00 0000	SIU (equivalent to 16550)

★ **Table 6-12. Internal I/O Space 2**

Physical Address	Internal I/O
0x0BFF FFFF to 0x0B00 0300	RFU
0x0B00 02FF to 0x0B00 02E0	GIU2
0x0B00 02DF to 0x0B00 02C0	PMU2
0x0B00 02BF to 0x0B00 02A0	PIU2
0x0B00 029F to 0x0B00 0260	RFU
0x0B00 025F to 0x0B00 0240	LED
0x0B00 023F to 0x0B00 0220	RFU
0x0B00 021F to 0x0B00 0200	ICU2
0x0B00 01FF to 0x0B00 01E0	RFU
0x0B00 01DF to 0x0B00 01C0	RTC2
0x0B00 01BF to 0x0B00 01A0	DSIU
0x0B00 019F to 0x0B00 0180	KIU
0x0B00 017F to 0x0B00 0160	AIU
0x0B00 015F to 0x0B00 0140	RFU
0x0B00 013F to 0x0B00 0120	PIU1
0x0B00 011F to 0x0B00 0100	GIU1
0x0B00 00FF to 0x0B00 00E0	DSU
0x0B00 00DF to 0x0B00 00C0	RTC1
0x0B00 00BF to 0x0B00 00A0	PMU
0x0B00 009F to 0x0B00 0080	ICU1
0x0B00 007F to 0x0B00 0060	CMU
0x0B00 005F to 0x0B00 0040	DCU
0x0B00 003F to 0x0B00 0020	DMAA
0x0B00 001F to 0x0B00 0000	BCU

6.3.4 LCD space

This space is used to access the external LCD controller.

★ The data bus’s bit width is set via the DBUS32 pin and the BCUCNTREG3’s LCD32/ISA32 bit.

- DBUS32 pin = 0:16 bits
- DBUS32 pin = 1
  BCUCNTREG3’s LCD32 = 0:16 bits
  BCUCNTREG3’s LCD32 = 1:32 bits

The data accessed via this space can be bit-inverted/bit-reinverted by the GMODE bit of BCUCNTREG2.

The LCD space is used exclusively from the high-speed system bus memory space. To switch between these two types of space, set the ISAM/LCD bit in BCUCNTREG1.

6.3.5 DRAM space

The VR4121's DRAM space is divided into four banks. If the BCUCNTREG3's EXT_MEM bit has been set to enable use of expanded memory, the usable space is called expansion space. The following describes how banks are allocated according to the data bus width when using ordinary memory space and expansion space.

- During 16-bit bus mode: All banks are allocated in ordinary memory space.
- During 32-bit bus mode: Banks 0 and 1 are allocated in ordinary memory space and banks 2 and 3 are allocated in expansion space.

(1) Capacity of DRAM space

The DRAM address space varies in size depending on the data bus's bit width and the capacity of the DRAM being used.

- The data bus bit width is set via the DBUS32 pin.
- The DRAM capacity can be set in either of the following two ways.
  By setting the BCUCNTREG1 register's DRAM64 bit or the BCUCNTREG3 register's EXT_DRAM64 bit
  By setting the RAMSIZEREG register's SIZE bit

(a) Capacity setting via BCUCNTREG register

The capacity can be selected as 16 Mbits or 64 Mbits for either the ordinary memory space or the expansion space. Use the BCUCNTREG1 register's DRAM64 bit when selecting the ordinary memory space and use the BCUCNTREG3 register's EXT_DRAM64 bit when selecting the expansion space. However, the 16-Mbit setting cannot be selected for SDRAM.

(b) Capacity setting via RAMSIZEREG register

The capacity can be selected for each bank as 16, 64, or 128 Mbits. However, 16 Mbits cannot be selected for SDRAM and 128 Mbits cannot be selected for EDO-type DRAM.

(2) Mapping of physical address

The area for each bank is determined based on the data bus bit width and the bank capacity, as described below.

(a) During 16-bit bus mode (DBUS32 = 0)

Bank 3 (UUCAS#/MRAS3#): 0x0000 0000 + (Bank 3 capacity + Bank 2 capacity + Bank 1 capacity + Bank 0 capacity) to
0x0000 0000 + (Bank 3 capacity + Bank 1 capacity + Bank 0 capacity + 1 byte)
Bank 2 (ULCAS#/MRAS2#): 0x0000 0000 + (Bank 2 capacity + Bank 1 capacity + Bank 0 capacity) to 0x0000 0000 + (Bank 1 capacity + Bank 0 capacity + 1 byte)
Bank 1 (MRAS1#): 0x0000 0000 + (Bank 1 capacity + Bank 0 capacity) to
0x0000 0000 + (Bank 0 capacity + 1 byte)
Bank 0 (MRAS0#): 0x0000 0000 + (Bank 0 capacity) to 0x0000 0000

Physical addresses in the DRAM address space are indicated below for all memory spaces, when using 16-Mbit DRAM or 64-Mbit DRAM.
For details of ADD(25:0) pin connection, refer to **11.3 Connection of Address Pins**.

Table 6-13. DRAM Address Example (When Using 16-Bit Data Bus)

Physical Address	When Using 16-Mbit DRAM (DRAM64 = 0 or SIZE3 = SIZE2 = SIZE1 = SIZE0 = 0)	When Using 64-Mbit DRAM (DRAM64 = 1 or SIZE3 = SIZE2 = SIZE1 = SIZE0 = 2)
0x03FF FFFF to 0x0200 0000	RFU	RFU
0x01FF FFFF to 0x0180 0000		Bank 3 (UUCAS#/MRAS#3)
0x017F FFFF to 0x0100 0000		Bank 2 (ULCAS#/MRAS#2)
0x00FF FFFF to 0x0080 0000		Bank 1 (MRAS#1)
0x007F FFFF to 0x0060 0000	Bank 3 (UUCAS#/MRAS3#]	Bank 0 (MRAS#0)
0x005F FFFF to 0x0040 0000	Bank 2 (ULCAS#/MRAS2#)	
0x003F FFFF to 0x0020 0000	Bank 1 (MRAS1#)	
0x001F FFFF to 0x0000 0000	Bank 0 (MRAS0#)	

(b) During 32-bit bus mode (DBUS32 = 1)

Bank 3 (ROMCS3#): 0x0000 0000 + (Bank 3 capacity + Bank 2 capacity + Bank 1 capacity + Bank 0 capacity) to
★ 0x0000 0000 + (Bank 2 capacity + Bank 1 capacity + Bank 0 capacity + 1 byte)
Bank 2 (ROMCS2#): 0x0000 0000 + (Bank 2 capacity + Bank 1 capacity + Bank 0 capacity) to
0x0000 0000 + (Bank 1 capacity + Bank 0 capacity + 1 byte)
Bank 1 (MRAS1#): 0x0000 0000 + (Bank 1 capacity + Bank 0 capacity) to
0x0000 0000 + (Bank 0 capacity + 1 byte)
Bank 0 (MRAS0#): 0x0000 0000 + (Bank 0 capacity) to 0x0000 0000

Physical addresses in the RAM address space are indicated below for when 16-Mbit DRAM (banks 3 and 2), 16-Mbit DRAM (banks 1 and 0), or 64-Mbit DRAM is used.
For details of ADD(25:0) pin connection, refer to **11.3 Connection of Address Pins**.

Table 6-14. DRAM Address Example When Using 16-Mbit Expansion DRAM (When Using 32-Bit Data Bus)

Physical Address	When Using 16-Mbit DRAM (DRAM64 = 0, EXT_DRAM64 = 0, or SIZE3 = SIZE2 = SIZE1 = SIZE0 = 2)	When Using 64-Mbit DRAM (DRAM64 = 1, EXT_DRAM64 = 0, or SIZE1 = SIZE0 = 4, SIZE3 = SIZE2 = 2)
0x03FF FFFF to 0x2800 0000	RFU	RFU
0x027F FFFF to 0x0240 0000		Bank 3 (ROMCS3#)[Note]
0x023F FFFF to 0x0200 0000		Bank 2 (ROMCS2#)[Note]
0x01FF FFFF to 0x0180 0000		Bank 1 (MRAS1#)
0x017F FFFF to 0x0100 0000		
0x00FF FFFF to 0x00e0 0000	Bank 3 (ROMCS3#)	Bank 0 (MRAS0#)
0x00dF FFFF to 0x00c0 0000		
0x00bF FFFF to 0x00a0 0000	Bank 2 (ROMCS2#)	
0x009F FFFF to 0x0080 0000		
0x007F FFFF to 0x0060 0000	Bank 1 (MRAS1#)	
0x005F FFFF to 0x0040 0000		
0x003F FFFF to 0x0020 0000	Bank 0 (MRAS0#)	
0x001F FFFF to 0x0000 0000		

Note Can be used exclusively from the expansion ROM.

Physical addresses in the RAM address space are indicated below for when 64-Mbit DRAM (banks 3 and 2), 16-Mbit DRAM (banks 1 and 0), or 64-Mbit DRAM is used.

For details of ADD(25:0) pin connection, refer to **11.3 Connection of Address Pins**.

Table 6-15. DRAM Address Example When Using 64-Mbit Expansion DRAM (When Using 32-Bit Data Bus)

Physical Address	When Using 16-Mbit DRAM (DRAM64 = 0, EXT_DRAM64 = 1, or SIZE1 = SIZE0 = 2, SIZE3 = SIZE2 = 4)	When Using 64-Mbit DRAM (DRAM64 = 1, EXT_DRAM64 = 1, or SIZE3 = SIZE2 = SIZE1 = SIZE0 = 4)
0x03FF FFFF to 0x0300 0000	RFU	Bank 3 (ROMCS3#)[Note]
0x02FF FFFF to 0x0280 0000		Bank 2 (ROMCS2#)[Note]
0x027F FFFF to 0x0200 0000	Bank 3 (ROMCS3#)	
0x01FF FFFF to 0x0180 0000		Bank 1 (MRAS1#)
0x017F FFFF to 0x0100 0000	Bank 2 (ROMCS2#)	
0x00FF FFFF to 0x0080 0000		Bank 0 (MRAS0#)
0x007F FFFF to 0x0060 0000	Bank 1 (MRAS1#)	
0x005F FFFF to 0x0040 0000		
0x003F FFFF to 0x0020 0000	Bank 0 (MRAS0#)	
0x001F FFFF to 0x0000 0000		

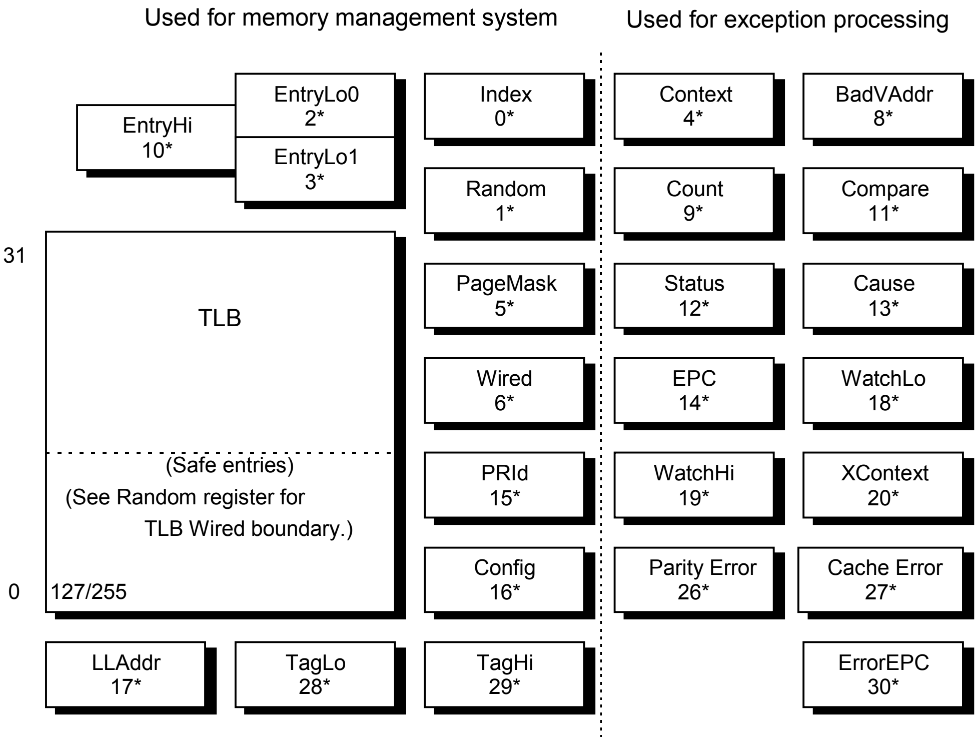
Note Can be used exclusively from the expansion ROM.

6.4 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 6-9. 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. See **CHAPTER 1 INTRODUCTION** for details. Also see **CHAPTER 7 EXCEPTION PROCESSING** for the CP0 functions and the relationships between exception processing and registers.

Figure 6-9. CP0 Registers and TLB



Remark *: Register number

Caution **When accessing the CP0 register, some instructions require consideration of the interval time until the next instruction is executed, because there is a delay from when the contents of the CP0 register change to when this change is reflected on the CPU operation. This time lag is called CP0 hazard. For details, see CHAPTER 30 VR4121 COPROCESSOR 0 HAZARDS.**

6.4.1 Format of a TLB entry

Figure 6-10 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 6-10. 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 which bit of the EntryHi register corresponds to the TLB G bit.

6.5 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.

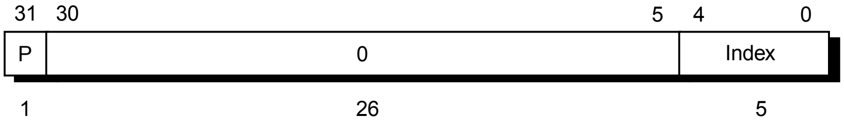
6.5.1 Index register (0)

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.

★ As the contents of the index register are set to undefined after reset, initialization by software is necessary.

Figure 6-11. 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.

6.5.2 Random register (1)

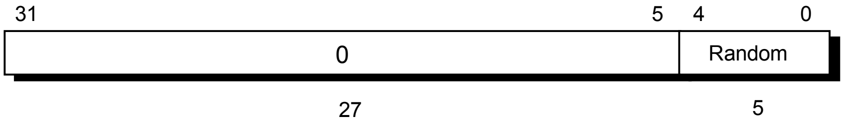
The Random register is a read-only register. The low-order 5 bits are used in referencing a TLB entry. This register is decremented each time an instruction is executed. The values that can be set in the register are as follows:

- 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 6-12 shows the format of the Random register.

Figure 6-12. Random Register



Random: TLB random index

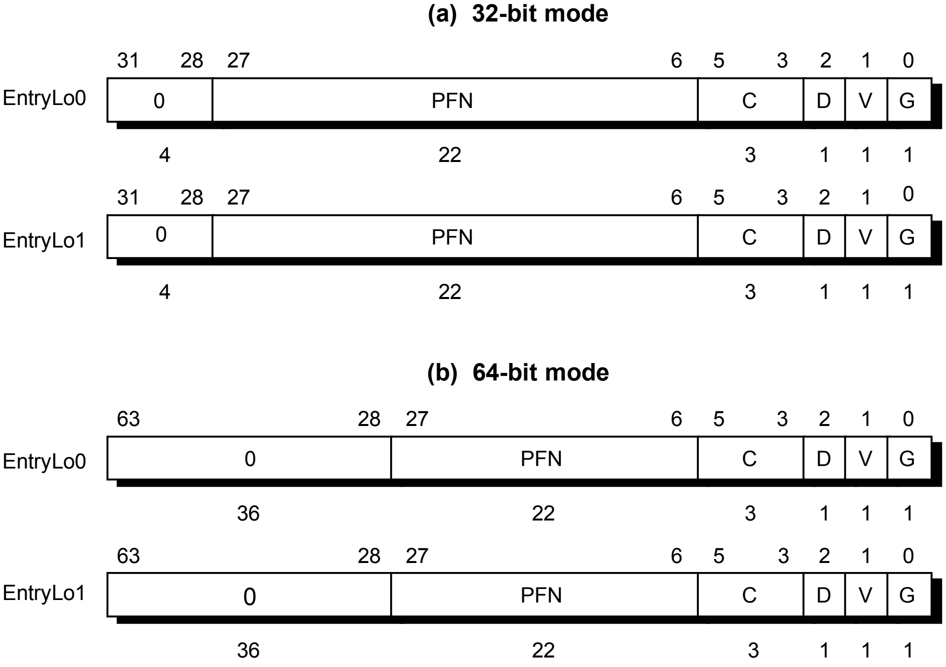
0: RFU. Write 0 in a write operation. When this field is read, 0 is read.

6.5.3 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.

★ As the contents of the index register are set to undefined after reset, initialization by software is necessary.

Figure 6-13. 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 6-16).

D: Dirty. If this bit is set to 1, the page is marked as dirty and, therefore, writable. 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 6-16 lists the page attributes selected according to the value in the C bits.

Table 6-16. Cache Algorithm

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

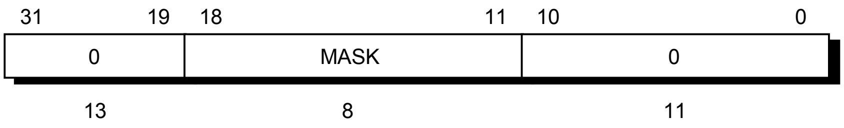
6.5.4 PageMask register (5)

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 6-17. 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.

★ As the contents of the index register are set to undefined after reset, initialization by software is necessary.

Figure 6-14. PageMask 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 6-17 lists the mask pattern for each page size. If the mask pattern is one not listed below, the TLB behaves unexpectedly.

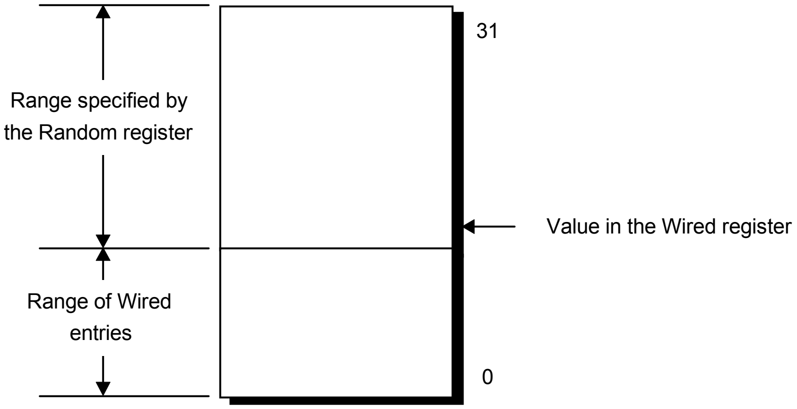
Table 6-17. Mask Values and Page Sizes

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

6.5.5 Wired register (6)

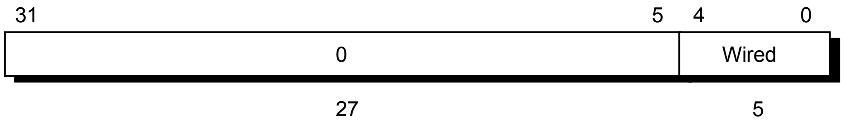
The Wired register is a read/write register that specifies the lower boundary of the random entry of the TLB as shown in Figure 6-15. Wired entries cannot be overwritten by a TLBWR instruction. They can, however, be overwritten by a TLBWI instruction. Random entries can be overwritten by both instructions.

Figure 6-15. 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 **6.5.2 Random register (1)**). Figure 6-16 shows the format of the Wired register.

Figure 6-16. Wired Register



Wired: TLB wired boundary

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

6.5.6 EntryHi register (10)

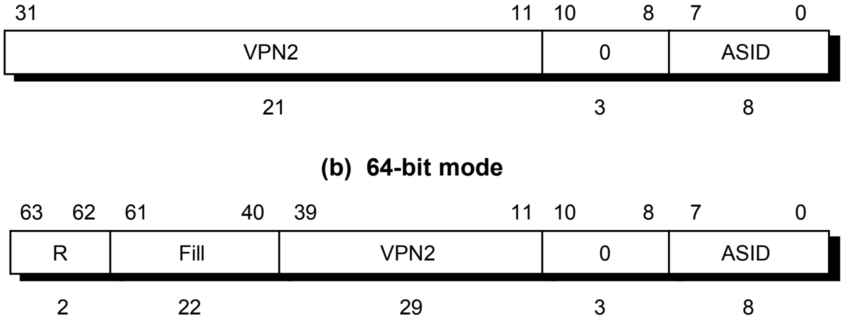
The EntryHi register is write-accessible. It is used to access the on-chip TLB. The EntryHi register holds the high-order bits of a TLB entry for TLB read and write operations. If a TLB Mismatch, TLB Invalid, or TLB Modified exception occurs, the EntryHi register holds the high-order bit of the TLB entry. The EntryHi register is also set with the virtual page number (VPN2) for a virtual address where an exception occurred and the ASID. See **CHAPTER 7 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.

★ As the contents of the index register are set to undefined after reset, initialization by software is necessary.

Figure 6-17. EntryHi Register



(a) 32-bit mode

(b) 64-bit mode

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.

6.5.7 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 6-18 shows the format of the PRId register.

Figure 6-18. PRId Register

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

Imp: CPU core processor ID number (0x0C for the VR4121)

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.

6.5.8 Config register (16)

The Config register specifies various configuration options selected on VR4121 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 VR4121, 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 6-19 shows the format of the Config register.

★ As the contents of the index register are set to undefined after reset, initialization by software is necessary.

Figure 6-19. 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 3	2 0
0	EC	EP	AD	0	M16	0	1	0	BE	10	CS	IC	DC	0	K0
1	3	4	1	2	1	2	1	1	1	2	1	3	3	3	3

EC: System interface clock (TClock) frequency ratio (read only)
0 → Processor clock frequency divided by 1.5
1 → Processor clock frequency divided by 2
2 → Processor clock frequency divided by 2.5
3 → Processor clock frequency divided by 3
4 → Processor clock frequency divided by 4
5 → Processor clock frequency divided by 5
6 → Processor clock frequency divided by 6
7 → Processor clock frequency divided by 1

EP: Transfer data pattern (cache write-back pattern) setting
0 → DD: 1 word/1 cycle
Others → RFU

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
1 → MIPS16 instruction can be executed.

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

CS: Cache size mode indication (fixed to 1 in the VR4121)
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 VR4121, $2^{(IC+10)}$ bytes.
4 → 16 Kbytes
Others → RFU

DC: Data cache size indication. In the VR4121, $2^{(DC+10)}$ bytes.
3 → 8 Kbytes
Others → RFU

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 and the AD bit to 0. If they are set with any other values, the processor may behave unexpectedly.

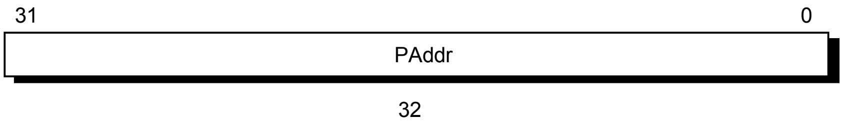
6.5.9 Load linked address (LLAddr) register (17)

The read/write Load Linked Address (LLAddr) register is not used with the VR4121 processor except for diagnostic purpose, and serves no function during normal operation.

LLAddr register is implemented just for compatibility between the VR4121 and VR4000/VR4400.

★ The contents in the LLAddr register are undefined after reset.

Figure 6-20. LLAddr Register



PAddr: 32-bit physical address

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

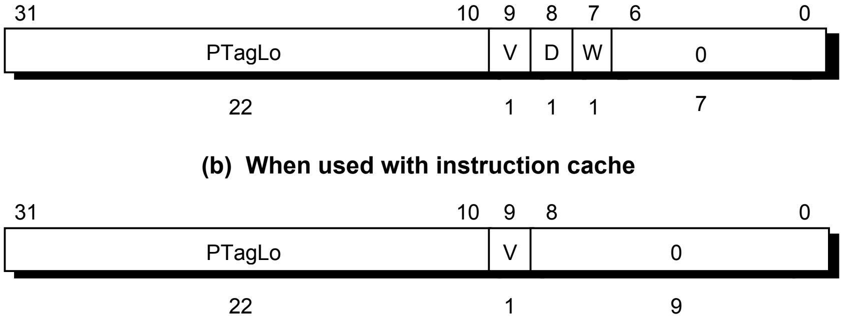
The TagLo and TagHi registers are 32-bit read/write registers that hold the primary cache tag during cache initialization, cache diagnostics, or cache error processing. The Tag registers are written by the CACHE and MTC0 instructions.

Figures 6-21 and 6-22 show the format of these registers.

★ The contents of these registers are undefined after reset.

Figure 6-21. TagLo Register

(a) When used with data cache



(b) When used with instruction cache

PTagLo: Specifies physical address bits 31 to 10.

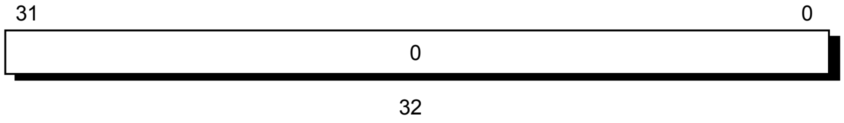
V: Valid bit

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 6-22. TagHi Register



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

6.5.11 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] 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] of the 64-bit virtual address are compared to the contents of the VPN2 (virtual page number divided by two) of each TLB entry.

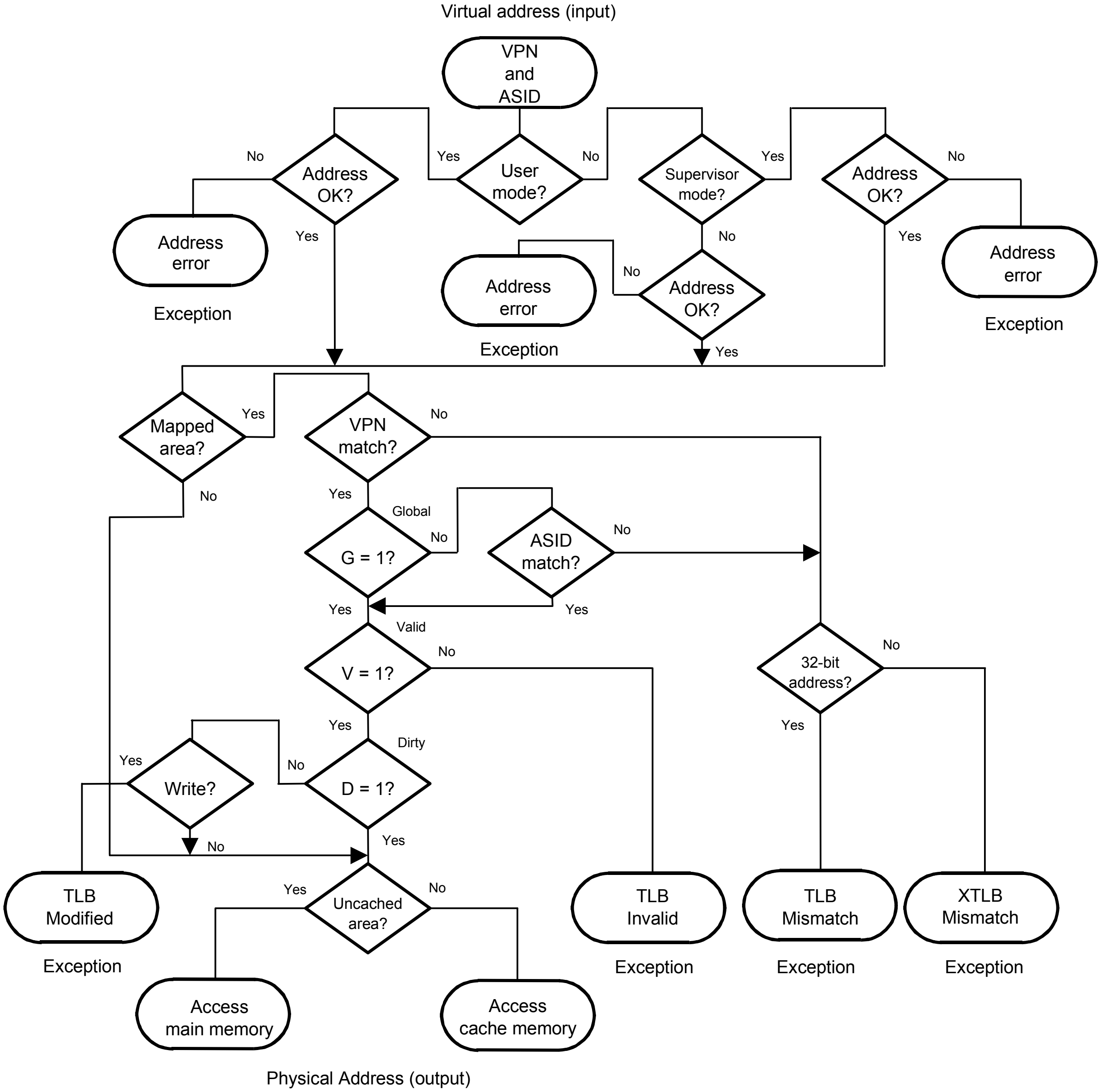
If a TLB entry matches, the physical address and access control bits (C, D, and V) are retrieved from the matching TLB entry. While the V bit of the entry must be set to 1 for a valid address translation to take place, it is not involved in the determination of a matching TLB entry.

Figure 6-23 illustrates the TLB address translation flow.

Note The number of bits differs from page sizes. The table below shows the examples of high-order bits of the virtual address in page size of 256 Kbytes and 1 Kbyte.

Page Size Mode	256 Kbytes	1 Kbyte
32-bit mode	bits 31 to 19	bits 31 to 11
64-bit mode	bits 63, 62, 39 to 19	bits 63, 62, 39 to 11

Figure 6-23. TLB Address Translation



6.5.12 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 **CHAPTER 7 EXCEPTION PROCESSING** for details of the TLB Miss exception.

6.5.13 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.

(2) 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.

(3) 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.

(4) 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.

CHAPTER 7 EXCEPTION PROCESSING

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

7.1 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 **CHAPTER 6 MEMORY MANAGEMENT SYSTEM** for a description of system operating modes). If an exception occurs while executing a MIPS16 instruction, the processor stops the MIPS16 instruction execution, and shifts to the 32-bit instruction execution mode.

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. Note that no branch delay slot generated by executing a branch instruction exists when the processor operates in the MIPS16 mode.

When MIPS16 instructions are enabled to be executed, bit 0 of the EPC register indicates the operating mode in which an exception occurred. It indicates 1 when in the MIPS16 instruction mode, and indicates 0 when in the MIPS III instruction mode.

The VR4121 processor supports a Supervisor mode and high-speed TLB refill for all address spaces. The VR4121 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:

(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) 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.

(3) Exception/error levels

Returning from an exception resets the exception level to normal (0) (for details, see **CHAPTER 28 MIPS III INSTRUCTION SET DETAILS**).

The registers that retain address, cause, and status information during exception processing are described in **7.3 Exception Processing Registers**. For a description of the exception process, see **7.4 Details of Exceptions**.

7.2 Precision of Exceptions

VR4121 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.

7.3 Exception Processing Registers

This section describes the CP0 registers that are used in exception processing. Table 7-1 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 **CHAPTER 6 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 7-1 are used in exception processing, and are described in the sections that follow.

Table 7-1. 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

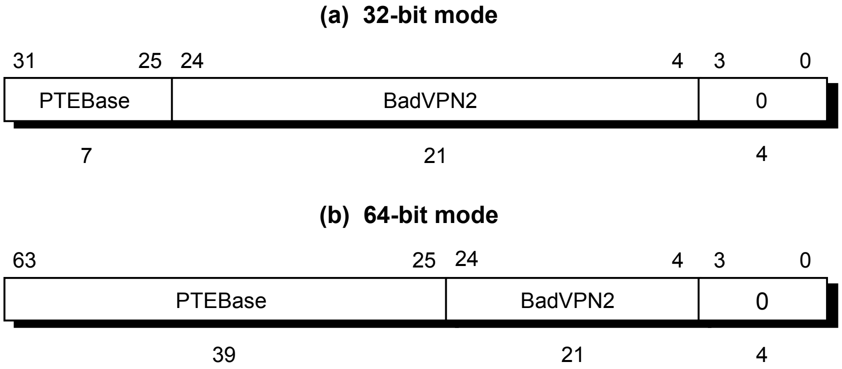
Caution This register is prepared to maintain compatibility with the VR4100. This register is not used in the VR4121 hardware.

7.3.1 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 7-1 shows the format of the Context register.

Figure 7-1. 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.

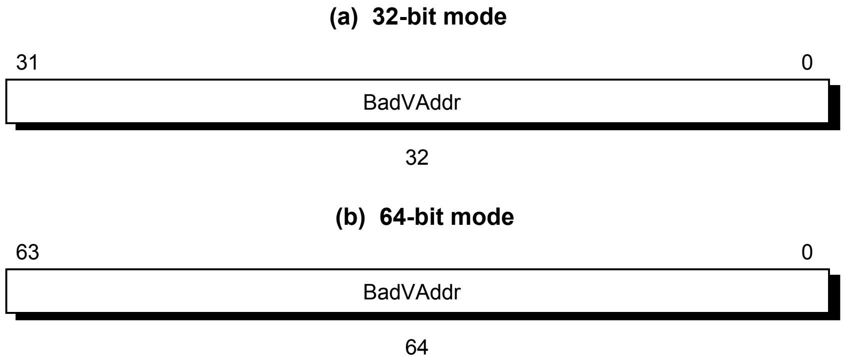
The 21-bit BadVPN2 field contains bits 31 to 11 of the virtual address that caused the TLB miss; bit 10 is excluded because a single TLB entry maps to an even-odd page pair. For a 1-Kbyte page size, this format can directly address the pair-table of 8-byte PTEs. When the page size is 4 Kbytes or more,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

7.3.2 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 7-2 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 7-2. 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.

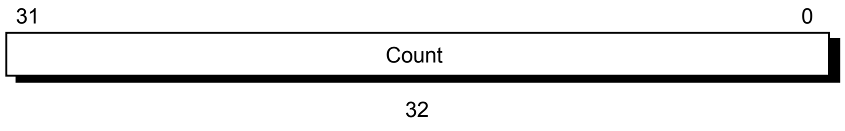
7.3.3 Count register (9)

The read/write Count register acts as a timer. It is incremented in synchronization with the frequencies of MasterOut clock (refer to **1.8 Clock Interface**), 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 7-3 shows the format of the Count register.

Figure 7-3. Count Register Format



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

7.3.4 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 **7.3.3 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 7-4 shows the format of the Compare register.

★ As the contents of the index register are set to undefined after reset.

Figure 7-4. Compare Register Format

31 ... 0
Compare
32

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

7.3.5 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 7-5 shows the format of the Status register.

Figure 7-5. Status Register Format

31 ... 29	28	27 26	25	24 ... 16	15 ... 8	7	6	5	4 3	2	1	0
0	CU0	0	RE	DS	IM	KX	SX	UX	KSU	ERL	EXL	IE
3	1	2	1	9	8	1	1	1	2	1	1	1

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 since the VR4121 supports the little-endian order only.

DS: Diagnostic Status field (see **Figure 7-6**).

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 VR4121.
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, refer to **CHAPTER 15 ICU (INTERRUPT CONTROL UNIT)**.

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 7-6 shows the details of the Diagnostic Status (DS) field. All DS field bits other than the TS bit are writable.

Figure 7-6. 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 VR4121 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.

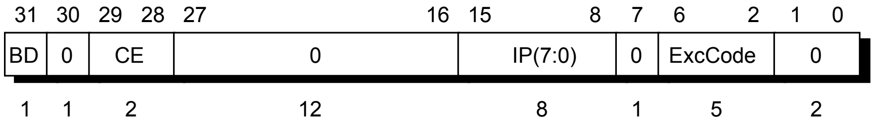
- TS and SR are cleared to 0.
- ERL and BEV are set to 1.
- SR is 0 after Cold reset, and is 1 after Soft reset or NMI interrupt.

Remark Cold reset and Soft reset are CPU core reset (see **8.3 Reset of the CPU Core**). For the reset of all the VR4121 including peripheral units, refer to **CHAPTER 8 INITIALIZATION INTERFACE** and **CHAPTER 16 PMU (POWER MANAGEMENT UNIT)**.

7.3.6 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 7-2**). 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 7-7 shows the fields of this register; Table 7-2 describes the Cause register codes.

Figure 7-7. Cause Register Format



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 VR4121.

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, refer to **CHAPTER 15 ICU (INTERRUPT CONTROL UNIT)**.

ExcCode: Exception code field (refer to **Table 7-2** for details).

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

Table 7-2. 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 VR4121 has eight interrupt request sources, IP7 to IP0. For the detailed description of interrupts, refer to **CHAPTER 10 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.

7.3.7 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. The contents of this register change depending on whether execution of MIPS16 instructions is enabled or disabled. Setting the MIPS16EN pin after RTC reset specifies whether execution of the MIPS16 instructions is enabled or disabled.

When the MIPS16 instruction execution is disabled, 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).

When the MIPS16 instruction execution is enabled, the EPC register contains either:

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

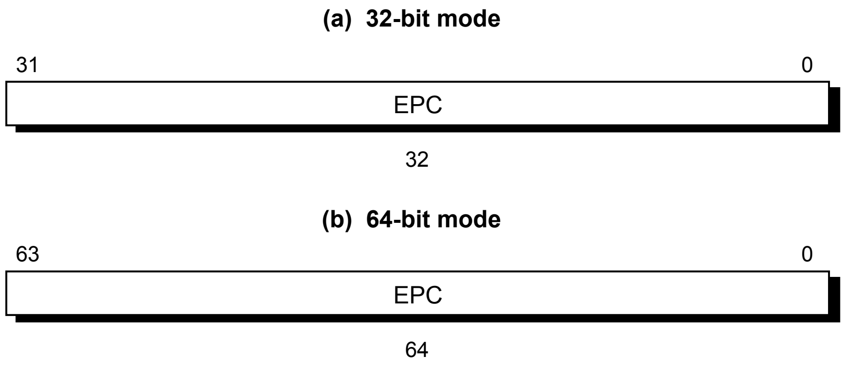
When the 16-bit instruction is executed, the EPC register contains either:

- Virtual address of the instruction that caused the exception and ISA mode at which an exception occurs.
- Virtual address of the immediately preceding Extend or jump instruction and ISA mode at which an exception occurs (when the instruction associated with the exception is in a branch delay slot of the jump instruction or in the instruction following the Extend instruction, 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 7-8 shows the EPC register format when MIPS16 ISA is disabled, and Figure 7-9 shows the EPC register format when MIPS16 ISA is enabled.

Figure 7-8. EPC Register Format (When MIPS16 ISA Is Disabled)



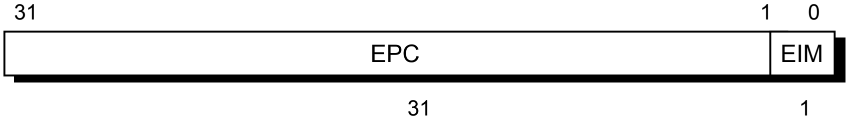
(a) 32-bit mode

(b) 64-bit mode

EPC: Restart address after exception processing.

Figure 7-9. EPC Register Format (When MIPS16 ISA Is Enabled)

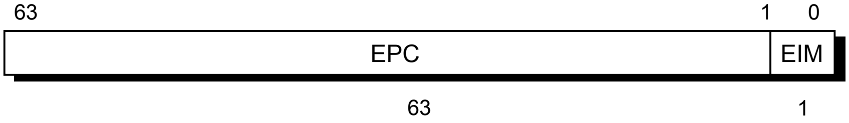
(a) 32-bit mode



EPC: Virtual address that caused the exception (31:1).

EIM: ISA mode at which an exception occurs.
(1: when MIPS16 SIA instruction is executed, 0: when MIPS III ISA instruction is executed.)

(b) 64-bit mode



EPC: Virtual address that caused the exception (63:1).

EIM: ISA mode at which an exception occurs.
(1: when MIPS16 ISA instruction is executed, 0: when MIPS III ISA instruction is executed.)

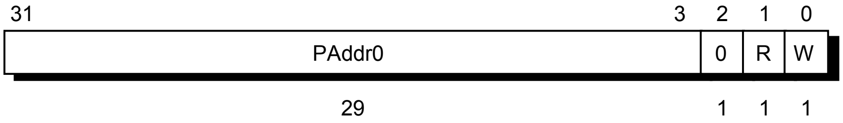
7.3.8 WatchLo (18) and WatchHi (19) registers

The VR4121 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 7-10 and 7-11 show the format of the WatchLo and WatchHi registers.

★ As the contents of the index register are set to undefined after reset, initialization by software is necessary.

Figure 7-10. WatchLo Register Format



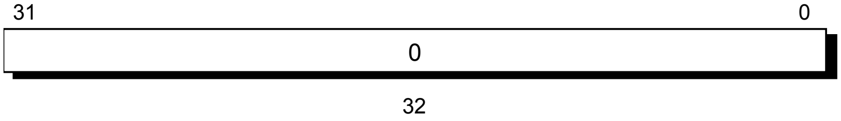
PAddr0: Specifies physical address bits 31 to 3.

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 7-11. WatchHi Register Format



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

7.3.9 XContext register (20)

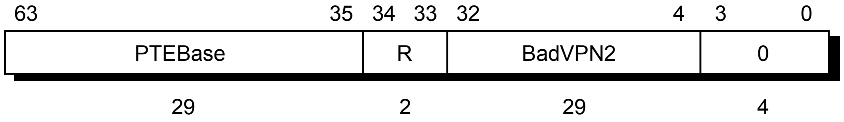
The read/write XContext register contains a pointer to an entry in the page table entry (PTE) array, an operating system data structure that stores virtual-to-physical address translations. If a TLB miss occurs, the operating system loads the untranslated data from the PTE into the TLB to handle the 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 7-12 shows the format of the XContext register.

Figure 7-12. 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.

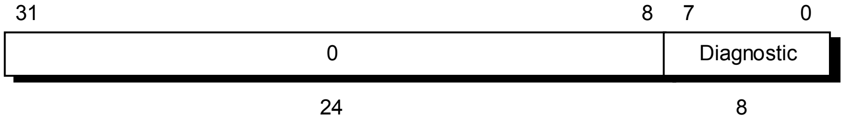
The 29-bit BadVPN2 field has bits 39 to 11 of the virtual address that caused the TLB miss; bit 10 is excluded because a single TLB entry maps to an even-odd page pair. For a 1-Kbyte page size, this format may be used directly to address the pair-table of 8-byte PTEs. For 4-Kbyte or more page and PTE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

7.3.10 Parity error register (26)

The Parity Error (PErr) register is a readable/writable register. This register is defined to maintain software-compatibility with the VR4100, and is not used in hardware because the VR4121 has no parity.

Figure 7-13 shows the format of the PErr register.

Figure 7-13. Parity Error Register Format



Diagnostic: 8-bit self diagnostic field.

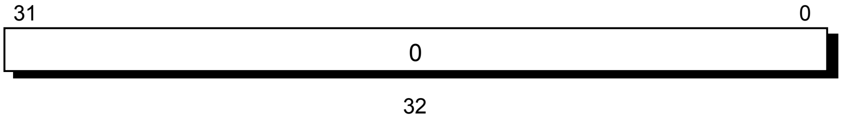
0: RFU. Write 0 in a write operation. When this field is read, 0 is read.

7.3.11 Cache error register (27)

The Cache Error register is a readable/writable register. This register is defined to maintain software-compatibility with the VR4100, and is not used in hardware because the VR4121 has no parity.

Figure 7-14 shows the format of the Cache Error register.

Figure 7-14. Cache Error Register Format



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

7.3.12 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. The contents of this register change depending on whether execution of MIPS16 instructions is enabled or disabled. Setting the MIPS16EN pin after RTC reset specifies whether the execution of MIPS16 instructions is enabled or disabled.

When the MIPS16 instruction execution is disabled, 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.

When the MIPS16 instruction execution is enabled during a 32-bit instruction execution, this address can be:

- Virtual address of the instruction that caused the exception and ISA mode at which an exception occurs.
- Virtual address of the immediately preceding branch or jump instruction and ISA mode at which an exception occurs when the instruction associated with the exception is in a branch delay slot.

When the MIPS16 instruction execution is enabled during a 16-bit instruction execution, this address can be:

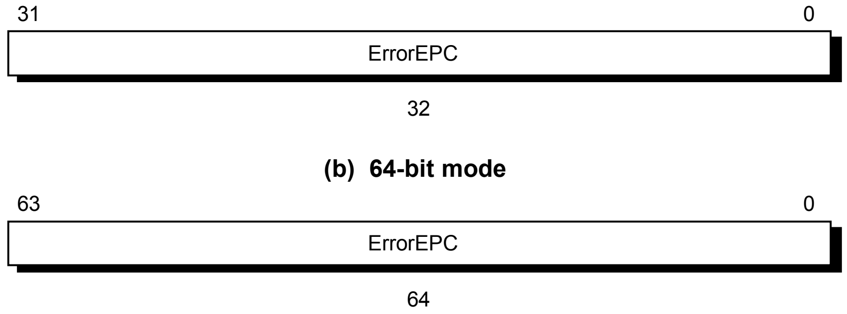
- Virtual address of the instruction that caused the exception and ISA mode at which an exception occurs.
- Virtual address of the immediately preceding jump instruction or Extend instruction and ISA mode at which an exception occurs when the instruction associated with the exception is in a branch delay slot of the jump instruction or is the instruction following the Extend instruction.

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 7-15 shows the format of the ErrorEPC register when the MIPS16 ISA is disabled. Figure 7-16 shows the format of the ErrorEPC register when the MIPS16 ISA is enabled.

Figure 7-15. ErrorEPC Register Format (When MIPS16 ISA Is Disabled)



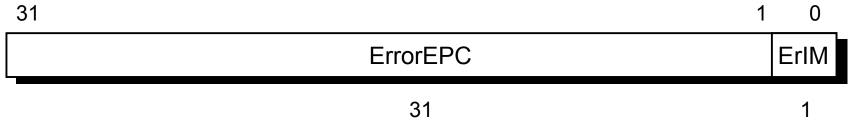
(a) 32-bit mode

(b) 64-bit mode

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

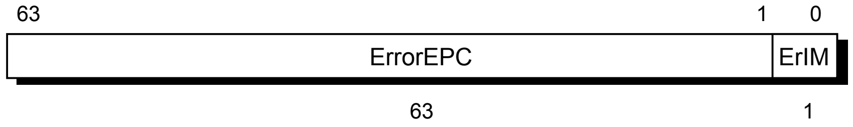
Figure 7-16. ErrorEPC Register Format (When MIPS16 ISA Is Enabled)

(a) 32-bit mode



ErrorEPC: Virtual restart address (31:1) after Cold reset, Soft reset, or NMI exception.
ErIM: ISA mode at which an error exception occurs (1: MIPS16 ISA, 0: MIPS III ISA).

(b) 64-bit mode



ErrorEPC: Virtual restart address (63:1) after Cold reset, Soft reset, or NMI exception.
ErIM: ISA mode at which an error exception occurs (1: MIPS16 ISA, 0: MIPS III ISA).

7.4 Details of Exceptions

This section describes causes, processes, and services of the VR4121's exceptions.

7.4.1 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 **CHAPTER 28 MIPS III INSTRUCTION SET DETAILS**.

7.4.2 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.

- 0xBFC0 0000 in 32-bit mode (virtual address)
- 0xFFFF FFFF BFC0 0000 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 7-3. 64-Bit Mode Exception Vector Base Addresses

	Vector Base Address (Virtual)	Vector Offset
Cold Reset Soft Reset NMI	0xFFFF FFFF BFC0 0000 (BEV bit is automatically set to 1)	0x0000
TLB Refill (EXL = 0)	0xFFFF FFFF 8000 0000 (BEV = 0) 0xFFFF FFFF BFC0 0200 (BEV = 1)	0x0000
XTLB Refill (EXL = 0)		0x0080
Other exceptions		0x0180

Table 7-4. 32-Bit Mode Exception Vector Base Addresses

	Vector Base Address (Virtual)	Vector Offset
Cold Reset Soft Reset NMI	0xBFC0 0000 (BEV bit is automatically set to 1)	0x0000
TLB Refill (EXL = 0)	0x8000 0000 (BEV = 0) 0xBFC0 0200 (BEV = 1)	0x0000
XTLB Refill (EXL = 0)		0x0080
Other exceptions		0x0180

(1) TLB Refill exception vector

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

- 0x8000 0000 in 32-bit mode
- 0xFFFF FFFF 8000 0000 in 64-bit mode

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

- 0xBFC0 0200 in 32-bit mode
- 0xFFFF FFFF BFC0 0200 in 64-bit mode

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

7.4.3 Priority of exceptions

While more than one exception can occur for a single instruction, only the exception with the highest priority is reported. Table 7-5 lists the priorities.

★ **Table 7-5. Exception Priority Order**

High	Cold Reset
↑	Soft Reset
|	NMI
|	Address Error (instruction fetch)
|	TLB/XTLB Refill (instruction fetch)
|	TLB Invalid (instruction fetch)
|	Interrupt (other than NMI)
|	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
Low	Bus Error (data access)

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

7.4.4 Cold reset exception

(1) Cause

The Cold Reset exception occurs when the ColdReset# signal (internal) is asserted and then deasserted. This exception is not maskable. The Reset# signal (internal) must be asserted along with the ColdReset# signal (for details, see **CHAPTER 8 INITIALIZATION INTERFACE**).

(2) Processing

The CPU provides a special interrupt vector for this exception:

- 0xBFC0 0000 (virtual address) in 32-bit mode
- 0xFFFF FFFF BFC0 0000 (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:

- When the MIPS16 instruction execution is disabled while the ERL of Status register is 0, the PC value at which an exception occurs is set to the ErrorEPC register.
  When the MIPS16 instruction execution is enabled while the ERL of Status register is 0, the PC value at which an exception occurs is set to the ErrorEPC register and the ISA mode in which an exception occurs is set to the least significant bit of 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.

(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

7.4.5 Soft reset exception

(1) Cause

A Soft Reset (sometimes called Warm Reset) occurs when the ColdReset# signal (internal) remains deasserted while the Reset# signal (internal) goes from assertion to deassertion (for details, see **CHAPTER 8 INTIALIZATION 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 VR4121, a soft reset never occurs.

(2) Processing

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

- 0xBFC0 0000 (virtual) in 32-bit mode
- 0xFFFF FFFF BFC0 0000 (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:

- When the MIPS16 instruction execution is disabled, the PC value at which an exception occurs is set to the ErrorEPC register.
  When the MIPS16 instruction execution is enabled, the PC value at which an exception occurs is set to the ErrorEPC register and the ISA mode in which an exception occurs is set to the least significant bit of 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

7.4.6 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 **CHAPTER 10 CPU CORE INTERRUPTS** and **CHAPTER 15 ICU (INTERRUPT CONTROL UNIT)**).

(2) Processing

The CPU provides a special interrupt vector for this exception:

- 0xBFC0 0000 (virtual) in 32-bit mode
- 0xFFFF FFFF BFC0 0000 (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:

- When the MIPS16 instruction execution is disabled, the PC value at which an exception occurs is set to the ErrorEPC register.
  When the MIPS16 instruction execution is enabled, the PC value at which an exception occurs is set to the ErrorEPC register and the ISA mode in which an exception occurs is set to the least significant bit of 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

7.4.7 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 when the MIPS16 instruction is disabled
- Branching to address whose least-significant 2 bits are 10 when the MIPS16 instruction is enabled

(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.

When the MIPS16 instruction is disabled, 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.

When the MIPS16 instruction is enabled, the EPC register contains the address of the instruction that caused the exception, and the least significant bit stores the ISA mode in which an exception occurs. However, if this instruction is in a branch delay slot or is the instruction following the Extend instruction, the EPC register contains the address of the preceding jump or Extend instruction, and the BD bit of the Cause register is set to 1.

(3) Servicing

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

7.4.8 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.

When the MIPS16 instruction is disabled, 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.

When the MIPS16 instruction is enabled, the EPC register contains the address of the instruction that caused the exception, and the least significant bit stores the ISA mode in which an exception occurs. However, if this instruction is in a branch delay slot or is the instruction following the Extend instruction, the EPC register contains the address of the preceding jump or Extend 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.

When the MIPS16 instruction is disabled, 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.

When the MIPS16 instruction is enabled, the EPC register contains the address of the instruction that caused the exception, and the least significant bit stores the ISA mode in which an exception occurs. However, if this instruction is in a branch delay slot or is the instruction following the Extend instruction, the EPC register contains the address of the preceding jump or Extend 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 writable (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.

When the MIPS16 instruction is disabled, 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.

When the MIPS16 instruction is enabled, the EPC register contains the address of the instruction that caused the exception, and the least significant bit stores the ISA mode in which an exception occurs. However, if this instruction is in a branch delay slot or is the instruction following the Extend instruction, the EPC register contains the address of the preceding jump or Extend 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 (/writable) 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.

7.4.9 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, refer to **11.4.7 Illegal access notification**.

(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.

When the MIPS16 instruction is disabled, 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.

When the MIPS16 instruction is enabled, the EPC register contains the address of the instruction that caused the exception, and the least significant bit stores the ISA mode in which an exception occurs. However, if this instruction is in a branch delay slot or is the instruction following the Extend instruction, the EPC register contains the address of the preceding jump or Extend 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.

7.4.10 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.

7.4.11 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.

When the MIPS16 instruction is disabled, 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.

When the MIPS16 instruction is enabled, the EPC register contains the address of the instruction that caused the exception, and the least significant bit stores the ISA mode in which an exception occurs. However, if this instruction is in a branch delay slot or is the instruction following the Extend instruction, the EPC register contains the address of the preceding jump or Extend instruction, and the BD bit of the Cause register is set to 1; otherwise this bit is cleared.

(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.

When a Breakpoint exception occurs while executing the MIPS16 instruction, a value of 2 should be added 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.

7.4.12 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.
When the MIPS16 instruction is disabled, 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.
When the MIPS16 instruction is enabled, the EPC register contains the address of the instruction that caused the exception, and the least significant bit stores the ISA mode in which an exception occurs. However, if this instruction is in a branch delay slot or is the instruction following the Extend instruction, the EPC register contains the address of the preceding jump or Extend 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:

(a) If the process is entitled access to the coprocessor, the coprocessor is marked usable and the corresponding state is restored to the coprocessor.
(b) 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.
(c) 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.
(d) 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.

7.4.13 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) when executing the MIPS16 instruction
- I8 instruction with an undefined minor op code (bits 10 to 8) when executing the MIPS16 instruction

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.

When the MIPS16 instruction is disabled, 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.

When the MIPS16 instruction is enabled, the EPC register contains the address of the instruction that caused the exception, and the least significant bit stores the ISA mode in which an exception occurs. However, if this instruction is in a branch delay slot or is the instruction following the Extend instruction, the EPC register contains the address of the preceding jump or Extend 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.

7.4.14 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.

7.4.15 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.

7.4.16 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.
When the MIPS16 instruction is disabled, 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.
When the MIPS16 instruction is enabled, the EPC register contains the address of the instruction that caused the exception, and the least significant bit stores the ISA mode in which an exception occurs. However, if this instruction is in a branch delay slot or is the instruction following the Extend instruction, the EPC register contains the address of the preceding jump or Extend 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.

7.4.17 Interrupt exception

(1) Cause

The Interrupt exception occurs when one of the eight interrupt conditions[Note] is asserted. In the VR4121, 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 VR4121.

(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.

When the MIPS16 instruction is disabled, 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.

When the MIPS16 instruction is enabled, the EPC register contains the address of the instruction that caused the exception, and the least significant bit stores the ISA mode in which an exception occurs. However, if this instruction is in a branch delay slot or is the instruction following the Extend instruction, the EPC register contains the address of the preceding jump or Extend 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.

7.5 Exception Processing and Servicing Flowcharts

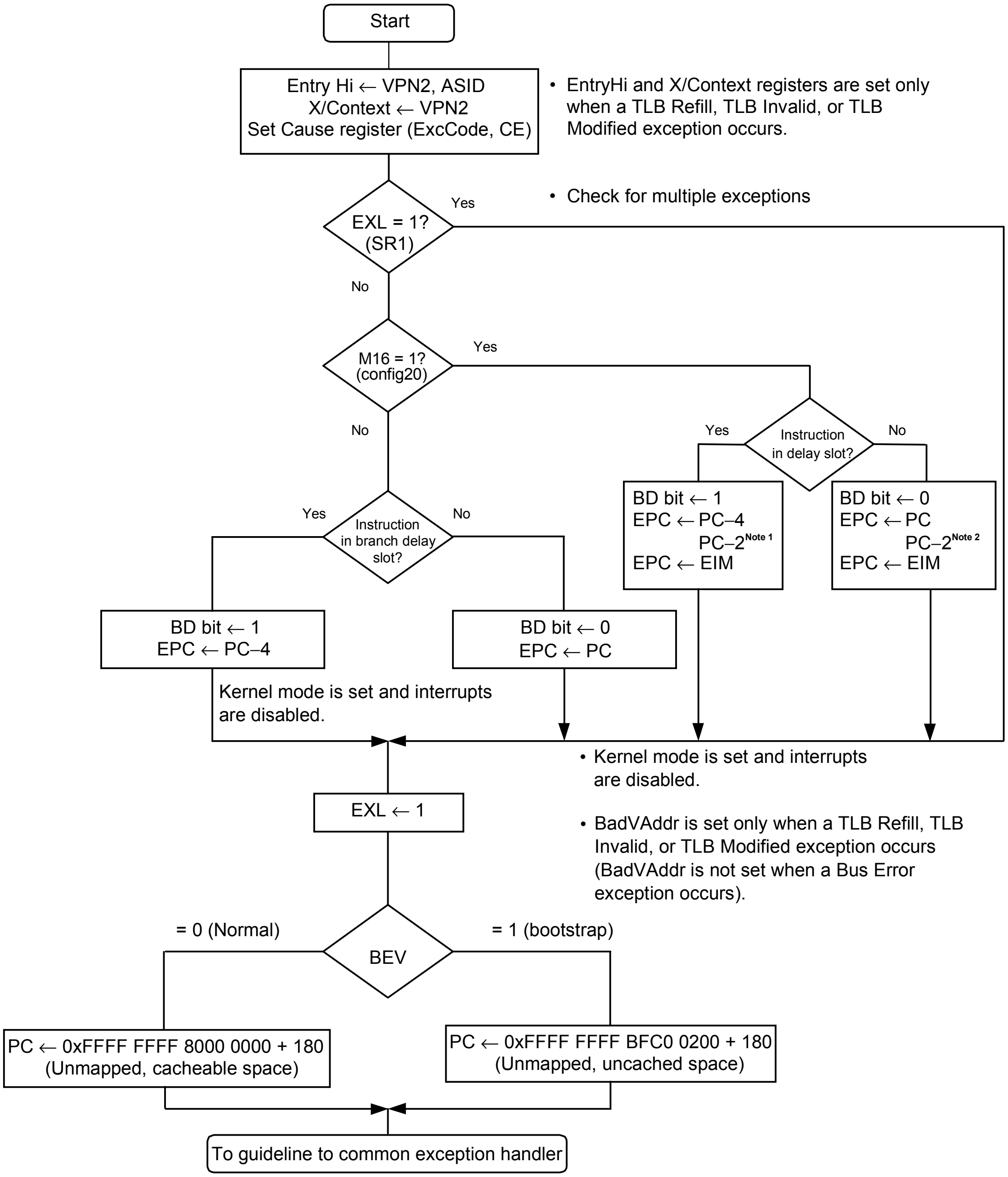
The remainder of this chapter contains flowcharts for the following exceptions and guidelines for their handlers:

- 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 7-17. 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
PC–2Note 1
EPC ← EIM
BD bit ← 0
EPC ← PC
PC–2Note 2
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 ← 0xFFFF FFFF 8000 0000 + 180
(Unmapped, cacheable space)
PC ← 0xFFFF FFFF BFC0 0200 + 180
(Unmapped, uncached space)
To guideline to common exception handler

Notes 1. When the JR or JALR instruction of MIPS16 instructions

2. When the Extend instruction of MIPS16 instructions

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 7-17. Common Exception Handling (2/2)

(b) Servicing common exceptions (software)

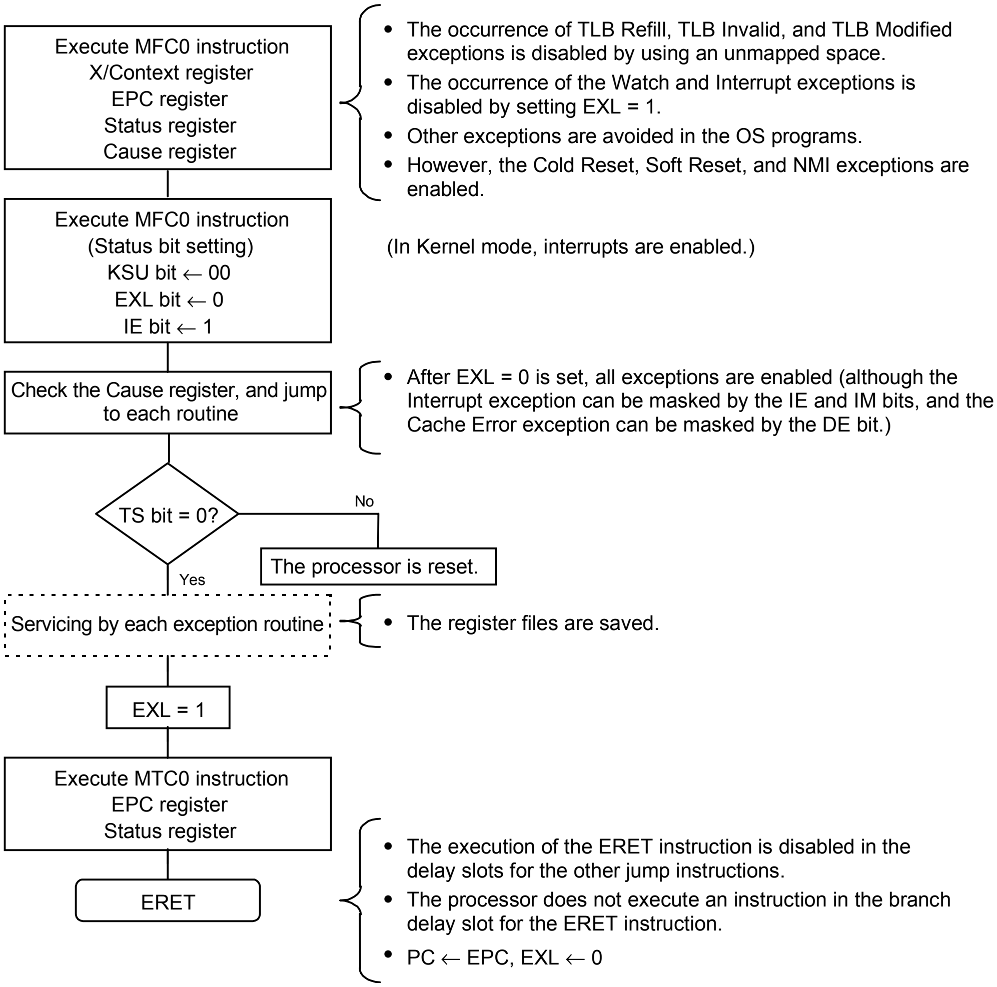
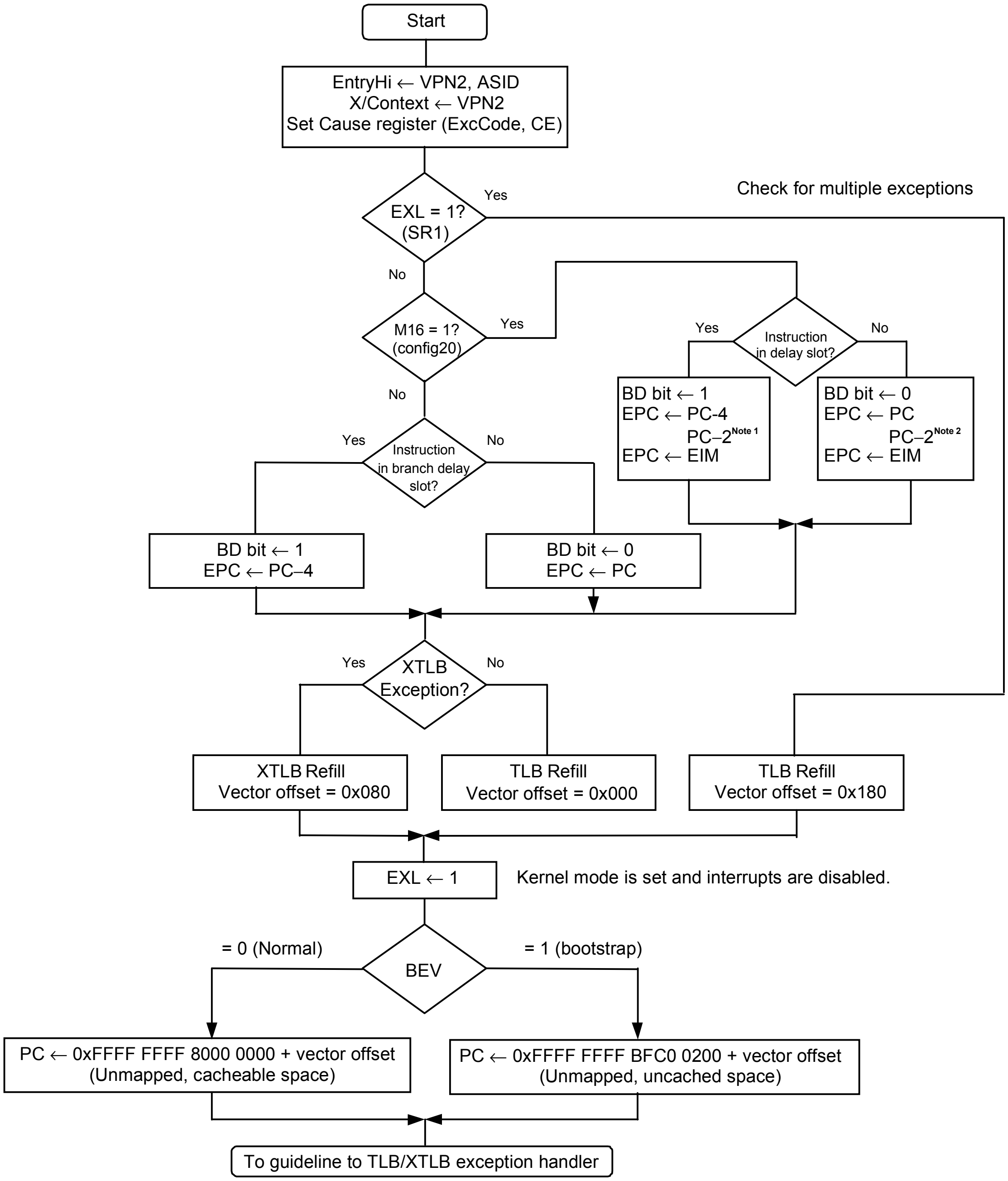

Execute MFC0 instruction
X/Context register
EPC register
Status register
Cause register
• The occurrence of TLB Refill, TLB Invalid, and TLB Modified exceptions is disabled by using an unmapped space.
• The occurrence of the Watch and Interrupt exceptions is disabled by setting EXL = 1.
• Other exceptions are avoided in the OS programs.
• However, the Cold Reset, Soft Reset, and NMI exceptions are enabled.
Execute MFC0 instruction
(Status bit setting)
KSU bit ← 00
EXL bit ← 0
IE bit ← 1
(In Kernel mode, interrupts are enabled.)
Check the Cause register, and jump to each routine
• 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.)
TS bit = 0?
No
The processor is reset.
Yes
Servicing by each exception routine
• The register files are saved.
EXL = 1
Execute MTC0 instruction
EPC register
Status register
ERET
• 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 7-18. TLB/XTLB Refill Exception Handling (1/2)

(a) Handling TLB/XTLB Refill exceptions (hardware)

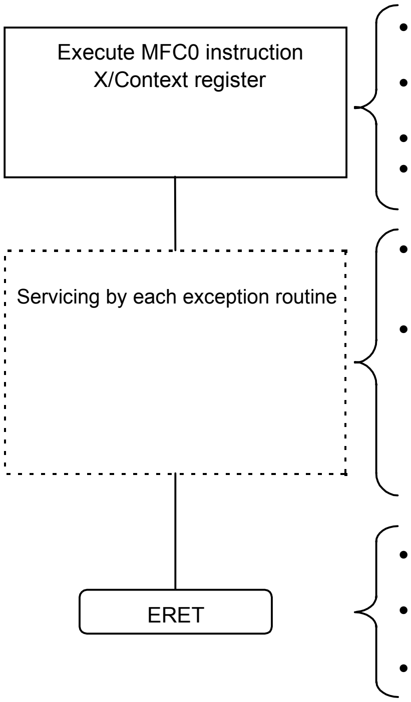


Notes 1. When the JR or JALR instruction of MIPS16 instructions

2. When the Extend instruction of MIPS16 instructions

Figure 7-18. TLB/XTLB Refill Exception Handling (2/2)

(b) Servicing TLB/XTLB Refill exceptions (software)

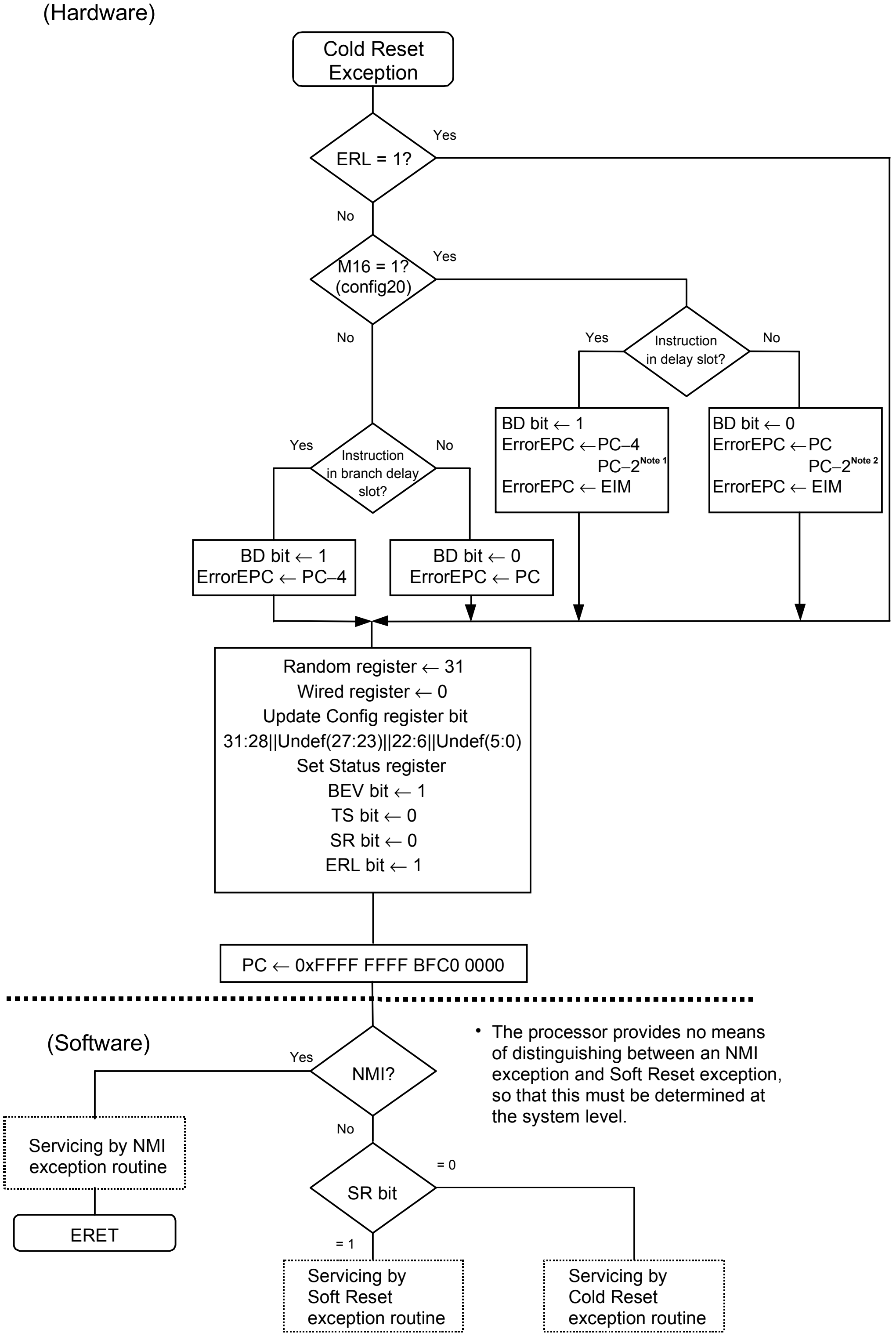


- The occurrence of TLB Refill, TLB Invalid, and TLB Modified exceptions is disabled by using an unmapped space.
- The occurrence of the Watch and Interrupt exceptions is disabled by setting EXL= 1.
- Other exceptions are avoided in the OS programs.
- However, the Cold Reset, Soft Reset, and NMI exceptions 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 7-19. Cold Reset Exception Handling

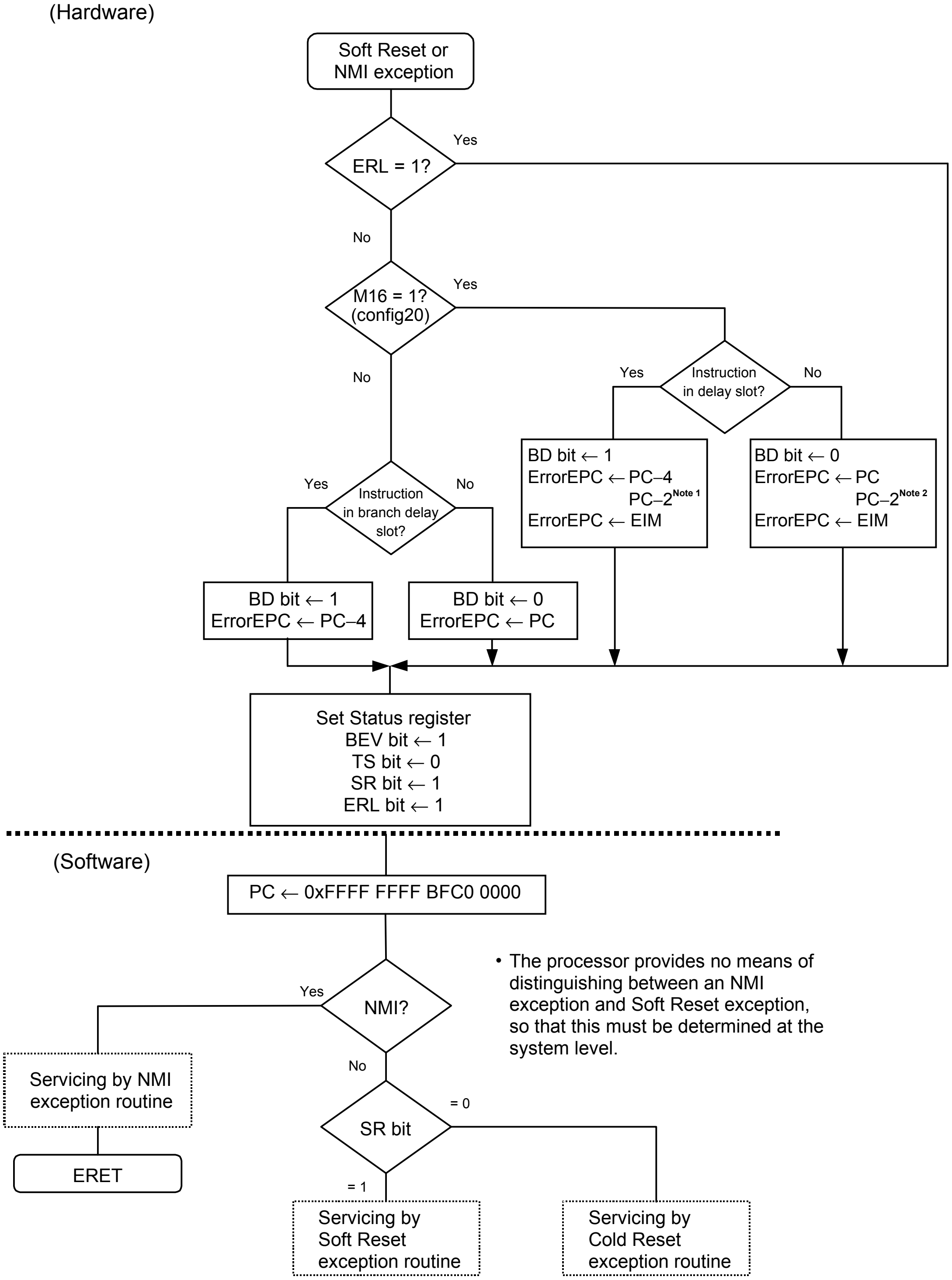


- The processor provides no means of distinguishing between an NMI exception and Soft Reset exception, so that this must be determined at the system level.

Notes 1. When the JR or JALR instruction of MIPS16 instructions

2. When the Extend instruction of MIPS16 instructions

Figure 7-20. 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.

Notes 1. When the JR or JALR instruction of MIPS16 instructions

2. When the Extend instruction of MIPS16 instructions

[MEMO]

CHAPTER 8 INITIALIZATION INTERFACE

This chapter describes the initialization interface and processor modes. It also explains the reset signal descriptions and types, signal- and timing-related dependence, and the initialization sequence during each mode that can be selected by the user.

Remark # that follows signal names indicates active low.

8.1 Reset Function

There are five ways to reset the VR4121. Each is summarized below.

8.1.1 RTC reset

During power-on, set the RTCRST# pin as active. After waiting (approx. 2 s) for the 32.768-kHz oscillator to begin ★ oscillating when the 3.3-V power supply is stable at 3.0 V or above, setting the RTCRST# pin as inactive causes the RTC unit to begin counting. Then, the states of the DBUS32/GPIO48, MIPS16EN, CLKSEL2/TxD, CLKSEL1/RTS#, and CLKSEL0/DTR# pins are read after one RTC cycle. Next, when the POWER pin, DCD# pin, or GPIO[3] pin becomes inactive, the VR4121 asserts the POWERON pin and uses the BATTINH/BATTINT# signal to perform a ★ battery level check. When starting through activation of the POWER pin, maintain the POWER pin at active for at least 32 ms to remove chatterings. If the battery check result is OK, the VR4121 asserts the MPOWER pin and waits for the stabilization time period (about 350 ms) for the external agent's DC/DC converter, then begins PLL oscillation and starts all clocks (a period of about 16 ms following the start of PLL oscillation is required for stabilization of PLL oscillation).

★ During an RTC reset, supplying voltage to the 2.5-V power-supply type (VDD2, VDDP, VDDPD) can be stopped to reduce the leak current. The following operation will not be affected by supplying voltage of 2.5 V to these power supplies within the period from when the MPOWER pin becomes active to when PLL starts oscillation.

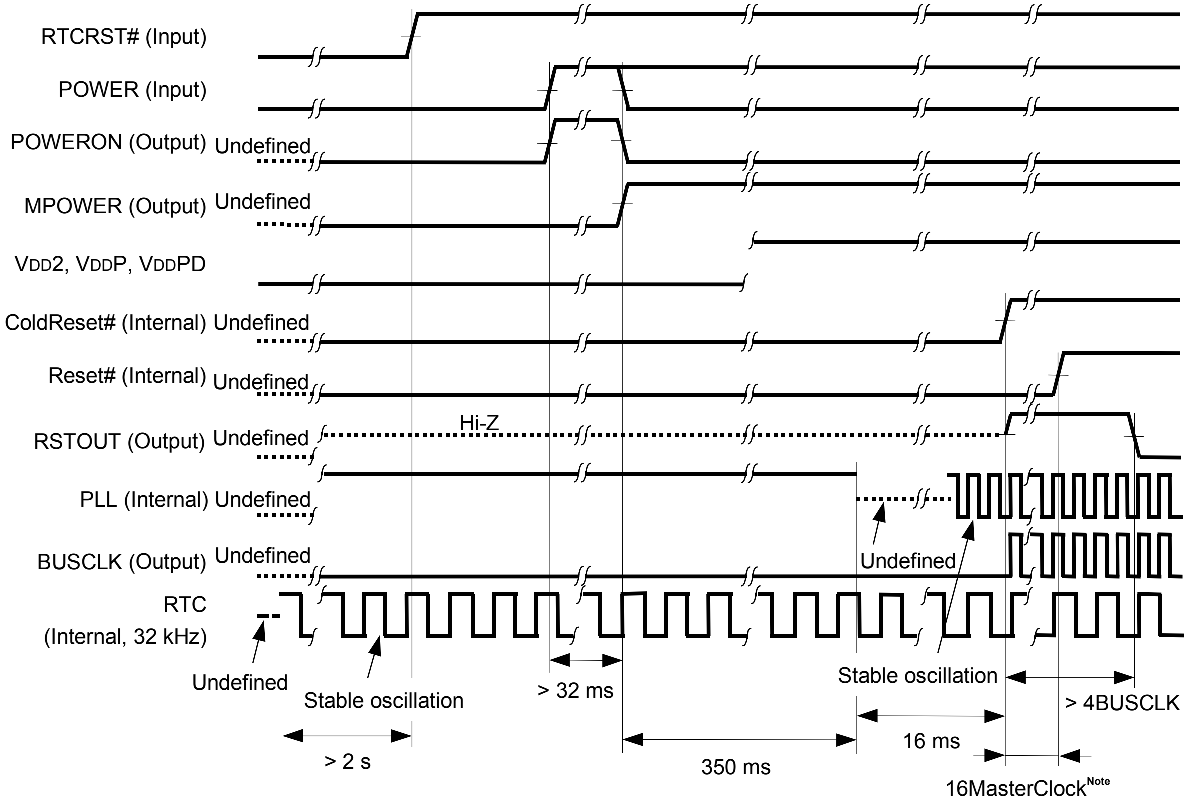
An RTC reset does not save any of the status information and it completely initializes the processor's internal state. Since the DRAM is not switched to self refresh mode, the contents of DRAM after an RTC reset are not at all guaranteed.

After a reset, the processor becomes the system bus master and it begins the Cold reset exception sequence to access the reset vectors in the ROM space. Since only part of the internal status is reset when a reset occurs in the VR4121, the processor should be completely initialized by software.

After Power-On, the processor's pin statuses are undefined since the RTCRST# becomes active, until the 32.768-kHz clock oscillator starts oscillation. The pin statuses after oscillation starts are described in **CHAPTER 2 PIN FUNCTION**.

Once high-level output occurs from the MPOWER pin, the VR4121 does not output at high level from the RSTOUT pin until the POWER pin stabilization standby time, activation standby time, and PLL oscillation stabilization time have elapsed. High-level output from RSTOUT is output at least once per 4BUSCLK cycle. This indicates that RSTOUT has gone to high level within (at most) the period of the 3BUSCLK cycle following the start of vector address fetch operation for initialization. Accordingly, even when RSTOUT is active, the system should be configured so as to enable execution of ROM fetch operations.

Figure 8-1. RTC Reset



Note MasterClock is a base clock used in the CPU core.

When configuring a system that includes the VR4121, we recommend that, in addition to the power supply to the VR4121, at least one or two other 3.3-V power supplies should be made available to the system.

In the VR4121, the power supply control method that is used during an RTC reset differs depending on the type of DRAM being used. The type of DRAM to be used is set via the SMODE(2:1) pin during an RTC reset.

MPOWER is used as the power supply control signal for external devices when using EDO-type DRAM. MPOWER and SPOWER/GPIO7 are used as this signal when using SDRAM.

Power supply control for both types of DRAM is described below.

★ **(1) When using EDO-type DRAM**

In addition to the power supply to the VR4121, make at least one other 3.3-V power supply available to the system. A power supply configuration example is shown below.

Example

3.3-V power supply for VR4121

Application: Power supply for VR4121 operation and DRAM data retention

Target devices for power supply: VR4121 (3.3-V power supply), DRAM (EDO type)

3.3-V power supply A

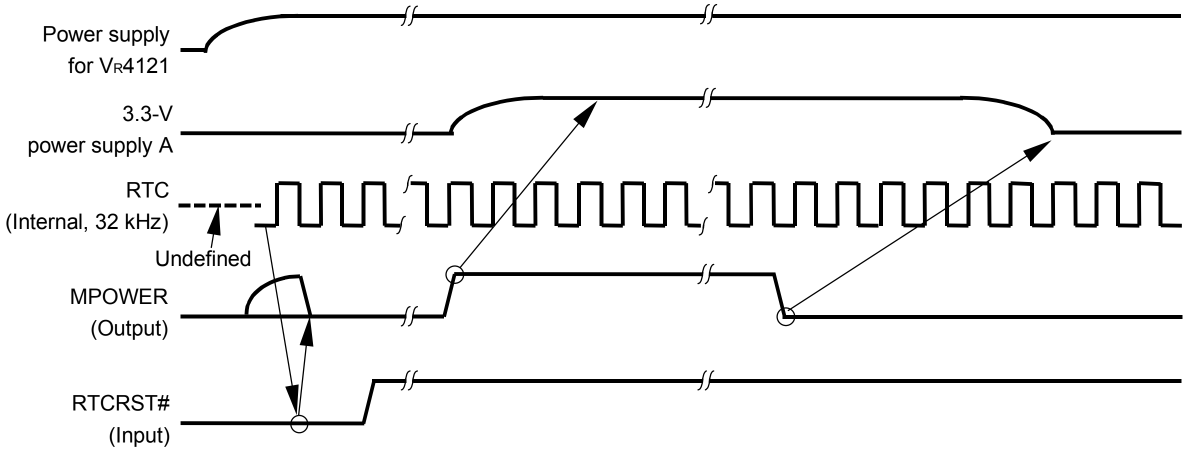
Application: Power consumption reduction

Target devices for power supply: ROM, External LSI

Power supply control pin: MPOWER

The power supply control timing when using EDO type DRAM is shown below.

Figure 8-2. Power Supply Control Timing When Using EDO Type DRAM



★ (2) When using SDRAM

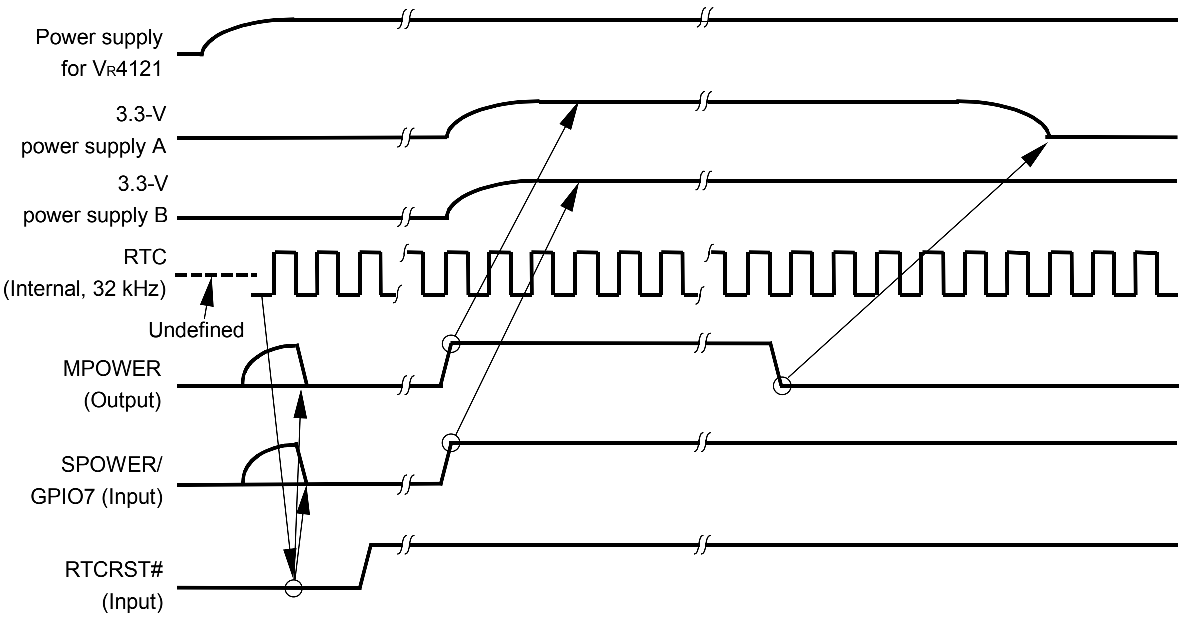
In addition to the 3.3-V power supply to the VR4121, make at least two other 3.3-V power supply available to the system. A power supply configuration example is shown below.

Example

3.3-V power supply for VR4121
Application: Power supply for VR4121 operation and DRAM data retention
Target devices for power supply: VR4121 (3.3-V power supply), SDRAM
3.3-V power supply A
Application: Power consumption reduction
Target devices for power supply: SROM/ROM, External LSI
Power supply control pin: MPOWER
3.3-V power supply B
Application: Power consumption reduction and device protection
Target devices for power supply: SDRAM
Power supply control pin: SPOWER/GPIO7

The power supply control timing when using SDRAM is shown below.
When using SDRAM, once the RTC reset sequence begins, the VR4121 sets the SPOWER/GPIO7 pin to low level and specifies suspending the power supply to SDRAM. After that, synchronization enables the MPOWER pin's transition to high level, at which time the SPOWER/GPIO7 pin also goes to high level. From then on, the SPOWER/GPIO7 pin is held at high level even when changing to shut-down or Hibernate mode so as to continue supplying power to SDRAM.

Figure 8-3. Power Supply Control Timing When Using SDRAM



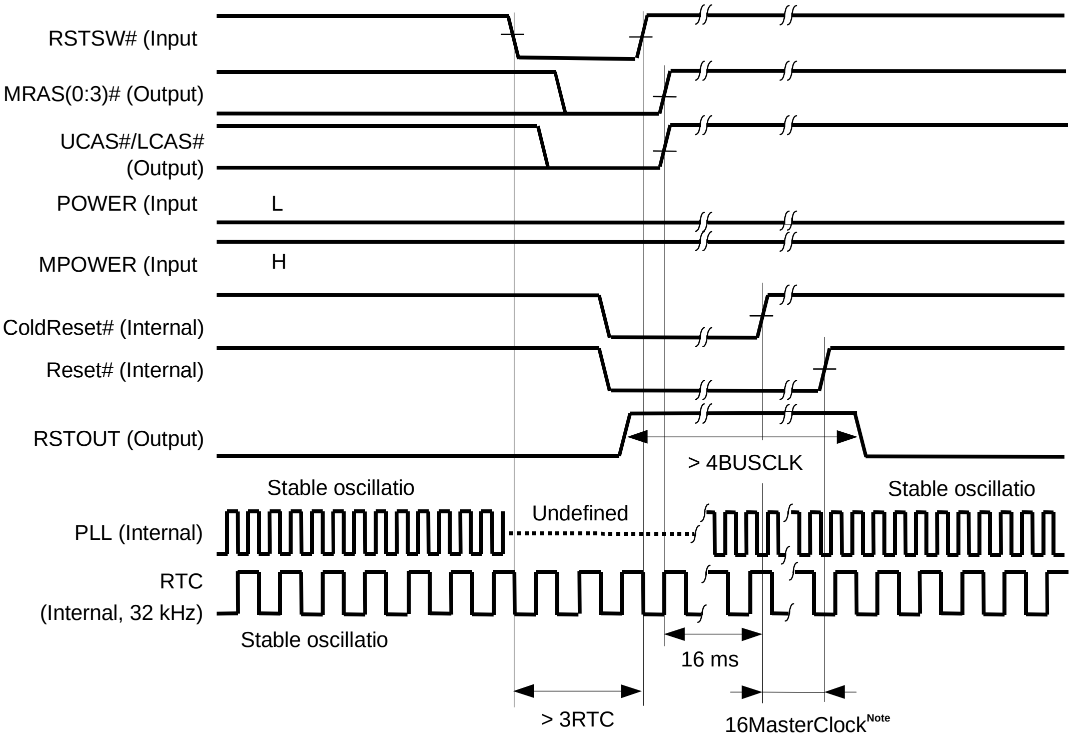
8.1.2 RSTSW

After the RSTSW# pin becomes active and then becomes inactive 100 μs later, the VR4121 starts PLL oscillation and starts all clocks (a period of about 16 ms following the start of PLL oscillation is required for stabilization of PLL oscillation).

A reset by RSTSW initializes the entire internal state except for the RTC timer and the PMU.

After a reset, the processor becomes the system bus master and it begins the Cold reset exception sequence to access the reset vectors in the ROM space. Since only part of the internal status is reset when a reset occurs in the VR4121, the processor should be completely initialized by software.

Figure 8-4. RSTSW



Note MasterClock is a base clock used in the CPU core.

Caution **If the RSTSW# signal becomes active at the same time the CPU transits to the Hibernate mode, the CPU may be activated without asserting the POWER signal after the MPOWER signal becomes inactive.**

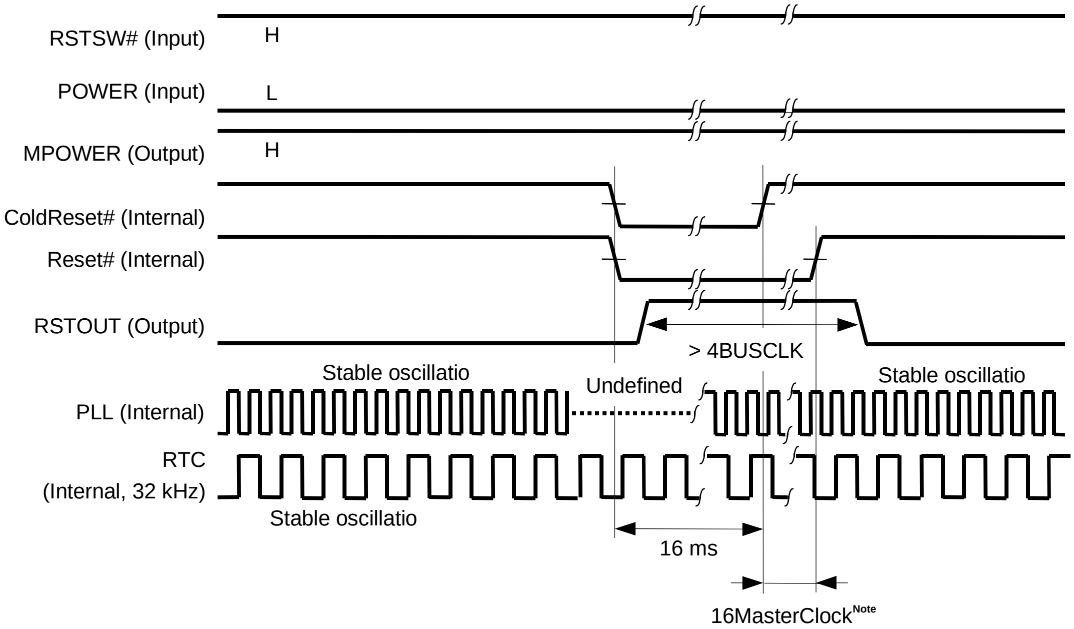
8.1.3 Deadman's switch

After the Deadman's switch unit is enabled, if the Deadman's switch is not cleared within the specified time period, the VR4121 is immediately returned to reset status. Setting and clearing of the Deadman's switch is performed by software.

A reset by the Deadman's switch initializes the entire internal state except for the RTC timer and the PMU. Since the DRAM is not switched to self refresh mode, the contents of DRAM after a Deadman's switch reset are not at all guaranteed.

After a reset, the processor becomes the system bus master and it begins the Cold reset exception sequence to access the reset vectors in the ROM space. Since only part of the internal status is reset when a reset occurs in the VR4121, the processor should be completely initialized by software.

Figure 8-5. Deadman's Switch



Note MasterClock is a base clock used in the CPU core.

8.1.4 Software shutdown

When the software executes the HIBERNATE instruction, the VR4121 sets the DRAM to self refresh mode and sets the MPOWER pin as inactive, then enters reset status. Recovery from reset status occurs when the POWER pin is asserted, when a WakeUpTimer interrupt occurs, when the DCD# pin is asserted, or when the GPIO(0:3), GPIO(9:12) pins are asserted.

★ During a software shutdown, supplying voltage to the 2.5-V power-supply systems ($V_{DD}2$, $V_{DD}P$, $V_{DD}PD$) can be stopped to reduce the leakage current. The following operation will not be affected by supplying voltage of 2.5 V to these power supplies within the period from when the MPOWER pin becomes active to when PLL starts oscillation.

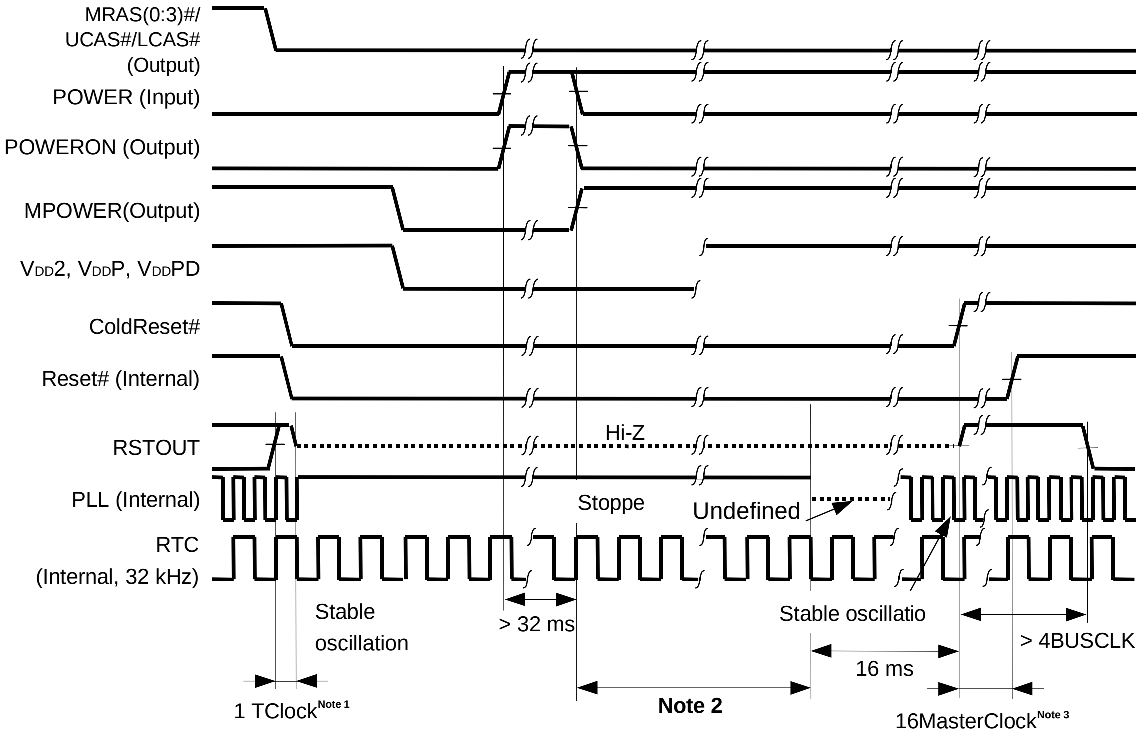
A reset by software shutdown initializes the entire internal state except for the RTC timer and the PMU.

After a reset, the processor becomes the system bus master and it begins the Cold reset exception sequence to access the reset vectors in the ROM space. Since only part of the internal status is reset when a reset occurs in the VR4121, the processor should be completely initialized by software.

When switching from Full-speed mode to Hibernate mode, the VR4121 enters self refresh mode in order to protect
★ data in DRAM. At that time, the RSTOUT pin enters the high impedance state in order to prepare the power-off of connected external devices after a high-level is output for 1 TClock from the RSTOUT pin. Once the RSTOUT pin enters the high impedance state, at least 2.5 RTC clock cycles must elapse before output to the MPOWER pin goes to low level.

When recovering from a shutdown, once high-level output occurs from the MPOWER pin, output at high level from the RSTOUT pin does not begin until the POWER pin stabilization standby time, activation standby time, and PLL oscillation stabilization time have elapsed. High-level output from RSTOUT is output at least once per 4BUSCLK cycle. This indicates that RSTOUT has gone to high level within (at most) the 3BUSCLK cycle following the start of vector address fetch operation for initialization. Accordingly, even when RSTOUT is active, the system should be configured so as to enable execution of ROM fetch operations.

★ **Figure 8-6. Software Shutdown**



Notes 1. TClock is a clock used to access internal peripheral units.

2. Wait time for activation. It can be changed by setting P MUWAITREG (see **16.2.5 PMUWAITREG (0x0B00 00A8)**).

3. MasterClock is the basic clock used in the CPU core.

8.1.5 HALTimer shutdown

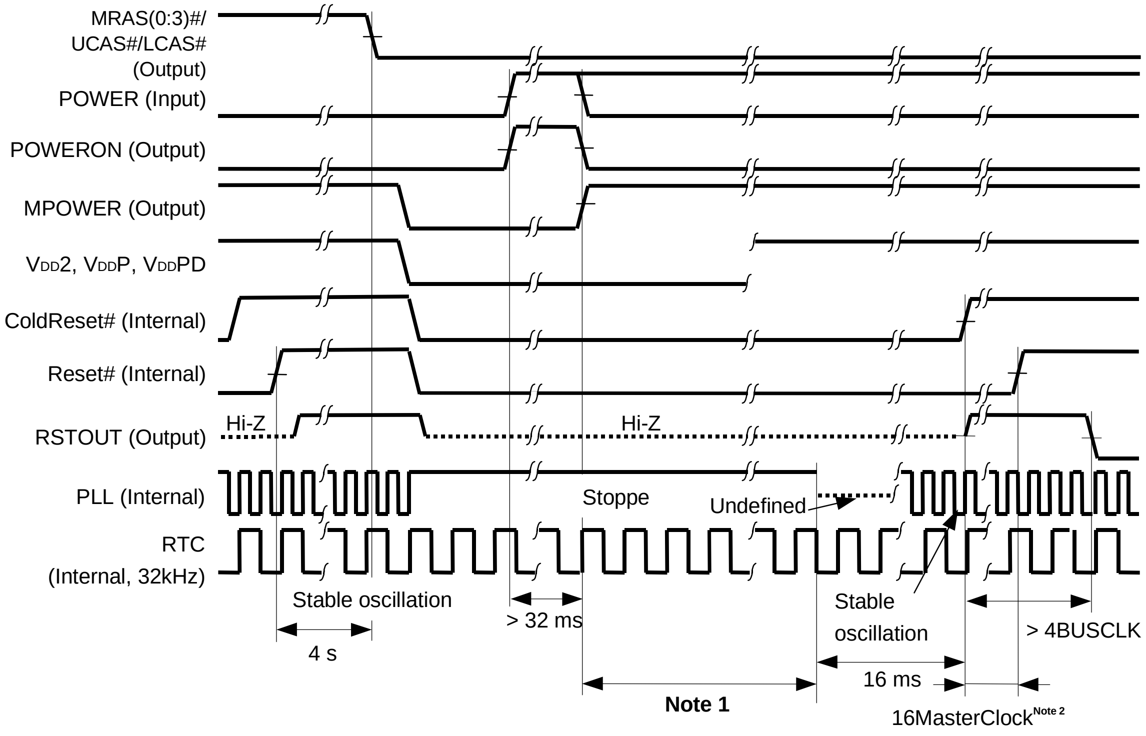
After an RTC reset is canceled, if the HAL timer is not canceled by software within about four seconds (the HALTIMERRST bit of the PMUCNTREG register is not set to 1), the V R4121 enters reset status (see **16.1.2 Shutdown control**). Recovery from reset status occurs when the POWER pin is asserted or when a WakeUpTimer interrupt occurs.

During a HALTimer shutdown, supplying voltage to the 2.5-V power-supply systems (VDD2, VDDP, VDDPD) can be ★ stopped to reduce the leak current. The following operation will not be affected by supplying voltage of 2.5 V to these power supplies within the period from when the MPOWER pin becomes active to when PLL starts oscillation. However, this function is available from version 2.0.

A reset by HAL timer initializes the entire internal state except for the RTC timer and the PMU.

After a reset, the processor becomes the system bus master and it begins the Cold reset exception sequence to access the reset vectors in the ROM space. Since only part of the internal status is reset when a reset occurs in the VR4121, the processor should be completely initialized by software.

Figure 8-7. HALTimer Shutdown



Notes 1. Wait time for activation. It can be changed by setting P MUWAITREG (see **16.2.5 PMUWAITREG (0x0B00 00A8)**).

2. MasterClock is the basic clock used in the CPU core.

8.2 Power-on Sequence

The factors that cause the VR4121 to switch from hibernate mode or shutdown mode to full speed mode are called power-on factors. There are four power-on factors: assertion of the POWERON pin, assertion of the DCD# pin, activation of the wakeup timer, and assertion of the GPIO pins (GPIO(0:3), GPIO(9:12)). When an activation factor occurs, the VR4121 asserts the POWERON pin, then provides notification to external agents that the VR4121 is ready for power-on. Three RTC clocks after the POWERON pin is asserted, the VR4121 checks the state of the BATTINH/BATTINT# pin. If the BATTINH/BATTINT# pin's state is low, the POWERON pin is deasserted one RTC clock after the BATTINH/BATTINT# pin check is completed, then the VR4121 is not activated. If the BATTINH/BATTINT# pin's state is high, the POWERON pin is deasserted three RTC clocks after the BATTINH/BATTINT# pin check is completed, then the MPOWER pin is asserted and the VR4121 is activated.

★ While the MPOWER pin is inactive, supplying voltage to the 2.5-V power-supply systems ($V_{DD}2$, $V_{DD}P$, $V_{DD}PD$) can be stopped to reduce the leak current. The following operation will not be affected by supplying voltage of 2.5 V to these power supplies within the period from when the MPOWER pin becomes active to when PLL starts oscillation. However, this function is available from version 2.0.

Figure 8-8 shows a timing chart of VR4121 activation and Figure 8-9 shows a timing chart of when activation fails due to the BATTINH/BATTINT# pin's “low” state.

Figure 8-8. VR4121 Activation Sequence (When Battery Check Is OK)

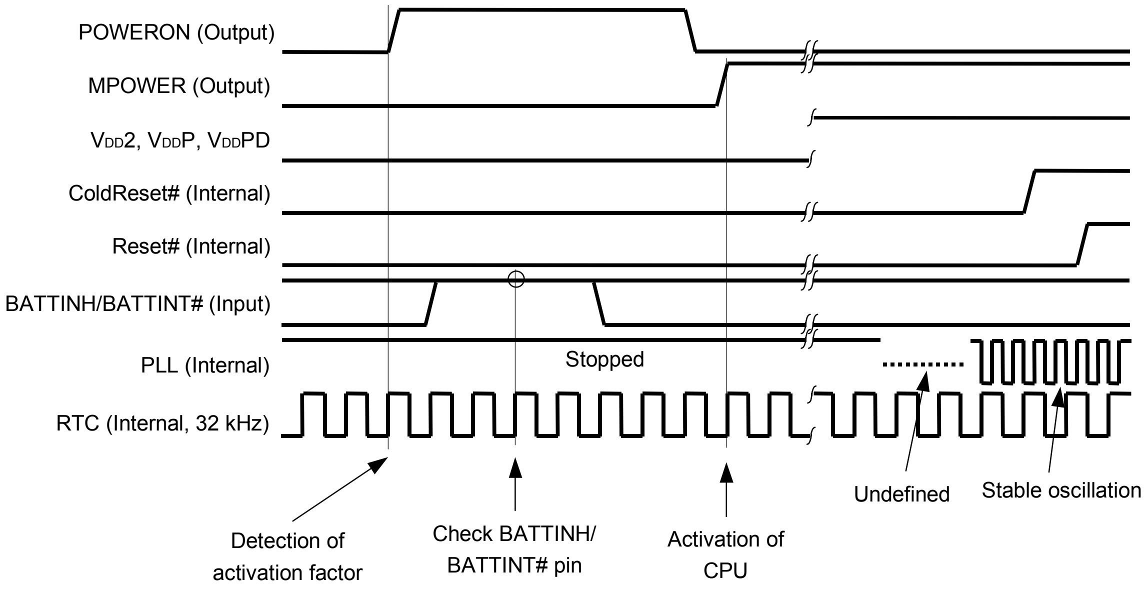
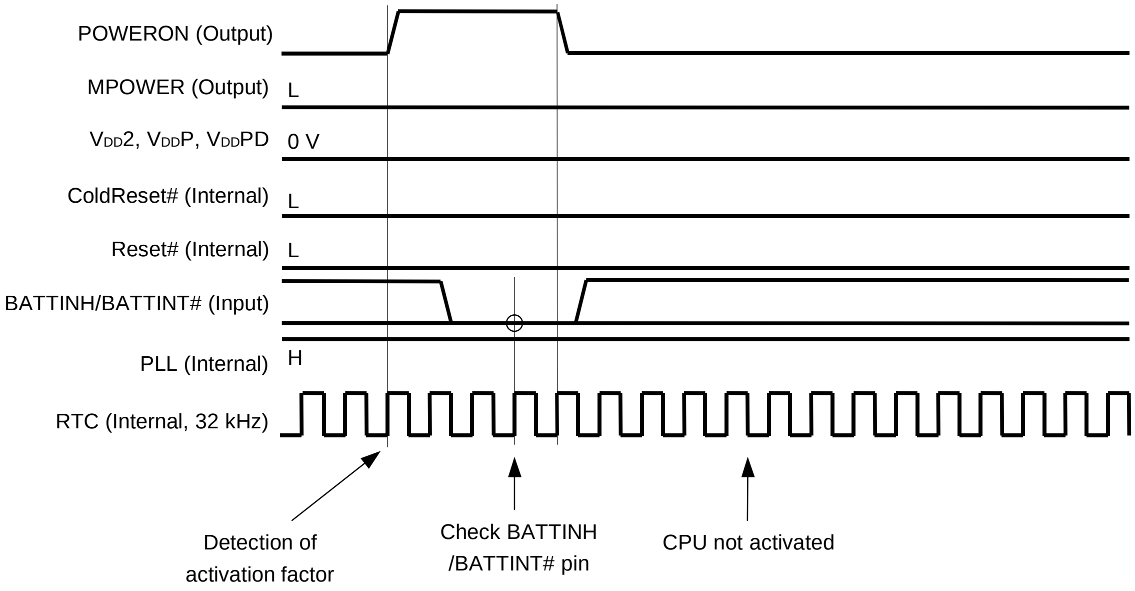


Figure 8-9. VR4121 Activation Sequence (When Battery Check Is NG)



8.3 Reset of the CPU Core

This section describes the reset sequence of the VR4120 CPU core. For details about factors of reset or reset of the whole VR4121, refer to **8.1 Reset Function** and **CHAPTER 16 PMU (POWER MANAGEMENT UNIT)**.

8.3.1 Cold reset

In the VR4121, a cold reset sequence is executed in the CPU core in the following cases:

- RTC reset
- RSTSW reset
- Deadman's SW shutdown
- Software shutdown
- HAL Timer shutdown
- Battery low shutdown
- Battery lock release shutdown

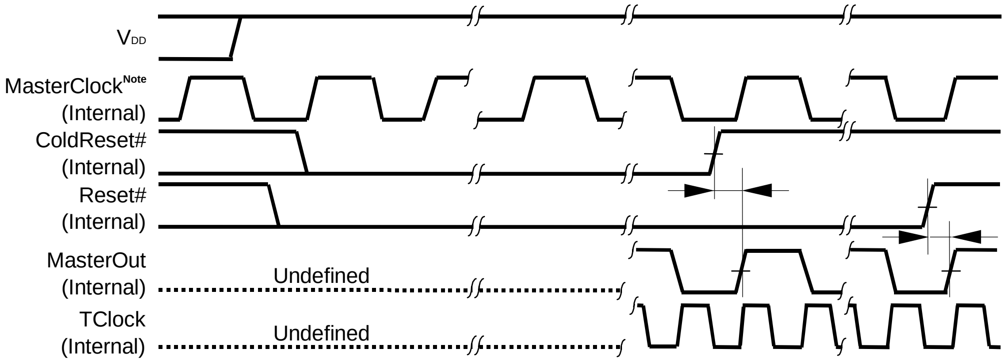
A Cold Reset completely initializes the CPU core, except for the following register bits.

- 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 0x04800; the other bits are undefined.
- The values of the other registers are undefined.

Once power to the processor is established, the ColdReset# (internal) and the Reset# (internal) signals are asserted and a Cold Reset is started. After approximately 2 ms assertion, the ColdReset# signal is deasserted synchronously with MasterOut. Then the Reset# signal is deasserted synchronously with MasterOut, and the Cold Reset is completed.

Upon reset, the CPU core becomes bus master and drives the SysAD bus (internal). After Reset# is deasserted, the CPU core branches to the Reset exception vector and begins executing the reset exception code.

Figure 8-10. Cold Reset



Note MasterClock is the basic clock used in the CPU core.

8.3.2 Soft reset

Caution Soft Reset is not supported in the present V_R4121.

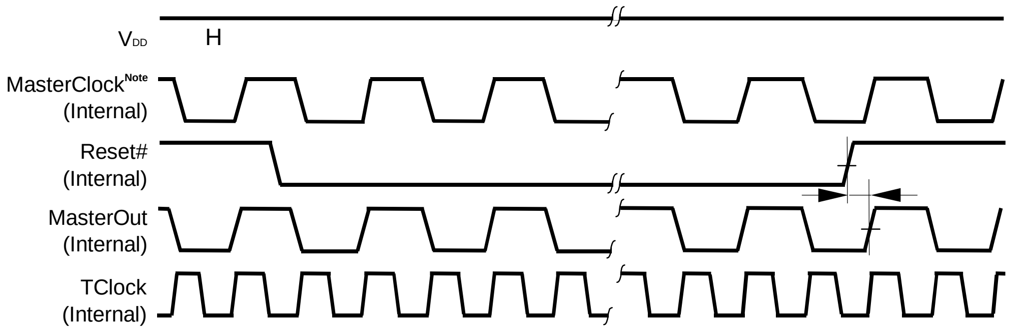
A Soft Reset initializes the CPU core without affecting the clocks; in other words, a Soft Reset is a logic reset. In a Soft Reset, the CPU core retains as much state information as possible; all state information except for the following is retained:

- 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.

A Soft Reset is started by assertion of the Reset# signal, and is completed at the deassertion of the Reset# signal synchronized with MasterOut. In general, data in the CPU core is preserved for debugging purpose.

Upon reset, the CPU core becomes bus master and drives the SysAD bus (internal). After Reset# is deasserted, the CPU core branches to the Reset exception vector and begins executing the reset exception code.

Figure 8-11. Soft Reset



Note MasterClock is the basic clock used in the CPU core.

8.4 VR4121 Processor Modes

The VR4121 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.

8.4.1 Power modes

The VR4121 supports four power modes: Fullspeed mode, Standby mode, Suspend mode, and Hibernate mode.

(1) Fullspeed mode

This is the normal operation mode.

The VR4121's default status sets operation under Fullspeed mode. After the processor is reset, the VR4121 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 VR4121 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 VR4121 stops supplying TClock to peripheral units. Accordingly, during Suspend mode peripheral units can only be activated by a special interrupt unit (DCD# control, etc.). While in this mode, the register and cache contents are retained.

When the SUSPEND instruction completes the WB stage, the VR4121 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 VR4121 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 returns from Hibernate mode to Fullspeed mode by power-pin activation, Wakeup timer interrupt generation, DCD# pin activation, or GPIO(0:3) or GPIO(9:12) pin activation.

In this mode, supplying voltage to the 2.5-V power-supply systems ($V_{DD}2$, $V_{DD}P$, $V_{DD}PD$) 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).

8.4.2 Privilege mode

The V_R4121 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.

8.4.3 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 V_R4121 supports the little-endian order only.

8.4.4 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 0xFFFF FFFF BFC0 0200 and the ordinary execution vector is changed to address 0xFFFF FFFF BFC0 0380.

When the BEV bit is cleared, the TLB miss exception vector's address is changed to 0xFFFF FFFF 8000 0000 and the ordinary execution vector is changed to address 0xFFFF FFFF 8000 0180.

8.4.5 Cache error check

The Status register's CE bit has no meaning because the VR4121 does not support cash parity.

8.4.6 Parity error prohibit

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

8.4.7 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.

[MEMO]

CHAPTER 9 CACHE MEMORY

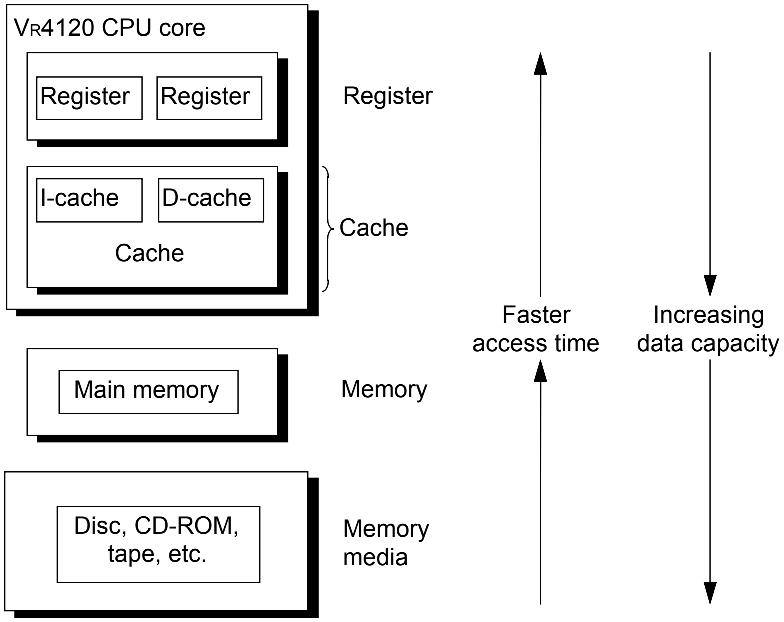
This chapter describes in detail the cache memory: its place in the VR4120 CPU core memory organization, and individual organization of the caches.

9.1 Memory Organization

Figure 9-1 shows the VR4120 CPU 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 9-1 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 9-1. Logical Hierarchy of Memory



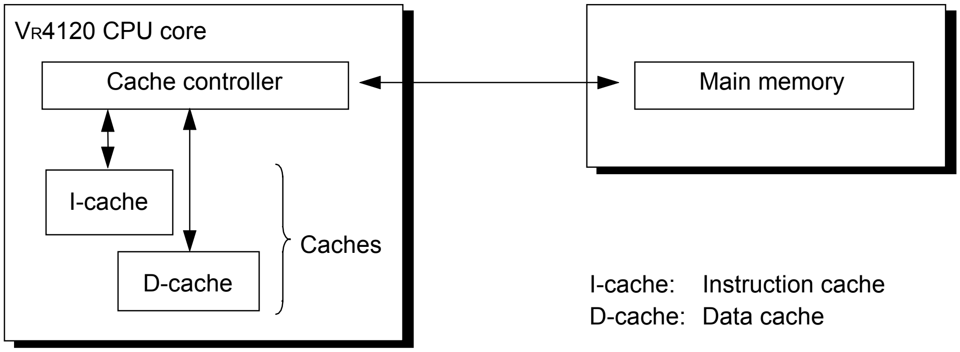
The VR4120 CPU core has two on-chip caches: one holds instructions (the instruction cache), the other holds data (the data cache). The instruction and data caches can be read in one PClock cycle.

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.

9.2 Cache Organization

This section describes the organization of the on-chip data and instruction caches. Figure 9-2 provides a block diagram of the VR4120 CPU core cache and memory model.

Figure 9-2. 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 **9.2.1 Organization of the instruction cache (I-cache)** and **9.2.2 Organization of the data cache (D-cache)**.

(2) Cache Sizes

The instruction cache in the VR4120 CPU core is 16 Kbytes; the data cache is 8 Kbytes.

9.2.1 Organization of the instruction cache (I-cache)

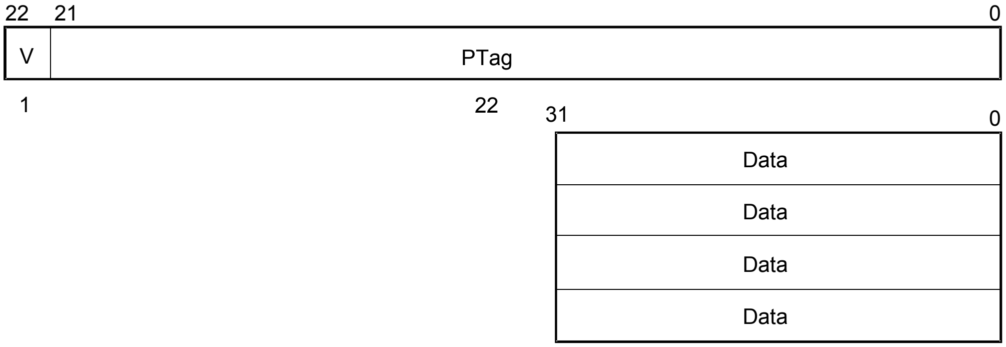
Each line of I-cache data (although it is actually an instruction, it is referred to as data to distinguish it from its tag) has an associated 23-bit tag that contains a 22-bit physical address, and a single valid bit.

The VR4120 CPU 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 9-3 shows the format of a 4-word (16-byte) I-cache line.

Figure 9-3. Instruction Cache Line Format



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

9.2.2 Organization of the data cache (D-cache)

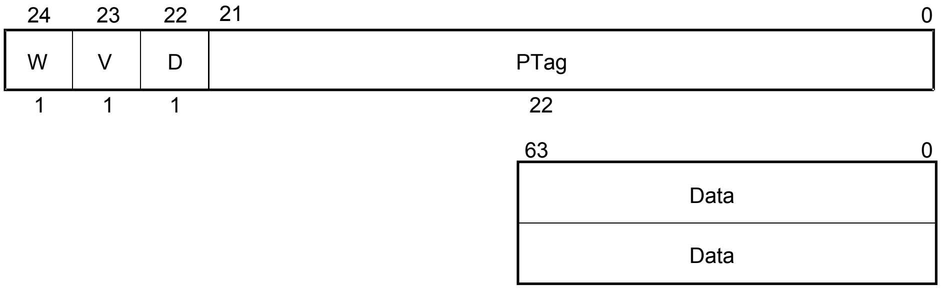
Each line of D-cache data has an associated 25-bit tag that contains a 22-bit physical address, a Valid bit, a Dirty bit, and a Write-back bit.

The VR4120 CPU 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 9-4 shows the format of a 4-word (16-byte) D-cache line.

Figure 9-4. 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

9.2.3 Accessing the caches

Figure 9-5 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.

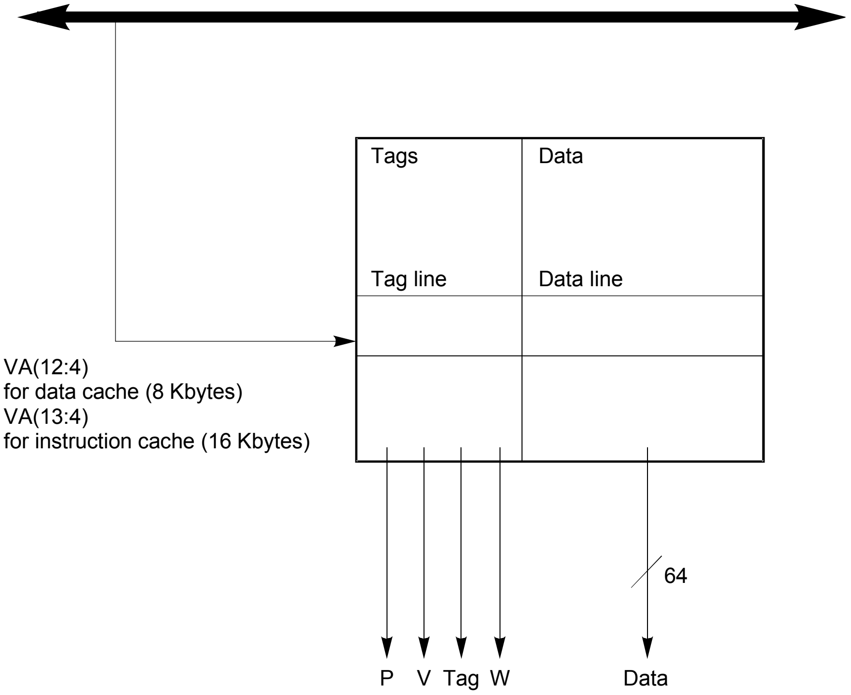
(1) Data cache addressing

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 9-5. Cache Data and Tag Organization



9.3 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.

9.3.1 Cache write policy

The VR4120 CPU 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 VR4121 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.

9.4 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 **CHAPTER 28 MIPS III INSTRUCTION SET DETAILS**.

9.5 Cache State Transition Diagrams

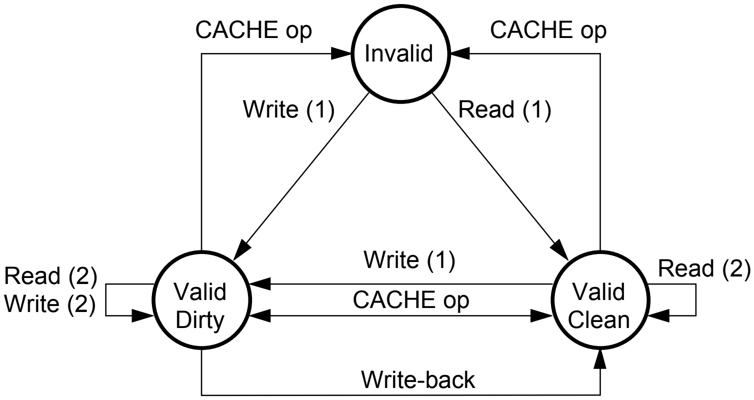
The following section describes the cache state diagrams for the data and instruction cache lines. These state diagrams do not cover the initial state of the system, since the initial state is system-dependent.

9.5.1 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 9-6. Data Cache State Diagram

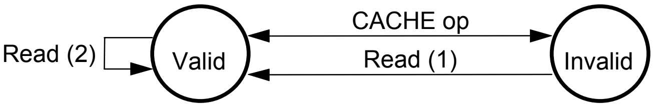


9.5.2 Instruction cache state transition

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 9-7. Instruction Cache State Diagram



9.6 Cache Data Integrity

Figures 9-8 to 9-22 shows checking operations for various cache accesses.

Figure 9-8. Data Check Flow on Instruction Fetch

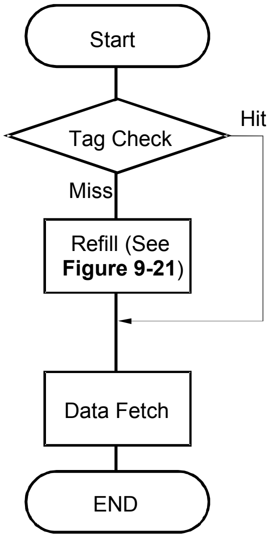


Figure 9-9. Data Check Flow on Load Operations

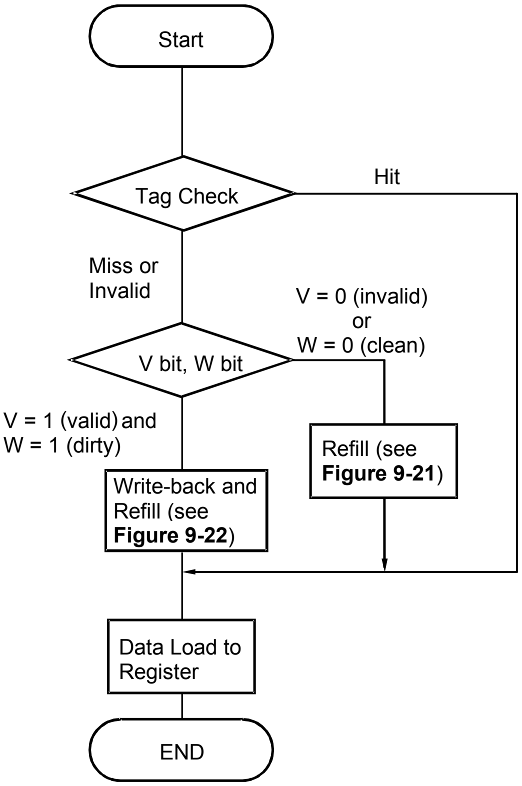

Start
Tag Check
Hit
Miss or Invalid
V = 0 (invalid) or W = 0 (clean)
V bit, W bit
V = 1 (valid) and W = 1 (dirty)
Refill (see **Figure 9-21**)
Write-back and Refill (see **Figure 9-22**)
Data Load to Register
END

Figure 9-10. Data Check Flow on Store Operations

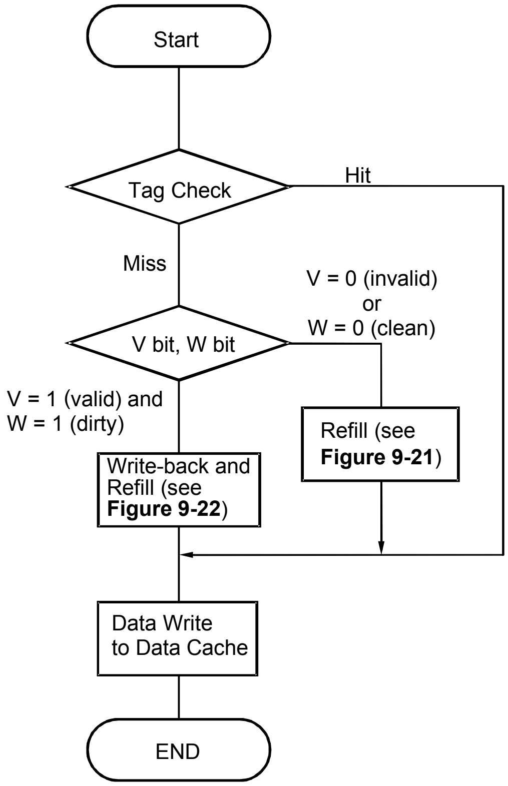


Figure 9-11. Data Check Flow on Index_Invalidate Operations

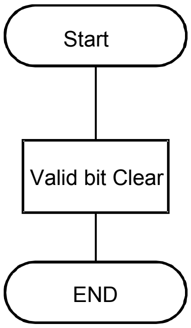


Figure 9-12. Data Check Flow on Index_Writeback_Invalidate Operations

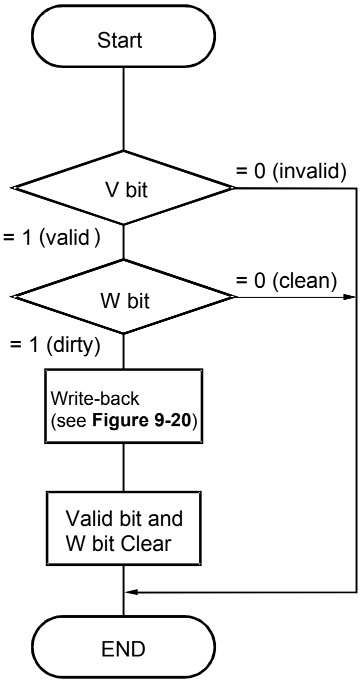


Figure 9-13. Data Check Flow on Index_Load_Tag Operations

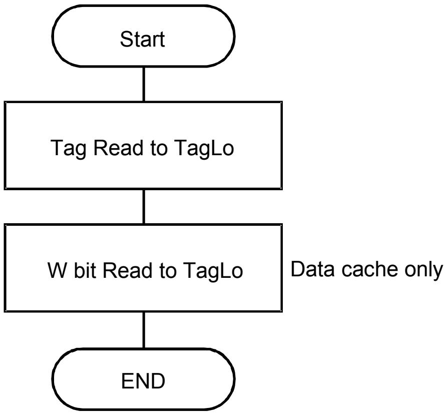


Figure 9-14. Data Check Flow on Index_Store_Tag Operations

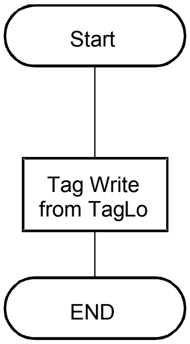


Figure 9-15. Data Check Flow on Create_Dirty Operations

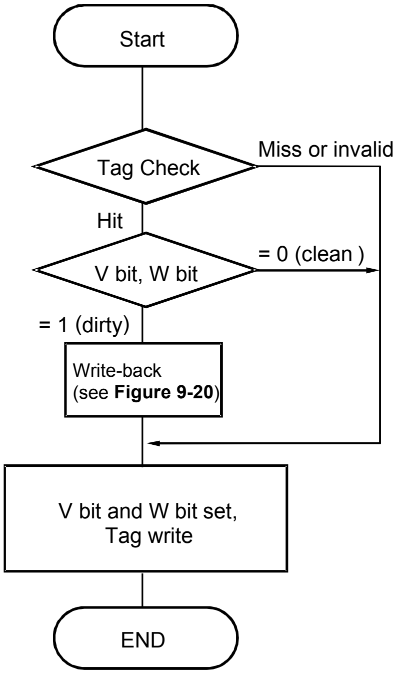


Figure 9-16. Data Check Flow on Hit_Invalidate Operations

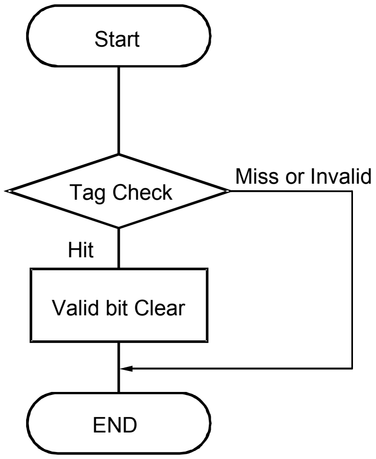


Figure 9-17. Data Check Flow on Hit_Writeback_Invalidate Operations

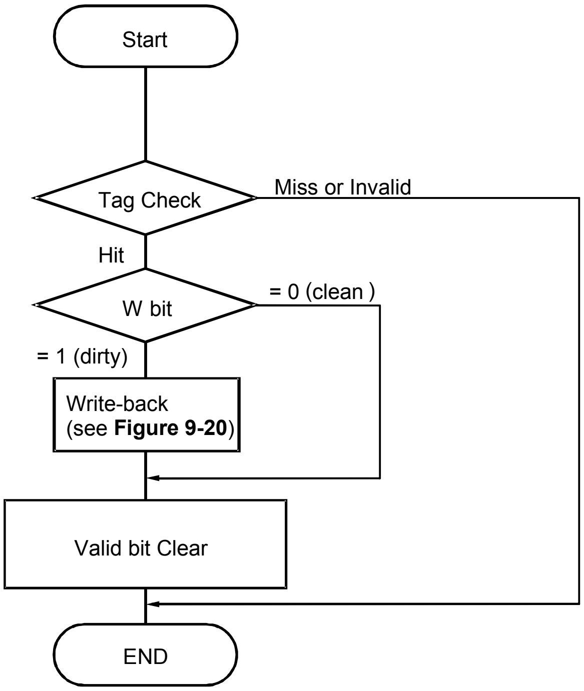


Figure 9-18. Data Check Flow on Fill Operations

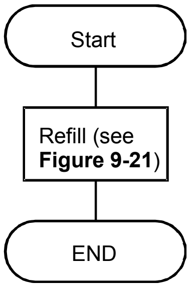


Figure 9-19. Data Check Flow on Hit_Writeback Operations

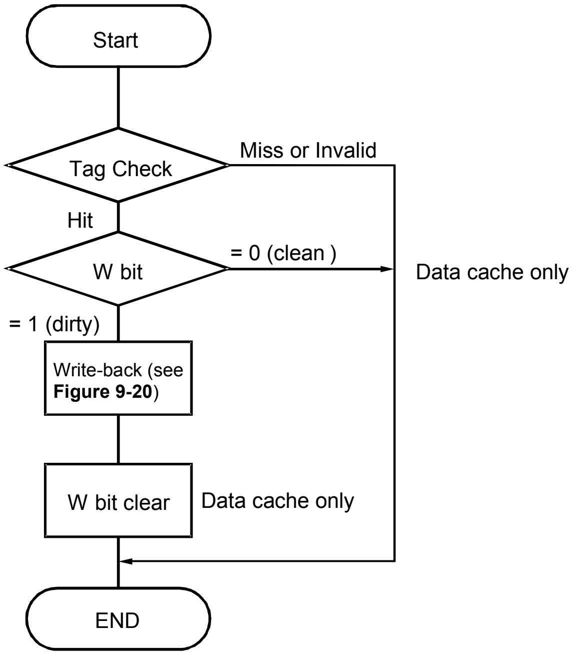


Figure 9-20. Writeback Flow

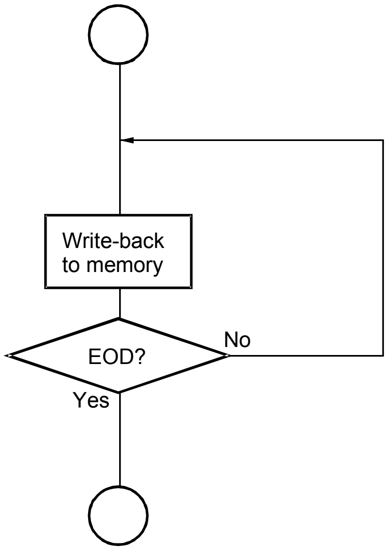


Figure 9-21. Refill Flow

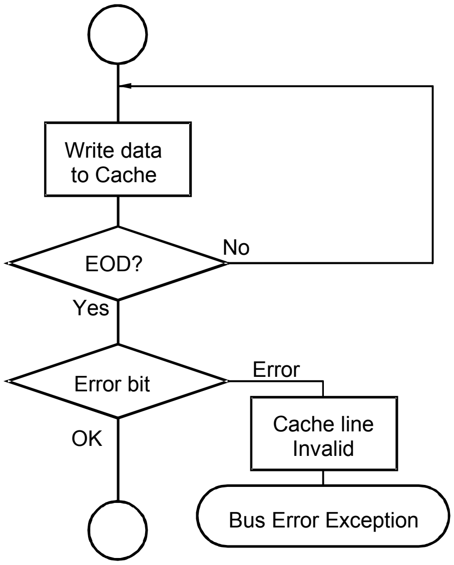
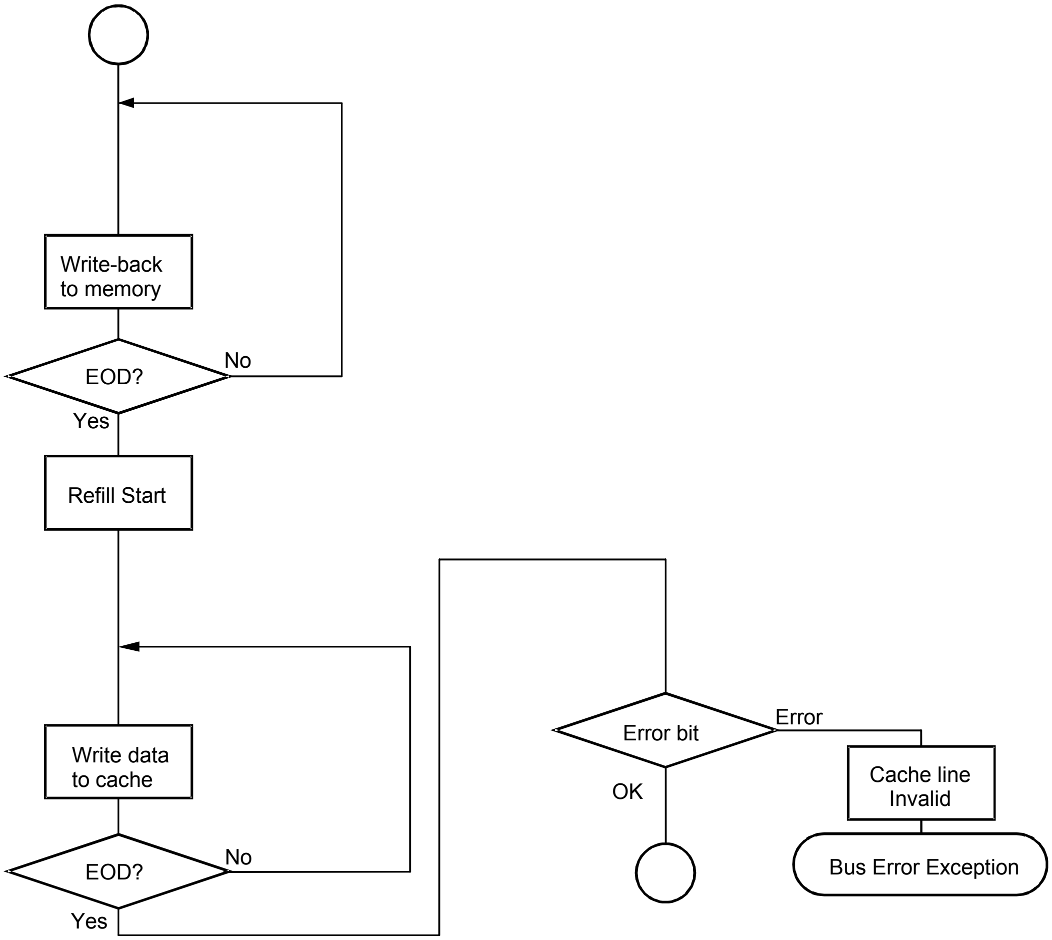


Figure 9-22. Writeback & Refill Flow



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.

9.7 Manipulation of the Caches by an External Agent

The VR4121 does not provide any mechanisms for an external agent to examine and manipulate the state and contents of the caches.

[MEMO]

CHAPTER 10 CPU CORE INTERRUPTS

Four types of interrupt are available on the CPU core. These are:

(1) one non-maskable interrupt, NMI
(2) five ordinary interrupts
(3) two software interrupts
(4) one timer interrupt

For the interrupt request input to the CPU core, see **CHAPTER 15 ICU (INTERRUPT CONTROL UNIT)**.

10.1 Non-maskable Interrupt (NMI)

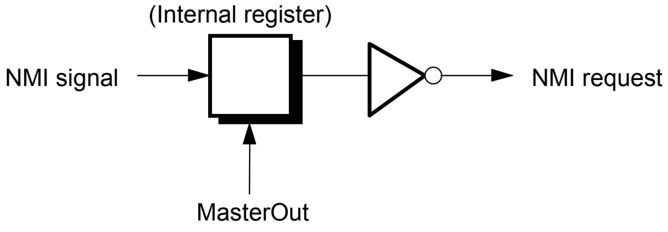
The non-maskable interrupt is acknowledged by asserting the NMI signal (internal), forcing the processor to branch to the Reset Exception vector. This signal is latched into an internal register at the rising edge of MasterOut signal (internal), as shown in Figure 10-1.

NMI only takes effect when the processor pipeline is running.

This interrupt cannot be masked.

Figure 10-1 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 10-1. Non-maskable Interrupt Signal



10.2 Ordinary Interrupts

Ordinary interrupts are acknowledged by asserting the Int(4:0) signals (internal). However, Int4 never occurs in the VR4121.

★ This interrupt request can be masked with the IM(6:2), IE, EXL, and ERL fields of the Status register.

10.3 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, EXL, and ERL fields of the Status register.

10.4 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 request is maskable through the IM7 bit, IE, EXL, and ERL fields of the Status register.

10.5 Asserting Interrupts

10.5.1 Detecting hardware interrupts

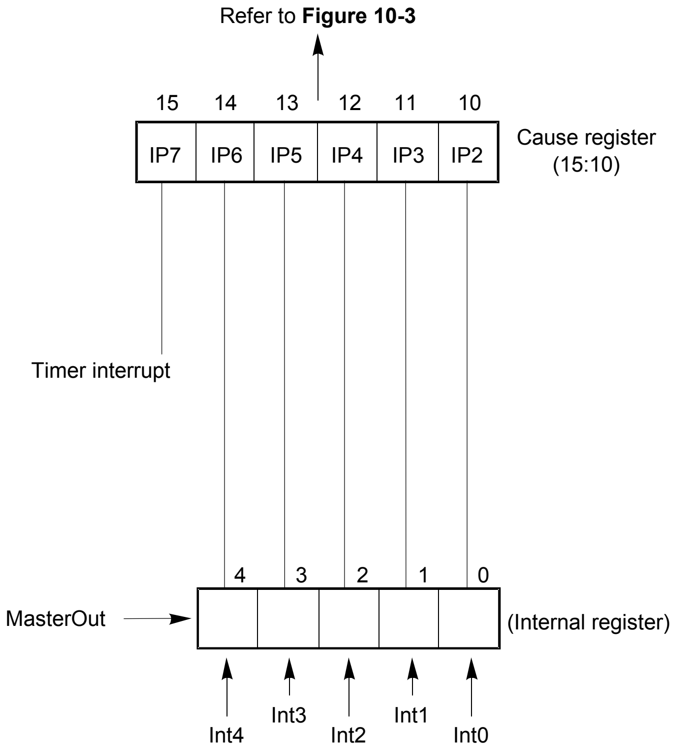
Figure 10-2 shows how the hardware interrupts are readable through the Cause register.

- The timer interrupt signal, IP7, is directly readable as bit 15 of the Cause register.
- ★ Since the Int0 to 4 signals are input to bits 10 to 14 in the Cause register, direct referencing from the Cause register is possible.

IP1 and IP0 of the Cause register, which are described in **CHAPTER 7 EXCEPTION PROCESSING**, are software interrupts. There is no hardware mechanism for setting or clearing the software interrupts.

★

Figure 10-2. Hardware Interrupt Request Signals



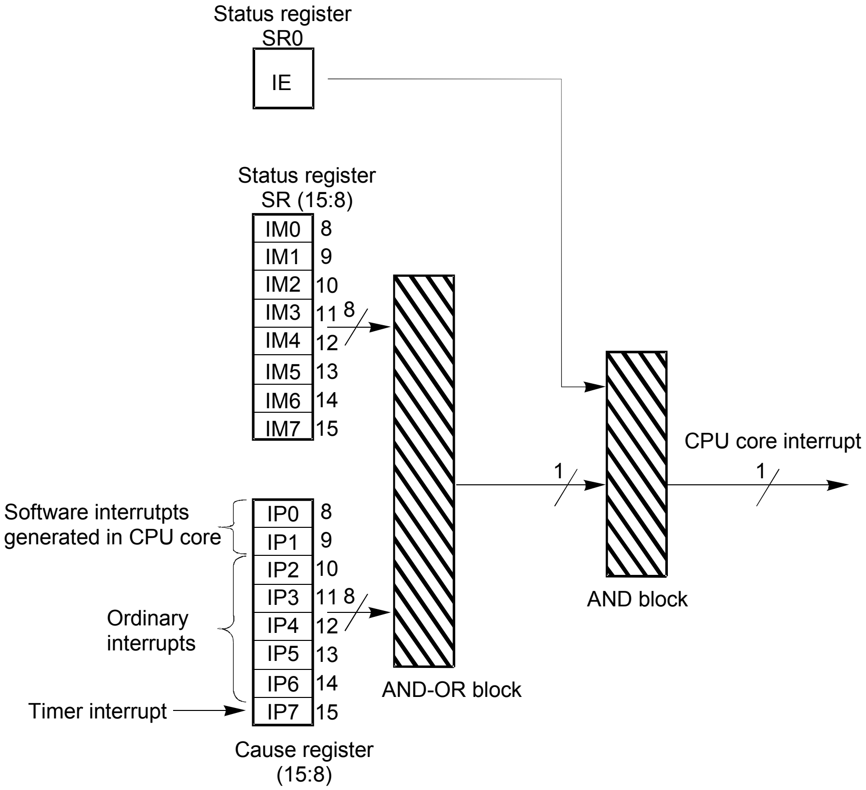
Remark Int4 does not occur in the VR4121.

10.5.2 Masking interrupt signals

Figure 10-3 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 10-3 to produce the CPU core interrupt signal. The EXL bit in the Status register also enables these interrupts.

Figure 10-3. 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 11 BCU (BUS CONTROL UNIT)

This chapter describes the BCU's operations and register settings.

11.1 General

In the VR4121, the BCU receives data that has passed via the VR4120 CPU core and the SysAD bus. The BCU also controls external agents via the system bus, such as an LCD controller, DRAM, (EDO type DRAM, Synchronous DRAM), ROM (Flash memory or masked ROM, Synchronous ROM), or PCMCIA controller, and it transmits and receives data with these external agents via the ADD bus and DATA bus.

11.2 Register Set

The BCU registers are listed below.

Table 11-1. BCU Registers

Address	R/W	Register Symbols	Function
0x0B00 0000	R/W	BCUCNTREG1	BCU Control Register 1
0x0B00 0002	R/W	BCUCNTREG2	BCU Control Register 2
0x0B00 0004	R/W	ROMSIZEREG	ROM Size Register
0x0B00 0006	R/W	RAMSIZEREG	DRAM Size Register
0x0B00 000A	R/W	BCUSPEEDREG	BCU Access Cycle Change Register
0x0B00 000C	R/W	BCUERRSTREG	BCU BUS ERROR Status Register
0x0B00 000E	R/W	BCURFCNTREG	BCU Refresh Control Register
0x0B00 0010	R	REVIDREG	Revision ID Register
0x0B00 0012	R/W	BCURFCOUNTREG	BCU Refresh Count Register
0x0B00 0014	R	CLKSPEEDREG	Clock Speed Register
0x0B00 0016	R/W	BCUCNTREG3	BCU Control Register 3
0x0B00 001A	R/W	SDRAMMODEREG	SDRAM Mode Register
0x0B00 001C	R/W	SROMMODEREG	SROM Mode Register
0x0B00 001E	R/W	SDRAMCNTREG	SDRAM Control Register
★ 0x0B00 0300	R/W	BCUTOUTCNTREG	BCU Timeout Control Register
★ 0x0B00 0302	R/W	BCUTOUTCOUNTREG	BCU Timeout Count Register

Each register is described in detail as follows.

11.2.1 BCUCNTREG1 (0x0B00 0000)

(1/2)

Bit	D15	D14	D13	D12	D11	D10	D9	D8
Name	ROM64	DRAM64	ISAM/LCD	PAGE128	RFU	PAGE ROM2	RFU	PAGE ROM0
R/W	R/W	R/W	R/W	R/W	R	R/W	R	R/W
RTCRST	0	**Note 1**	0	0	0	0	0	0
After reset	0	**Note 1**	0	0	0	0	0	0

★

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	RFU	ROMW EN2	RFU	ROMW EN0	LTOUTEN	RD64D	BUSH ERREN	RSTOUT
R/W	R	R/W	R	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15	ROM64	Sets the capacity of the ROM to be used 1: 64 Mbits 0: 32 Mbits
14	DRAM64	Sets the capacity of the DRAM to be used 1: 64 Mbits 0: 16 Mbits
13	ISAM/LCD	Assigns space from 0x0A00 0000 to 0x0AFF FFFF as the physical address space. 1: As ISA high-speed memory space 0: As LCD space
12	PAGE128	Sets the maximum burst access size for Page ROM. 1: 128 bits (16 bytes) 0: 64 bits (8 bytes)
11	RFU	Write 0 to this bit. 0 is returned after a read.
10	PAGEROM2	This function differs according to the setting of the SMODE(2:1) pin. • When SMODE(2:1) = 00 or 10 (SROM is not used) PageROM access is enabled for bank 3/2 (during 16-bit mode) or bank 3/2/1 (during 32-bit mode) of the ROM space 1: PageROM bus access 0: Ordinary ROM bus access • When SMODE(2:1) = 01 (SROM other than boot bank[Note 2] is used) PageROM access is enabled for bank 3 (during 16-bit mode) or bank 1 (during 32-bit mode) of the ROM space 1: PageROM bus access 0: Ordinary ROM bus access • When SMODE(2:1) = 11 (SROM is used for all banks) This bit setting has no significance.
9	RFU	Write 0 to this bit. 0 is returned after a read.

Notes 1. When using SDRAM: 1
When using EDO-type DRAM: 0

2. During 16-bit mode: Bank 3 (selected via ROMCS3#)
During 32-bit mode: Bank 1 (selected via ROMCS1#)

(2/2)

Bit	Name	Function
8	PAGEROM0	This function differs according to the setting of the SMODE(2:1) pin. • When SMODE(2:1) = 00 or 10 (SROM is not used PageROM access is enabled for banks 1 and 0 (during 16-bit mode) or bank 0 (during 32-bit mode) of the ROM space 1: PageROM bus access 0: Ordinary ROM bus access • When SMODE(2:1) = 01 (SROM other than boot bank[Note] is used) or 11 (SROM is used for all banks) This bit setting has no significance.
7	RFU	Write 0 to this bit. 0 is returned after a read.
6	ROMWEN2	This function differs according to the setting of the SMODE(2:1) pin. • When SMODE(2:1) = 00 or 10 (SROM is not used) A flash memory write cycle is enabled for the ROM space in banks 3 and 2 (during 16-bit mode) or in banks 3, 2, and 1 (during 32-bit mode) and a read-only bus cycle is issued for the flash memory register. 1: Enabled (the PAGEROM2 bit has no effect) 0: Prohibited • When SMODE(2:1) = 01 (SROM other than boot bank[Note] is used) A flash memory write cycle is enabled for the ROM space in bank 3 (during 16-bit mode) or in bank 1 (during 32-bit mode) and a read-only bus cycle is issued for the flash memory register. 1: Enabled (the PAGEROM2 bit has no effect) 0: Prohibited • When SMODE(2:1) = 11 (SROM is used for all banks) This bit setting has no significance.
5	RFU	Write 0 to this bit. 0 is returned after a read.
4	ROMWEN0	This function differs according to the setting of the SMODE(2:1) pin. • When SMODE(2:1) = 00 or 10 (SROM is not used) A flash memory write cycle is enabled for the ROM space in banks 1 and 0 (during 16-bit mode) or in bank 0 (during 32-bit mode) and a read-only bus cycle is issued for the flash memory register. 1: Enabled (the PAGEROM0 bit has no effect) 0: Prohibited • When SMODE(2:1) = 01 (SROM other than boot bank[Note] is used) or 11 (SROM is used for all banks) This bit setting has no significance.
★ 3	LTOUTEN	Timeout time change is permitted. (available from version 2.0) 1: Enabled 0: Disabled (VR4111-compatible mode). 2 refresh cycles of timeout detection time, which is the same as that of VR4111
★ 2	RD64D	ROM/DRAM capacity setting by RM64/DRAM64/EXT_ROM64/EXT_DRAM64. This bit is set to 1 when the capacity is set by ROMSIZEREG/RAMSIZEREG. 1: Disabled 0: Enabled
1	BUSHERREN	This is the bus timeout detection enable bit, which is used when a bus hold has been received. 1: Performs timeout detection when a bus hold has been received. 0: Does not perform timeout detection when a bus hold has been received.
0	RSTOUT	RSTOUT control bit 1: High level 0: Low level

Notes During 16-bit mode: Bank 3 (selected via ROMCS3#)
During 32-bit mode: Bank 1 (selected via ROMCS1#)

This register is used to set parameters such as the bus interface's bus cycle and memory type to be used.

When setting this register, BCUCNTREG3 (0x0B00 0016) must also be set.

★ There are two methods to set the capacity of ROM/DRAM in the VR4121.

- Setting by using the ROM64 and DRAM64 bits (VR4111-compatible)
  The ROM64 bit is used to set the capacity of all the ROM area in the 16-bit bus mode and ROM area other than the expansion space in the 32-bit bus mode.
  The DRAM64 bit is used to set the capacity of all the DRAM area in the 16-bit bus mode and DRAM area other than the expansion space in the 32-bit bus mode.
  When using these methods, set the RD64D bit to 0.

- Setting by using ROMSIZEREG (0x0B00 0004) and RAMSIZEREG (0x0B00 0006)
  Each bank capacity is set by using ROMSIZEREG and RAMSIZEREG. The capacity of expansion space ROM and the DRAM area can also be set by BCUCNTREG3.
  Using this method sets the RD64D bit to 1.

For a more detailed description of the allocation of banks in the ROM and DRAM areas, see **6.3.1 ROM address space** and **6.3.5 DRAM address space**.

The RSTOUT bit is used to set and display the RSTOUT pin's status. The RSTOUT pin's status changes at the following times.

★ - When bus timeout is detected
  If the defined refresh cycle cannot be executed (bus timeout) during Fullspeed mode, the RSTOUT pin becomes active after an interrupt request (BERRST) is sent to the CPU core. To detect the bus timeout during the bus hold cycle, the BUSHERREN bit in this register must be set to 1.

- When set by software
  The RSTOUT pin becomes active when the RSTOUT bit is set to "1" by software and it becomes inactive when the RSTOUT bit is set to "0". No interrupt request (BERRST) is sent to the CPU core when the RSTOUT pin's status is set by software.

- During power down mode
  When changing from Fullspeed mode to Hibernate mode, the RSTOUT pin enters the high impedance state. Since BCUCNTREG1 is initialized, the RSTOUT bit's value becomes "0". When the system recovers from Hibernate mode and resumes operation based on the ordinary clock, the RSTOUT pin becomes active again, then becomes inactive once the CPU's initialization processing begins. After a power-down, No interrupt request (BERRST) is sent to the CPU core when the RSTOUT pin's status becomes active. For details of Hibernate mode switching and initialization, see **CHAPTER 8 INITIALIZATION INTERFACE** and **CHAPTER 16 PMU (POWER MANAGEMENT UNIT)**.

11.2.2 BCUCNTREG2 (0x0B00 0002)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	GMODE
R/W	R	R	R	R	R	R	R	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:1	RFU	Write 0 to these bits. 0 is returned after a read.
0	GMODE	This is the access data control bit for LCD space. 1: Do not invert the access data for LCD space 0: Invert the access data for LCD space

This register is used to specify whether data is inverted (translated to 2's complement) or not when accessing the LCD space.

The LCD space is accessed when the ISAM/LCD bit of BCUCNTREG1 is 0. When it is 1, this address space is used as the ISA high-speed memory space. In this case, the contents of the BCUCNTREG2 register are invalid, and inversion of access data is not performed.

11.2.3 ROMSIZEREG (0x0B00 0004)

(1/2)

Bit	15	14	13	12	11	10	9	8
Name	RFU	SIZE32	SIZE31	SIZE30	RFU	SIZE22	SIZE21	SIZE20
R/W	R	R/W	R/W	R/W	R	R/W	R/W	R/W
RTCRST	0	**Note 1**	**Note 1**	**Note 1**	0	**Note 1**	**Note 1**	**Note 1**
★ After reset	0	**Note 1**	**Note 1**	**Note 1**	0	**Note 1**	**Note 1**	**Note 1**

Bit	7	6	5	4	3	2	1	0
Name	RFU	SIZE12	SIZE11	SIZE10	RFU	SIZE02	SIZE01	SIZE00
R/W	R	R/W	R/W	R/W	R	R/W	R/W	R/W
RTCRST	0	**Note 1**	**Note 1**	**Note 1**	0	**Note 1**	**Note 1**	**Note 1**
★ After reset	0	**Note 2**	**Note 2**	**Note 2**	0	**Note 2**	**Note 2**	**Note 2**

<table>
<tr><th>Bit</th><th>Name</th><th>Function</th></tr>
<tr><td>15</td><td>RFU</td><td>Write 0 to this bit. 0 is returned after a read.</td></tr>
<tr><td>14:12</td><td>SIZE3(2:0)</td><td>Sets the ROM capacity of bank 3 (when in 16-bit bus mode) or bank 1 (when in 32-bit bus mode).
<table>
<tr><th>SIZE3(2:0)</th><th colspan="2">16-bit mode (Mbytes)</th><th colspan="2">32-bit mode (Mbytes)</th></tr>
<tr><th></th><th>SROM</th><th>ROM</th><th>SROM</th><th>ROM</th></tr>
<tr><td>111</td><td>RFU</td><td>RFU</td><td>RFU</td><td>RFU</td></tr>
<tr><td>110</td><td>RFU</td><td>RFU</td><td>RFU</td><td>RFU</td></tr>
<tr><td>101</td><td>RFU</td><td>RFU</td><td>RFU</td><td>RFU</td></tr>
<tr><td>100</td><td>RFU</td><td>RFU</td><td>RFU</td><td>32</td></tr>
<tr><td>011</td><td>RFU</td><td>16</td><td>RFU</td><td>16</td></tr>
<tr><td>010</td><td>RFU</td><td>8</td><td>8</td><td>8</td></tr>
<tr><td>001</td><td>4</td><td>4</td><td>4</td><td>RFU</td></tr>
<tr><td>000</td><td>RFU</td><td>RFU</td><td>RFU</td><td>RFU</td></tr>
</table>
</td></tr>
<tr><td>11</td><td>RFU</td><td>Write 0 to this bit. 0 is returned after a read.</td></tr>
</table>

Notes 1. The following value is applied depending on the data bus's bit width (n = 0 to 3).
During 16-bit mode: SIZEn(2:0) = 001
During 32-bit mode: SIZEn(2:0) = 010

2. The following value is applied depending on the data bus's bit width (n = 0, 1).
During 16-bit mode: SIZEn(2:0) = 001
During 32-bit mode, EXT_ROM64 = 0: SIZEn(2:0) = 010
During 32-bit mode, EXT_ROM64 = 1: SIZEn(2:0) = 011

(2/2)

Bit	Name	Function				
10:8	SIZE2(2:0)	Sets the ROM capacity of bank 2 (when in 16-bit bus mode) or bank 0 (when in 32-bit bus mode).				
		SIZE2(2:0)	16-bit mode (Mbytes)		32-bit mode (Mbytes)	
			SROM	RON	SROM	ROM
		111	RFU	RFU	RFU	RFU
		110	RFU	RFU	RFU	RFU
		101	RFU	RFU	RFU	RFU
		100	RFU	RFU	RFU	32
		011	RFU	16	RFU	16
		010	RFU	8	8	8
		001	4	4	4	RFU
		000	RFU	RFU	RFU	RFU
7	RFU	Write 0 to this bit. 0 is returned after a read.				
6:4	SIZE1(2:0)	Sets the ROM capacity of bank 1 (when in 16-bit bus mode) or bank 3 (when in 32-bit bus mode).				
		SIZE1(2:0)	16-bit mode (Mbytes)		32-bit mode (Mbytes)	
			SROM	ROM	SROM	ROM
		111	RFU	RFU	RFU	RFU
		110	RFU	RFU	RFU	RFU
		101	RFU	RFU	RFU	RFU
		100	RFU	RFU	RFU	32
		011	RFU	16	RFU	16
		010	RFU	8	8	8
		001	4	4	4	RFU
		000	RFU	RFU	RFU	RFU
3	RFU	Write 0 to this bit. 0 is returned after a read.				
2:0	SIZE0(2:0)	Sets the ROM capacity of bank 0 (when in 16-bit bus mode) or bank 2 (when in 32-bit bus mode).				
		SIZE0(2:0)	16-bit mode (Mbytes)		32-bit mode (Mbytes)	
			SROM	RON	SROM	ROM
		111	RFU	RFU	RFU	RFU
		110	RFU	RFU	RFU	RFU
		101	RFU	RFU	RFU	RFU
		100	RFU	RFU	RFU	32
		011	RFU	16	RFU	16
		010	RFU	8	8	8
		001	4	4	4	RFU
		000	RFU	RFU	RFU	RFU

This register is used to set the capacity of each bank in the ROM space.

When in 16-bit mode, each bank is connected to one ROM or SROM device. When in 32-bit mode, each bank is connected to one ROM or SROM device, or two SROM devices.

Refer to **6.3.1 ROM address space** for details of the ROM space bank allocation.

★ ROM space capacity can also be set by using the ROM64 bit in BCUCNTREG1 (other than expansion space) or the EXT_ROM64 bit in BCUCNTREG3 (expansion space). Refer to the relevant descriptions on registers and **6.3.1 ROM address space**.

11.2.4 RAMSIZEREG (0x0B00 0006)

(1/2)

Bit	15	14	13	12	11	10	9	8
Name	RFU	SIZE32	SIZE31	SIZE30	RFU	SIZE22	SIZE21	SIZE20
R/W	R	R/W	R/W	R/W	R	R/W	R/W	R/W
RTCRST	0	**Note 1**	**Note 1**	**Note 1**	0	**Note 1**	**Note 1**	**Note 1**
★ After reset	0	**Note 2**	**Note 2**	**Note 2**	0	**Note 2**	**Note 2**	**Note 2**

Bit	7	6	5	4	3	2	1	0
Name	RFU	SIZE12	SIZE11	SIZE10	RFU	SIZE02	SIZE01	SIZE00
R/W	R	R/W	R/W	R/W	R	R/W	R/W	R/W
RTCRST	0	**Note 1**	**Note 1**	**Note 1**	0	**Note 1**	**Note 1**	**Note 1**
★ After reset	0	**Note 1**	**Note 1**	**Note 1**	0	**Note 1**	**Note 1**	**Note 1**

<table>
<tr><th>Bit</th><th>Name</th><th>Function</th></tr>
<tr><td>15</td><td>RFU</td><td>Write 0 to this bit. 0 is returned after a read.</td></tr>
<tr><td>14:12</td><td>SIZE3(2:0)</td><td>Sets the RAM capacity of bank 3 (when in 16-bit bus mode) or bank 3 (when in 32-bit bus mode).
<table>
<tr><th>SIZE3(2:0)</th><th colspan="2">16-bit mode (Mbytes)</th><th colspan="2">32-bit mode (Mbytes)</th></tr>
<tr><th></th><th>SDRAM</th><th>EDO DRAM</th><th>SDRAM</th><th>EDO DRAM</th></tr>
<tr><td>111</td><td>RFU</td><td>RFU</td><td>RFU</td><td>RFU</td></tr>
<tr><td>110</td><td>RFU</td><td>RFU</td><td>RFU</td><td>RFU</td></tr>
<tr><td>101</td><td>RFU</td><td>RFU</td><td>RFU</td><td>RFU</td></tr>
<tr><td>100</td><td>RFU</td><td>RFU</td><td>32</td><td>RFU</td></tr>
<tr><td>011</td><td>16</td><td>RFU</td><td>16</td><td>16</td></tr>
<tr><td>010</td><td>8</td><td>8</td><td>8</td><td>RFU</td></tr>
<tr><td>001</td><td>RFU</td><td>RFU</td><td>RFU</td><td>4</td></tr>
<tr><td>000</td><td>RFU</td><td>2</td><td>RFU</td><td>RFU</td></tr>
</table></td></tr>
<tr><td>11</td><td>RFU</td><td>Write 0 to this bit. 0 is returned after a read.</td></tr>
</table>

Notes 1. The following value is applied depending on the data bus's bit width and the type of RAM (n = 0 to 3).
During 16-bit mode, when using SDRAM: SIZEn(2:0) = 010
During 16-bit mode, when using EDO DRAM: SIZEn(2:0) = 000
★ During 32-bit mode, when using SDRAM: SIZEn(2:0) = 011
During 32-bit mode, when using EDO DRAM: SIZEn(2:0) = 001

2. The following value is applied depending on the data bus's bit width and the type of RAM (n = 2, 3).
During 16-bit mode, when using SDRAM: SIZEn(2:0) = 010
During 16-bit mode, when using EDO DRAM: SIZEn(2:0) = 000
During 32-bit mode, when using SDRAM: SIZEn(2:0) = 011
During 32-bit mode, when using EXT_DRAM64 = 0, EDO_DRAM: SIZEn(2:0) = 001
During 32-bit mode, when using EXT_DRAM64 = 1, EDO_DRAM: SIZEn(2:0) = 011

(2/2)

<table>
<tr><th>Bit</th><th>Name</th><th>Function</th></tr>
<tr><td>10:8</td><td>SIZE2(2:0)</td><td>Sets the RAM capacity of bank 2 (when in 16-bit bus mode) or bank 2 (when in 32-bit bus mode).
<table>
<tr><th>SIZE2(2:0)</th><th colspan="2">16-bit mode (Mbytes)</th><th colspan="2">32-bit mode (Mbytes)</th></tr>
<tr><th></th><th>SDRAM</th><th>EDO DRAM</th><th>SDRAM</th><th>EDO DRAM</th></tr>
<tr><td>111</td><td>RFU</td><td>RFU</td><td>RFU</td><td>RFU</td></tr>
<tr><td>110</td><td>RFU</td><td>RFU</td><td>RFU</td><td>RFU</td></tr>
<tr><td>101</td><td>RFU</td><td>RFU</td><td>RFU</td><td>RFU</td></tr>
<tr><td>100</td><td>RFU</td><td>RFU</td><td>32</td><td>RFU</td></tr>
<tr><td>011</td><td>16</td><td>RFU</td><td>16</td><td>16</td></tr>
<tr><td>010</td><td>8</td><td>8</td><td>8</td><td>RFU</td></tr>
<tr><td>001</td><td>RFU</td><td>RFU</td><td>RFU</td><td>4</td></tr>
<tr><td>000</td><td>RFU</td><td>2</td><td>RFU</td><td>RFU</td></tr>
</table></td></tr>
<tr><td>7</td><td>RFU</td><td>Write 0 to this bit. 0 is returned after a read.</td></tr>
<tr><td>6:4</td><td>SIZE1(2:0)</td><td>Sets the RAM capacity of bank 1 (when in 16-bit bus mode) or bank 1 (when in 32-bit bus mode).
<table>
<tr><th>SIZE1(2:0)</th><th colspan="2">16-bit mode (Mbytes)</th><th colspan="2">32-bit mode (Mbytes)</th></tr>
<tr><th></th><th>SDRAM</th><th>EDO</th><th>SDRAM</th><th>EDO</th></tr>
<tr><td>111</td><td>RFU</td><td>RFU</td><td>RFU</td><td>RFU</td></tr>
<tr><td>110</td><td>RFU</td><td>RFU</td><td>RFU</td><td>RFU</td></tr>
<tr><td>101</td><td>RFU</td><td>RFU</td><td>RFU</td><td>RFU</td></tr>
<tr><td>100</td><td>RFU</td><td>RFU</td><td>32</td><td>RFU</td></tr>
<tr><td>011</td><td>16</td><td>RFU</td><td>16</td><td>16</td></tr>
<tr><td>010</td><td>8</td><td>8</td><td>8</td><td>RFU</td></tr>
<tr><td>001</td><td>RFU</td><td>RFU</td><td>RFU</td><td>4</td></tr>
<tr><td>000</td><td>RFU</td><td>2</td><td>RFU</td><td>RFU</td></tr>
</table></td></tr>
<tr><td>3</td><td>RFU</td><td>Write 0 to this bit. 0 is returned after a read.</td></tr>
<tr><td>2:0</td><td>SIZE0(2:0)</td><td>Sets the RAM capacity of bank 0 (when in 16-bit bus mode) or bank 0 (when in 32-bit bus mode).
<table>
<tr><th>SIZE0(2:0)</th><th colspan="2">16-bit mode (Mbytes)</th><th colspan="2">32-bit mode (Mbytes)</th></tr>
<tr><th></th><th>SDRAM</th><th>EDO</th><th>SDRAM</th><th>EDO</th></tr>
<tr><td>111</td><td>RFU</td><td>RFU</td><td>RFU</td><td>RFU</td></tr>
<tr><td>110</td><td>RFU</td><td>RFU</td><td>RFU</td><td>RFU</td></tr>
<tr><td>101</td><td>RFU</td><td>RFU</td><td>RFU</td><td>RFU</td></tr>
<tr><td>100</td><td>RFU</td><td>RFU</td><td>32</td><td>RFU</td></tr>
<tr><td>011</td><td>16</td><td>RFU</td><td>16</td><td>16</td></tr>
<tr><td>010</td><td>8</td><td>8</td><td>8</td><td>RFU</td></tr>
<tr><td>001</td><td>RFU</td><td>RFU</td><td>RFU</td><td>4</td></tr>
<tr><td>000</td><td>RFU</td><td>2</td><td>RFU</td><td>RFU</td></tr>
</table></td></tr>
</table>

This register is used to set the capacity of each bank in the DRAM space.

Refer to **6.3.5 DRAM address space** for details of the DRAM space bank allocation.

★ DRAM space capacity can also be set by using the DRAM64 bit in BCUCNTERG1 (other than expansion space) or the EXT_DRAM64 bit (expansion space). Refer to the relevant descriptions on registers and **6.3.5 DRAM address space**.

11.2.5 BCUSPEEDREG (0x0B00 000A)

(1/2)

Bit	15	14	13	12	11	10	9	8
Name	MINPLS1	MINPLS0	WPROM1	WPROM0	RFU	WLCD/M2	WLCD/M1	WLCD/M0
R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	WISAA2	WISAA1	WISAA0	RFU	WROMA2	WROMA1	WROMA0
R/W	R	R/W	R/W	R/W	R	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:14	MINPLS(1:0)	Minimum low-level pulse width of MEMR#/MEMW# pin when accessing high-speed system memory space (ISAM/LCD = 1). 11: 4 Tisa 10: 3 Tisa 01: 2 Tisa 00: 1 Tisa
13:12	WPROM(1:0)	Page ROM access speed 11: RFU 10: 1 TClock 01: 2 TClock 00: 3 TClock
11	RFU	Write 0 to this bit. 0 is returned after a read.
★ 10:8	WLCD/M(2:0)	This is the access speed to physical address range 0x0A00 0000 to 0x0AFF FFFF. During LCD interface mode (when ISAM/LCD = 0), VR4111 compatible mode is set when the WLCD/M2 bit's value is "0" and access cycle reduction mode[Note] is set when its value is "1". Tlcd (ISAM/LCD=0) Tisa (ISAM/LCD=1) 111: 2 TClock 1 TClock 110: 3 TClock 2 TClock 101: 4 TClock 3 TClock 100: 5 TClock 4 TClock 011: 2 TClock 5 TClock 010: 4 TClock 6 TClock 001: 6 TClock 7 TClock 000: 8 TClock 8 TClock
7	RFU	Write 0 to this bit. 0 is returned after a read.

Note See **11.5.3 LCD interface** for the details of the access cycle reduction mode.

(2/2)

Bit	Name	Function
★ 6:4	WISAA(2:0)	Access speed for ISA memory space and ISA/IO space 111: RFU 110: RFU 101: 3 TClock[Note] 100: 4 TClock[Note] 011: 5 TClock 010: 6 TClock 001: 7 TClock 000: 8 TClock
3	RFU	Write 0 to this bit. 0 is returned after a read.
2:0	WROMA(2:0)	ROM access speed 111: 2 TClock 110: 3 TClock 101: 4 TClock 100: 5 TClock 011: 6 TClock 010: 7 TClock 001: 8 TClock 000: 9 TClock

Note When the WISAA(2:0) bits are set to 101 or 100, the AC characteristics between BUSCLK and the system bus interface signals (ADD25/SCLK, ADD(24:0), SHB#, MEMR#, MEMW#, IOR#, and IOW#) are not guaranteed.

This register is used to set the access speed for the LCD, system bus, pageROM, and ROM.

The lowest speed is set when "0" is set to all of the following bits: WLCD/M(2:0), WPROM(1:0), WISAA(2:0), and WROMA(2:0).

★ The value set to WPROM(1:0) is valid only when "1" has been set to the PAGEROM bit in BCUCNTREG1 and the pageROM access is enabled.

11.2.6 BCUERRSTREG (0x0B00 000C)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	BERRST
R/W	R	R	R	R	R	R	R	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:1	RFU	Write 0 to these bits. 0 is returned after a read.
0	BERRST	Bus error status. Clear to 0 when 1 is written. 1: Bus error 0: Normal

This register is used to indicate when a bus error interrupt request has occurred.

The bus error interrupt can be cleared by setting BERRST bit to 1.

For bus error generation causes, see **11.4.7 Illegal access notification**.

11.2.7 BCURFCNTREG (0x0B00 000E)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	BRF13	BRF12	BRF11	BRF10	BRF9	BRF8
R/W	R	R	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	1	0
★ After reset	0	0	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	7	6	5	4	3	2	1	0
Name	BRF7	BRF6	BRF5	BRF4	BRF3	BRF2	BRF1	BRF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
★ After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15:14	RFU	Write 0 to these bits. 0 is returned after a read.
13:0	BRF(13:0)	Number of DRAM refresh cycles (with TClock cycle).

Note Value before reset is retained.

This register is used to set the number of DRAM refresh cycles.

The refresh interval can be obtained by calculating the following expression:

Refresh interval = BRF(13:0) $\times$ TClock

Therefore, the setting value should be obtained by calculating the number of used DRAM refresh cycles (eg. 4,096 cycles/128 ms in the μPD42S16165) and the bus access cycles (each address space/bus hold cycle). Refer to CLKSPEEDREG (0x0B00 0014) for the TClock frequency.

★ Even when using SDRAM, use the TClock number rather than the VTClock number to set the refresh interval.

If a bus timeout occurs, one DRAM refresh cycle is lost. It occurs when the ready signal and HLDRQ# signal do not become high level in a certain period during the system bus I/O MEM, LCD/high-speed system bus, or bus hold cycle.

11.2.8 REVIDREG (0x0B00 0010)

Bit	15	14	13	12	11	10	9	8
Name	RID3	RID2	RID1	RID0	MJREV3	MJREV2	MJREV1	MJREV0
R/W	R	R	R	R	R	R	R	R
★ RTCRST	0	0	1	1	**Note**	**Note**	**Note**	**Note**
★ After reset	0	0	1	1	**Note**	**Note**	**Note**	**Note**

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	MNREV3	MNREV2	MNREV1	MNREV0
R/W	R	R	R	R	R	R	R	R
★ RTCRST	0	0	0	0	**Note**	**Note**	**Note**	**Note**
After reset	0	0	0	0	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15:12	RID(3:0)	This is the processor revision ID. 0x03 indicates the VR4121.
11:8	MJREV(3:0)	Major revision number
7:4	RFU	Write 0 to these bits. 0 is returned after a read.
3:0	MNREV(3:0)	Minor revision number

Note Varies depending on the delivery date.

This register is used to indicate revisions of the VR4121's peripheral units.

The revision number is stored as a value in the form *y.x*, where *y* is a major revision number and *x* is a minor revision number.

Even if the CPU core or the peripheral unit has been changed, there is no guarantee that REVIDREG will bee reflected, or that changes to the revision number necessarily reflect real CPU core's and units' changes. For this reason, software should not rely on the revision number in REVIDREG to characterize the units.

11.2.9 BCURFCOUNTREG (0x0B00 0012)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	BRFC13	BRFC12	BRFC11	BRFC10	BRFC9	BRFC8
R/W	R	R	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
★ After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	BRFC7	BRFC6	BRFC5	BRFC4	BRFC3	BRFC2	BRFC1	BRFC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	1
★ After reset	0	0	0	0	0	0	0	1

Bit	Name	Function
15:14	RFU	Write 0 to these bits. 0 is returned after a read.
13:0	BRFC(13:0)	Number of the current DRAM refresh cycle.

This register is used to indicate the current refresh cycle count value.

When the value of this register is 0x0000, a refresh cycle request is generated and the BCURFCNTREG value is set. The counter operates irrespective of refresh cycle generation.

This register is decremented in sync with TClock. Accordingly, the refresh cycle can be calculated based on the value set to the CLKPEEDREG register's DIVT(3:0) bit.

Even after the refresh cycle request is generated, if no refresh cycle is generated because of other bus cycles (system bus I/O MEM, LCD/high-speed system bus, or bus hold) and if this register is 0x0000, bus timeout occurs (during bus hold cycle, this depends on the setting of BCUCNTREG1's bit 1).

11.2.10 CLKSPEEDREG (0x0B00 0014)

Bit	15	14	13	12	11	10	9	8
Name	DIVT3	DIVT2	DIVT1	DIVT0	DIVVT3	DIVVT2	DIVVT1	DIVVT0
R/W	R	R	R	R	R	R	R	R
RTCRST	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined
After reset	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	CLKSP4	CLKSP3	CLKSP2	CLKSP1	CLKSP0
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	Undefined	Undefined	Undefined	Undefined	Undefined
After reset	0	0	0	Undefined	Undefined	Undefined	Undefined	Undefined

Bit	Name	Function
15:12	DIVT(3:0)	Division rate setting for peripheral unit's operating clock (TClock), relative to the CPU core's operating clock (PClock). 0110: Divide by 6 0101: Divide by 5 0100: Divide by 4 0011: Divide by 3 Other: RFU
11:8	DIVVT(3:0)	Division rate setting for SDRAM/SROM unit's operating clock (VTClock), relative to the CPU core's operating clock (PClock). 1010: Divide by 2.5 1001: Divide by 1.5 0110: Divide by 6 0101: Divide by 5 0100: Divide by 4 0011: Divide by 3 0010: Divide by 2 0001: Divide by 1 Other: RFU
7:5	RFU	Write 0 to these bits. 0 is returned after a read.
4:0	CLKSP(4:0)	Used to calculate the operating clock (PClock) of the CPU core.

The contents of this register indicate the division rate of the operating clock of each operation.
The number of each operation clock frequency can be obtained by calculating the following expression.

PClock = (18.432 MHz/CLKSP(4:0)) × 64 [MHz]
TClock = PClock/DIVT(3:0) [MHz]
VTClock = PClock/DIVVT(3:0) [MHz]

11.2.11 BCUCNTREG3 (0x0B00 0016)

Bit	15	14	13	12	11	10	9	8
Name	EXT_ ROM64	EXT_ DRAM64	EXT_ ROMCS3	EXT_ ROMCS2	EXT_ MEM	RFU	RFU	RFU
R/W	R/W	R/W	R/W	R/W	R/W	R	R	R
RTCRST	0	**Note 1**	0	0	0	0	0	0
★ After reset	**Note 2**	**Note 2**	**Note 2**	**Note 2**	**Note 2**	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	LCD32/ ISA32	RFU	RFU	RFU	RFU	BSEL	LCDSEL1	LCDSEL0
R/W	R/W	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
★ After reset	**Note 2**	0	0	0	0	**Note 2**	**Note 2**	**Note 2**

Bit	Name	Function
15	EXT_ROM64	Sets the capacity of the expansion ROM to be used. 1: 64 Mbits 0: 32 Mbits
14	EXT_DRAM64	Sets the capacity of the expansion DRAM to be used. 1: 64 Mbits 0: 16 Mbits
13:12	EXT_ROMCS(3:2)	Assigns space of banks 3 and 2 (32-bit mode). 11: Bank 3 = ROM, bank 2 = ROM 10: Bank 3 = ROM, bank 2 = DRAM 01: RFU 00: Bank 3 = DRAM, bank 2 = DRAM
11	EXT_MEM	Enables/Disables an access to expansion memory (ROM/DRAM). 1: Enable 0: Disable
10:8	RFU	Write 0 to these bits. 0 is returned after a read.
7	LCD32/ISA32	Sets the data bus size of LCD or high-speed system bus space (32-bit mode). 1: 32 bits 0: 16 bits
6:3	RFU	Write 0 to these bits. 0 is returned after a read.
2	BSEL	BUSCLK output setting 1: Output TClock 0: Output clock that is 1/4 of TClock
1	LCDSEL1	Device is allocated to physical address range 0x0AFF FFFF to 0x0A80 0000. 1: External to load reduction buffer 0: Internal to load reduction buffer
0	LCDSEL0	Device is allocated to physical address range 0x0A7F FFFF to 0x0A00 0000. 1: External to load reduction buffer 0: Internal to load reduction buffer

★ **Notes 1.** When using SDRAM: 1
Others: 0

2. Value before reset is retained.

This register is used to control use of the expanded memory space.

Bits 15:11 and 7 can be set only during 32-bit data bus mode (when DBUS32 = 1).

When using SDRAM and SROM with the VR4121, we recommend inserting a load reduction buffer to the DATA bus and ADD bus. The buffer's direction can be controlled by setting a value in LCDSEL(1:0) to indicate whether or not a memory device has been connected to the inserted buffer.

11.2.12 SDRAMMODEREG (0x0B00 001A)

Bit	15	14	13	12	11	10	9	8
Name	SCLK	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R/W	R	R	R	R	R	R	R
RTCRST	1	0	0	0	0	0	0	0
After reset	**Note**	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	LTMODE2	LTMODE1	LTMODE0	WT	BL2	BL1	BL0
R/W	R	R/W	R/W	R/W	R	R	R	R
RTCRST	0	0	1	1	1	0	0	1
After reset	0	**Note**	**Note**	**Note**	1	0	0	1

Bit	Name	Function
15	SCLK	SCLK output setting for ADD25/SCLK pin 1: SCLK is always output. 0: SCLK is output only when SDRAM or SROM is used.
14:7	RFU	Write 0 to these bits. 0 is returned after a read.
6:4	LTMODE(2:0)	SDRAM's CAS/latency setting 111: RFU 110: RFU 101: RFU 100: RFU 011: 3 010: 2 001: RFU 000: RFU
3	WT	Address ordering indication 1: Interleave 0: RFU
2:0	BL(2:0)	Burst length indication 111: RFU : 010: RFU 001: 2 000: RFU

Note Value before reset is retained.

This register sets and indicates the operation mode when using SDRAM or SROM.

If neither SDRAM nor SROM is being used, this register's settings have no significance.

11.2.13 SROMMODEREG (0x0B00 001C)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RL1	CL2	CL1	CL0	BT	BL1	BL0
R/W	R	R/W	R/W	R/W	R/W	R	R	R
RTCRST	0	1	1	0	1	1	0	1
After reset	0	**Note**	**Note**	**Note**	**Note**	1	0	1

Bit	Name	Function
15:7	RFU	Write 0 to these bits. 0 is returned after a read.
★ 6	RL1	SROM's RAS/latency setting 1: 2 0: RFU
5:3	CL(2:0)	SROM's CAS/latency setting 111: RFU 110: RFU 101: 6 100: 5 011: 4 010: 3 001: RFU 000: RFU
2	BT	Address ordering indication 1: Interleave 0: RFU
1:0	BL(1:0)	Burst length indication 11: RFU 10: RFU 01: 4 00: RFU

Note Value before reset is retained.

This register sets and indicates the operation mode when using SROM.
If SROM is not being used, this register's settings have no significance.

11.2.14 SDRAMCNTREG (0x0B00 001E)

(1/2)

★ Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	TRC3	TRC2	TRC1	TRC0
R/W	R	R	R	R	R/W	R/W	R/W	R/W
RTCRST	0	0	**Note 1**	**Note 2**	1	0	0	1
After reset	0	0	**Note 1**	**Note 2**	Undefined	Undefined	Undefined	Undefined

Bit	7	6	5	4	3	2	1	0
Name	RFU	TDAL2	TDAL1	TDAL0	RFU	TRCD2	TRCD1	TRCD0
R/W	R	R/W	R/W	R/W	R	R/W	R/W	R/W
RTCRST	0	1	0	0	0	1	0	0
After reset	0	Undefined	Undefined	Undefined	0	Undefined	Undefined	Undefined

Bit	Name	Function
15:14	RFU	Write 0 to these bits. 0 is returned after a read.
★ 13	SMODE2	Setting of the SMODE2 pin after reset (**2.2.14 Initial setting signals**).
★ 12	SMODE1	Setting of the SMODE1 pin after reset (**2.2.14 Initial setting signals**).
11:8	TRC(3:0)	This sets the minimum interval between bank active and bank active/self refresh/CBR refresh, between self refresh and bank active/self refresh/CBR refresh, and between CBR refresh and bank active/self refresh/CBR refresh. 1111: RFU : 1010: RFU 1001: 9 VTClock 1000: 8 VTClock 0111: 7 VTClock 0110: 6 VTClock 0101: 5 VTClock 0100: 4 VTClock 0011: 3 VTClock 0010: RFU ★ 0001: RFU 0000: RFU
7	RFU	Write 0 to this bit. 0 is returned after a read.

Notes 1. The setting of the SMODE2 pin

2. The setting of the SMODE1 pin

(2/2)

Bit	Name	Function
6:4	TDAL(2:0)	This sets the minimum interval between the first clock following the last data output in response to a write command with auto precharge and the start of bank active status. 111: RFU 110: RFU 101: RFU 100: 4 VTClock 011: 3 VTClock 010: 2 VTClock 001: RFU 000: RFU
3	RFU	Write 0 to this bit. 0 is returned after a read.
2:0	TRCD(2:0)	This sets the minimum interval between bank active and read/read with auto precharge/write/write with auto precharge. 111: RFU 110: RFU 101: RFU 100: 4 VTClock 011: 3 VTClock 010: 2 VTClock 001: RFU 000: RFU

★ Set the operation mode for the SDRAM being used in this register. When the SDRAM is not used, this register setting is ignored.

Set the value of TRC(3:0) according to the TRC/TRC1 setting for the SDRAM being used.

Set the value of TDAL(2:0) according to the TDAL setting for the SDRAM being used.

Set the value of TRCD(2:0) according to the TRC/TRAS setting for the SDRAM being used.

★

11.2.15 BCUTOUTCNTREG (0x0B00 0300)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	BCUTOUT CNT12	BCUTOUT CNT11	BCUTOUT CNT10	BCUTOUT CNT9	BCUTOUT CNT8
R/W	R	R	R	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	1	0	0	0	0
After reset	0	0	0	1	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	BCUTOUT CNT7	BCUTOUT CNT6	BCUTOUT CNT5	BCUTOUT CNT4	BCUTOUT CNT3	BCUTOUT CNT2	BCUTOUT CNT1	BCUTOUT CNT0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:13	RFU	Write 0 to these bits. 0 is returned after a read.
12:0	BCUTOUTCNT(12:0)	Setting of timeout detection time 0x1FFF: 8,191 refresh cycles 0x1FFE: 8,190 refresh cycles : 0x0001: 1 refresh cycle 0x0000: Timeout detection not performed

This register sets the timeout detection time.

The timeout is detected when a refresh is not issued even after queuing a refresh request up to the setup value.

The setting of this register is valid when the LTOUTEN bit in BCUCNTREG1 is set to 1. If the register is set to 0, the mode enters the VR4111 compatible mode, resulting in timeout detection after 2 refresh cycles.

★ 11.2.16 BCUTOUTCOUNTREG (0x0B00 0302)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	BCUTOUT COUNT13	BCUTOUT COUNT12	BCUTOUT COUNT11	BCUTOUT COUNT10	BCUTOUT COUNT9	BCUTOUT COUNT8
R/W	R	R	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	BCUTOUT COUNT7	BCUTOUT COUNT6	BCUTOUT COUNT5	BCUTOUT COUNT4	BCUTOUT COUNT3	BCUTOUT COUNT2	BCUTOUT COUNT1	BCUTOUT COUNT0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:14	RFU	Write 0 to these bits. 0 is returned after a read.
13:0	BCUTOUTCOUNT(13:0)	Indication of number of queued refresh requests.

This register indicates the number of queued refresh requests. The VR4121 issues all queued refresh requests under the following conditions.

- After the completion of other bus cycles and bus holds
- Before issuing a self-refresh in the Suspend and Hibernate modes

11.3 Connection of Address Pins

In the VR4121, the physical address output by the CPU core is supplied to an external device via the ADD bus. Table 11-2 shows the correspondence of bits in the physical address output to the ADD bus to the external device's address bits.

★ **Table 11-2. Address Bit Correspondence between ADD Bus and External Device**

ADD Pin Device	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ROM, LCD, ISA EDO DRAM	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
EDO DRAM (row)	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1
EDO DRAM (column) DATA(15:0)	25	24	23	22	21	20	19	20	19	08	07	06	05	04	03	02	01	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1
EDO DRAM (column) DATA(31:0)	25	24	23	22	21	20	19	20	19	08	07	06	05	04	03	02	21	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1
ROM, LCD, ISA SDRAM	Note 2	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SDRAM/SROM (row)	Note 2	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1
SDRAM/SROM (column)	Note 2	24	23	22	21	Note 3	24	23	09	08	07	06	05	04	03	02	01	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1	Note 1

Notes 1. Undefined (1 or 0)

2. SCLK

3. CMD

11.3.1 Connection to DRAM

Table 11-3 shows the connection example between the VR4121 and DRAM.

The data of row address (RAS signal) and column address (CAS signal) that correspond to RAS signal (MRAS(3:0)#) and CAS signal (UUCAS#, ULCAS#, UCAS#, and LCAS#), respectively, are output to DRAM (multiplexed-addressing supported), via the VR4121's ADD pin (ADD(23:9)).

See the table below for the physical address corresponding to each address.

Table 11-3. Example of DRAM Connection and Address Output from VR4121 (1/2)

(a) In 16-bit data bus mode (DBUS32 = 0)

DRAM Address Pin	16-Mbit DRAM (1 Mbit × 16)			64-Mbit DRAM (4 Mbits × 16)		
	ADD pin	Row	Column	ADD pin	Row	Column
A12/NC[Note 1]				ADD20	adr20	
A11/NC[Note 2]	ADD20	adr20		ADD22	adr22	
A10/NC[Note 2]	ADD19	adr19		ADD21	adr21	
A9	ADD18	adr18	adr20	ADD18	adr18	adr20
A8	ADD17	adr17	adr19	ADD17	adr17	adr19
A7	ADD16	adr16	adr8	ADD16	adr16	adr8
A6	ADD15	adr15	adr7	ADD15	adr15	adr7
A5	ADD14	adr14	adr6	ADD14	adr14	adr6
A4	ADD13	adr13	adr5	ADD13	adr13	adr5
A3	ADD12	adr12	adr4	ADD12	adr12	adr4
A2	ADD11	adr11	adr3	ADD11	adr11	adr3
A1	ADD10	adr10	adr2	ADD10	adr10	adr2
A0	ADD9	adr9	adr1	ADD9	adr9	adr1

Notes 1. μPD42S64165/μPD42S65165

2. μPD42S16165/μPD42S18165

Remark adr(22:1): Physical address bit of CPU Core or DMAAU.

Table 11-3. Example of DRAM Connection and Address Output from VR4121 (2/2)

(b) In 32-bit data bus mode (DBUS32 = 1)

DRAM Address Pin	16-Mbit DRAM (1 Mbit × 16)			64-Mbit DRAM (4 Mbits × 16)		
	ADD pin	Row	Column	ADD pin	Row	Column
A12/NC[Note 1]				ADD20	adr20	
A11/NC[Note 2]	ADD20	adr20		ADD23	adr23	
A10/NC[Note 2]	ADD19	adr19		ADD22	adr22	
A9	ADD18	adr18	adr20	ADD18	adr18	adr20
A8	ADD17	adr17	adr19	ADD17	adr17	adr19
A7	ADD16	adr16	adr8	ADD16	adr16	adr8
A6	ADD15	adr15	adr7	ADD15	adr15	adr7
A5	ADD14	adr14	adr6	ADD14	adr14	adr6
A4	ADD13	adr13	adr5	ADD13	adr13	adr5
A3	ADD12	adr12	adr4	ADD12	adr12	adr4
A2	ADD11	adr11	adr3	ADD11	adr11	adr3
A1	ADD10	adr10	adr2	ADD10	adr10	adr2
A0	ADD9	adr9	adr21	ADD9	adr9	adr21

Notes 1. μPD42S64165/μPD42S65165

2. μPD42S16165/μPD42S18165

Remark adr(23:2): Physical address bit of CPU Core or DMAAU.

Table 11-4 shows an example of connection between the VR4121 and SDRAM.

★ The address multiplex method is also supported in SDRAM. The ADD20 pin (the A10 pin in SDRAM) in the VR4121 performs the control that supports the precharge command.

Table 11-4. Examples of SDRAM Connection and Addresses Output from VR4121

(a) In 16-bit data bus mode (DBUS32 = 0)

SDRAM Address Pin	64-Mbit SDRAM (1 Mbit × 16 × 4)			128-Mbit SDRAM (2 Mbits × 16 × 4)		
	ADD pin	Row	Column	ADD pin	Row	Column
A13	ADD22	adr22		ADD22	adr22	
A12	ADD21	adr21		ADD21	adr21	
A11	ADD19	adr19		ADD19	adr19	
A10	ADD20	adr20	CMD	ADD20	adr20	CMD
A9	ADD17	adr17		ADD17	adr17	
A8	ADD18	adr18		ADD18	adr18	adr23
A7	ADD9	adr9	adr1	ADD9	adr9	adr1
A6	ADD16	adr16	adr8	ADD16	adr16	adr8
A5	ADD15	adr15	adr7	ADD15	adr15	adr7
A4	ADD14	adr14	adr6	ADD14	adr14	adr6
A3	ADD13	adr13	adr5	ADD13	adr13	adr5
A2	ADD12	adr12	adr4	ADD12	adr12	adr4
A1	ADD11	adr11	adr3	ADD11	adr11	adr3
A0	ADD10	adr10	adr2	ADD10	adr10	adr2

Remark adr(23:1): Physical address bit of CPU core or DMAAU

(b) In 32-bit data bus mode (DBUS32 = 1)

SDRAM Address Pin	64-Mbit SDRAM (512 Kbits × 32 × 4)			64-Mbit SDRAM (1 Mbit × 16 × 4)			128-Mbit SDRAM (2 Mbits × 16 × 4)		
	ADD pin	Row	Column	ADD pin	Row	Column	ADD pin	Row	Column
A13				ADD23	adr23		ADD23	adr23	
A12	ADD22	adr22		ADD22	adr22		ADD22	adr22	
A11	ADD21	adr21		ADD21	adr21		ADD21	adr21	
A10	ADD20	adr20	CMD	ADD20	adr20	CMD	ADD20	adr20	CMD
A9	ADD18	adr18		ADD18	adr18		ADD18	adr18	
A8	ADD19	adr19		ADD19	adr19		ADD19	adr19	adr24
A7	ADD17	adr17	adr9	ADD17	adr17	adr9	ADD17	adr17	adr9
A6	ADD16	adr16	adr8	ADD16	adr16	adr8	ADD16	adr16	adr8
A5	ADD15	adr15	adr7	ADD15	adr15	adr7	ADD15	adr15	adr7
A4	ADD14	adr14	adr6	ADD14	adr14	adr6	ADD14	adr14	adr6
A3	ADD13	adr13	adr5	ADD13	adr13	adr5	ADD13	adr13	adr5
A2	ADD12	adr12	adr4	ADD12	adr12	adr4	ADD12	adr12	adr4
A1	ADD11	adr11	adr3	ADD11	adr11	adr3	ADD11	adr11	adr3
A0	ADD10	adr10	adr2	ADD10	adr10	adr2	ADD10	adr10	adr2

Remark adr(24:2): Physical address bit of CPU core or DMAAU

11.3.2 Connection to ROM

Table 11-5 shows a connection example between the VR4121 and ROM.

Table 11-5. Examples of ROM Connection and Address Output from VR4121 (1/2)

(a) In 16-bit data bus mode (DBUS32 = 0)

ROM Address Pin	32-Mbit ROM (2 Mbits × 16)		64-Mbit ROM (4 Mbits × 16)		128-Mbit ROM (8 Mbits × 16)	
	ADD pin	adr	ADD pin	adr	ADD pin	adr
A22					ADD23	adr23
A21			ADD22	adr22	ADD22	adr22
A20	ADD21	adr21	ADD21	adr21	ADD21	adr21
A19	ADD20	adr20	ADD20	adr20	ADD20	adr20
A18	ADD19	adr19	ADD19	adr19	ADD19	adr19
A17	ADD18	adr18	ADD18	adr18	ADD18	adr18
A16	ADD17	adr17	ADD17	adr17	ADD17	adr17
A15	ADD16	adr16	ADD16	adr16	ADD16	adr16
A14	ADD15	adr15	ADD15	adr15	ADD15	adr15
A13	ADD14	adr14	ADD14	adr14	ADD14	adr14
A12	ADD13	adr13	ADD13	adr13	ADD13	adr13
A11	ADD12	adr12	ADD12	adr12	ADD12	adr12
A10	ADD11	adr11	ADD11	adr11	ADD11	adr11
A9	ADD10	adr10	ADD10	adr10	ADD10	adr10
A8	ADD9	adr9	ADD9	adr9	ADD9	adr9
A7	ADD8	adr8	ADD8	adr8	ADD8	adr8
A6	ADD7	adr7	ADD7	adr7	ADD7	adr7
A5	ADD6	adr6	ADD6	adr6	ADD6	adr6
A4	ADD5	adr5	ADD5	adr5	ADD5	adr5
A3	ADD4	adr4	ADD4	adr4	ADD4	adr4
A2	ADD3	adr3	ADD3	adr3	ADD3	adr3
A1	ADD2	adr2	ADD2	adr2	ADD2	adr2
A0	ADD1	adr1	ADD1	adr1	ADD1	adr1

Remark adr(23:1): Physical address bit of CPU Core or DMAAU.

Table 11-5. Examples of ROM Connection and Address Output from VR4121 (2/2)

(b) In 32-bit data bus mode (DBUS32 = 1)

ROM Address Pin	32-Mbit ROM (2 Mbits × 16)		64-Mbit ROM (4 Mbits × 16)		128-Mbit ROM (8 Mbits × 16)	
	ADD pin	adr	ADD pin	adr	ADD pin	adr
A22					ADD24	adr24
A21			ADD23	adr23	ADD23	adr23
A20	ADD22	adr22	ADD22	adr22	ADD22	adr22
A19	ADD21	adr21	ADD21	adr21	ADD21	adr21
A18	ADD20	adr20	ADD20	adr20	ADD20	adr20
A17	ADD19	adr19	ADD19	adr19	ADD19	adr19
A16	ADD18	adr18	ADD18	adr18	ADD18	adr18
A15	ADD17	adr17	ADD17	adr17	ADD17	adr17
A14	ADD16	adr16	ADD16	adr16	ADD16	adr16
A13	ADD15	adr15	ADD15	adr15	ADD15	adr15
A12	ADD14	adr14	ADD14	adr14	ADD14	adr14
A11	ADD13	adr13	ADD13	adr13	ADD13	adr13
A10	ADD12	adr12	ADD12	adr12	ADD12	adr12
A9	ADD11	adr11	ADD11	adr11	ADD11	adr11
A8	ADD10	adr10	ADD10	adr10	ADD10	adr10
A7	ADD9	adr9	ADD9	adr9	ADD9	adr9
A6	ADD8	adr8	ADD8	adr8	ADD8	adr8
A5	ADD7	adr7	ADD7	adr7	ADD7	adr7
A4	ADD6	adr6	ADD6	adr6	ADD6	adr6
A3	ADD5	adr5	ADD5	adr5	ADD5	adr5
A2	ADD4	adr4	ADD4	adr4	ADD4	adr4
A1	ADD3	adr3	ADD3	adr3	ADD3	adr3
A0	ADD2	adr2	ADD2	adr2	ADD2	adr2

Remark adr(24:2): Physical address bit of CPU Core or DMAAU.

Table 11-6. Examples of SROM Connection and Address Output from VR4121

(a) In 16-bit data bus mode (DBUS32 = 0)

SROM Address Pin	32-Mbit SROM (2 Mbits × 16)		
	ADD pin	Row	Column
A12	ADD21	adr21	
A11	ADD20	adr20	
A10	ADD19	adr19	
A9	ADD17	adr17	
A8	ADD18	adr18	
A7	ADD9	adr9	adr1
A6	ADD16	adr16	adr8
A5	ADD15	adr15	adr7
A4	ADD14	adr14	adr6
A3	ADD13	adr13	adr5
A2	ADD12	adr12	adr4
A1	ADD11	adr11	adr3
A0	ADD10	adr10	adr2

Remark adr(21:1): Physical address bit of CPU core or DMAAU

(b) In 32-bit data bus mode (DBUS32 = 1)

SROM Address Pin	32-Mbit SROM (1 Mbit × 32)			32-Mbit SROM (2 Mbits × 16)		
	ADD pin	Row	Column	ADD pin	Row	Column
A12	ADD9	adr9		ADD22	adr22	
A11	ADD21	adr21		ADD21	adr21	
A10	ADD20	adr20		ADD20	adr20	
A9	ADD19	adr19		ADD19	adr19	
A8	ADD18	adr18		ADD18	adr18	
A7	ADD17	adr17		ADD17	adr17	
A6	ADD16	adr16	adr8	ADD16	adr16	adr8
A5	ADD15	adr15	adr7	ADD15	adr15	adr7
A4	ADD14	adr14	adr6	ADD14	adr14	adr6
A3	ADD13	adr13	adr5	ADD13	adr13	adr5
A2	ADD12	adr12	adr4	ADD12	adr12	adr4
A1	ADD11	adr11	adr3	ADD11	adr11	adr3
A0	ADD10	adr10	adr2	ADD10	adr10	adr2

Remark adr(22:2): Physical address bit of CPU core or DMAAU

11.4 Notes on Using BCU

11.4.1 CPU core bus modes

The VR4121 is designed on the assumption that the CPU core is set to the following mode.

- Writeback data rate: D
- Accelerate data ratio: VR4x00 compatible mode

Therefore, set the Config Register as below:

- EP: 0000
- AD: 0

11.4.2 Access data size

In the VR4121, access size is restricted for each address space. Access sizes for the following address spaces are listed below.

★ **Table 11-7. Access Size Restrictions for Address Spaces**

Address Space	R/W	Access Size (bytes)						Remark
		16	8	4	3	2	1	
ROM/PageROM/SROM	R	○	○	○	○	○	○	
Flash memory	W	×	×	△	×	△	×	**Note 1**
System bus I/O space	R/W	○	○	○	×	○	○	
System bus memory space	R/W	○	○	○	×	○	○	
On-chip I/O space 1	R/W	○	○	○	×	○	○	
On-chip I/O space 2	R/W	×	○	○	×	○	×	
LCD space	R/W	×	○	○	×	○	○	**Notes 2, 3**
High-speed system bus memory space	R/W	×	○	○	×	○	○	**Note 3**
DRAM	R/W	○	○	○	○	○	○	

Notes 1. The access size when writing to flash memory must be the same as the data bus width such as below:
In 32-bit data bus mode: 4 bytes
In 16-bit data bus mode: 2 bytes

2. Use as uncached.

3. The LCD space and high-speed system bus memory space are mapped to the same physical address.
Use BCUCNTREG1's ISAM/LCD bit to switch between the two.

Remark ○, △ : accessible, × : not accessible

11.4.3 ROM interface

In the case of the VR4121, whether or not to use SROM is set by setting the SMODE(2:1) pin. Moreover, even if the setting to use SROM is made, it is possible to use other memories depending on the memory bank.

How to set and use ROM other than SROM is described below. When using SROM, see **11.5.6 SROM interface**.

(1) Switching among ROM, PageROM, and Flash Memory Modes

The VR4121 supports three modes (ROM, PageROM and Flash Memory). The mode setting in ROM bank is set via BCUCNTREG1's ROMWEN and PAGEROM bits.

Table 11-8. ROM Mode Settings and Access-Enabled Devices

Mode	Setting		Access-Enabled Devices		
	ROMWEN2/0	PAGEROM2/0	Memory read	Flash Memory register read	Flash Memory write
Ordinary ROM	0	0	Ordinary ROM PageROM Flash Memory	N/A	N/A
PageROM	0	1	PageROM	N/A	N/A
Flash Memory	1	don't care	Ordinary ROM PageROM Flash Memory	Flash Memory	Flash Memory

Remark The initial setting is ordinary ROM mode.

(2) Access Speed Setting

The VR4121 enables the access speed to be changed when operating in ordinary ROM mode or PageROM mode. For details, see **11.5.1 ROM access**.

11.4.4 Flash memory interface

(1) Notes for Specific Modes

The following two modes are available for flash memory.

- Ordinary ROM mode (memory read only)
- Flash Memory mode (supports memory write and register read)

The following notes apply to these modes.

(a) Notes for Ordinary ROM mode

- Write is prohibited
  The WR# pin is not asserted even when a write operation is attempted.
- Flash memory register read is prohibited
  The Ordinary ROM mode is the mode in which bus cycles suite for memory read operations are issued. Since the AC characteristics of flash memory are different for register read and memory read operations, accurate data cannot be obtained by reading the flash memory register while in this mode.

(b) Notes for Flash Memory mode

- Be sure to access in 2-byte or 4-byte units (depending on data bus width) when writing to flash memory.

(2) Example of write sequence for flash memory

An example of a write sequence for flash memory is shown below.

Caution This example's operations have not been confirmed using an actual system.

1. Using GPIO as an output port, apply the flash memory write voltage (V_{PP}).
   If the V_R4121's on-chip GPIOs cannot be used, set up an external output port and then control the write voltage.
2. Set the V_R4121 to flash memory mode (Set "1" to the BCUCNTREG's ROMWEN bit).
3. Wait until the flash memory write voltage become stable.
4. Issue the flash memory write command from the V_R4121.
5. Write data from the V_R4121 to flash memory.
6. Wait until the flash memory write completion signal (ry/by) becomes stable.
7. Wait until the flash memory write completion signal gives notification of write completion.
   After write to flash memory is completed, notification can be obtained by receiving an interrupt from the flash memory write completion signal (ry/by) or by polling the flash memory register.
8. Read the flash memory register.
   - If write succeeded, start processing from "9".
   - If write failed, start processing from "12".
9. If writing new data to flash memory, start processing from "4".
   If write to flash memory is completed, start processing from "10".

10. Compare the data written to flash memory with the original data.
    - If the data matches, perform processing at "11".
    - If the data does not match

      Start processing from "1" when rewriting.

      If processing is interrupted, start processing from "11".
11. Apply the write voltage (VPP) to the flash memory, and end processing after flash memory mode has been canceled.
12. Clear any error data in the flash memory register.
    - If writing again

      If the write voltage is too low, start processing from "1".

      In all other cases, start processing from "4".
    - If processing is completed, perform processing at "11".

(3) Notes for Using Memory Capacity

The capacity of the memory (ROM and DRAM) to be used can be set by using the ROM64 and DRAM64 bits of the BCUCNTREG1 register. In this case, take into consideration that the physical addresses and banks differ depending on the data bus mode, as shown in **6.3.1 ROM address space** and **6.3.5 DRAM space**.

(4) Usage of Expansion Memory when Data Bus Size is 32 bits

When 32-bit data bus size, the expansion memory (ROM and DRAM) can be used by setting the proper values to the EXT_ROM64, EXT_DRAM64, EXT_ROMCS(3:2), and EXT_MEM of BCUCNTREG3. In this case, set EXT_ROM64, EXT_DRAM64, and EXT_ROMCS(3:2) before setting EXT_MEM (access enable). When manipulating these settings, take into consideration that the physical addresses and banks of the memory differ as shown in **6.3.1 ROM address space** and **6.3.5 DRAM space**.

11.4.5 LCD controller interface

(1) Access Size

★ Available access sizes for accessing the LCD controller interface are 1 byte, 2 bytes, 4 bytes, and 8 bytes. 4-word (16-byte) block transfer generated in the cache area is not supported.

(2) Data Inversion

When "0" has been set to the BCUCNTREG1's ISAM/LCD bit and to BCUCNTREG2's GMODE bit, the VR4121 inverts the bits in the data being read or written via the LCD controller interface.

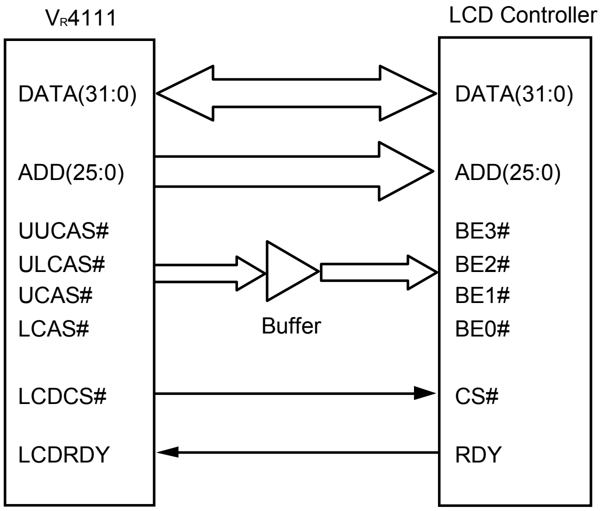
Table 11-9. Example of Bit Inversion in Data in VR4121 and at DATA(15:0) Pins

Data in VR4121	Data at DATA(15:0) Pins
0x0000	0xFFFF
0xA5A5	0x5A5A
0x1234	0xEDCB

(3) Data Bus Size

In the VR4121, if the LCD32 bit of BCUCNTREG3 is set to 1 during the 32-bit data bus mode (DBUS32 = 1), the data bus size of the LCD controller interface is expanded to 32 bits (16 bits as default). In this case, the signals UUCAS#/MRAS3#, ULCAS#/MRAS2#, UCAS#, and LCAS# are used to indicate the effective data on the DATA(31:0) pins, not SHB# nor ADD0.

Figure 11-1. Example of 32-bit LCD Controller Interface Connection



Caution Take sufficient consideration for the countermeasures against noise (such as undershoot, overshoot, or cross talk) for the UUCAS#/MRAS3#, ULCAS#/MRAS2#, UCAS#, and LCAS# signals, because they also function as the DRAM's CAS signals.
Using a buffer is recommended for long lines — for example, use cables to connect an LCD controller.

Remark Since the LCD's address space is 16 Mbytes, only the low-order 24 bits (ADD(23:0)) of the ADD bus are valid.

★ **(4) Access cycle reduction mode**

In the VR4121, an access cycle reduction mode that has a shorter bus-access time than that of the VR4111 is available when the WLCD/M(2:0) bit in BCUSPEEDREG (0X0B00 000A) is set to 111-100. When the WLCD/M(2:0) is 011-000, the access cycle of the VR4121 is the same as that of the VR4111. The table below shows major differences between these two modes. For further details, refer to **11.5.3 LCD interface**.

WLCD/M(2:0) Item	111-100 (Access cycle reduction)	011-000 (VR4111-compatible)
Address output → RD#/WR# ↓	1 TClock	2 TClock
Address output → write data output	0.5 TClock	1 TClock
RD#/WR# ↓ → LCDRDY sampling start	1 TClock	2 TClock
Tlcd setup unit	Every 1 TClock	Every 2 TClock

11.4.6 Load reduction buffer when SDRAM and SROM are used

When using a VR4121 that is connected to SDRAM and SROM, the address and data buses are operated at high speed when accessed, which requires a reduction of load capacity. Consequently, when accessing memory outside of the space allocated to SDRAM and SROM, we recommend connecting the VR4121 to the address and data buses via a load reduction buffer.

In the VR4121, the SYSDIR/GPIO6, HLDACK#, and CKE pins can be used to control the load reduction buffer. The SYSDIR/GPIO6 pin functions as the SYSDIR signal when SDRAM or SROM is used during an RTC reset. In other cases, it functions as the GPIO6 signal. The HLDACK# and CKE pins are shared with their respective functions.

(1) Target space for load reduction buffer

The load reduction buffer can be used with address and data bus to the following address spaces.

- ISA system bus I/O space (0x1400 0000 to 0x17FF FFFF)
- ISA system bus memory space (0x1400 0000 to 0x17FF FFFF)
- LCD interface/high-speed system bus memory space (0x0A00 0000 to 0x0AFF FFFF)

The LCD interface/high-speed system bus memory space enables selection of which buffers to use respectively for high-order and low-order addresses. The setting can be made via the BCUNCNTREG3's LCDSEL(1:0) bit.

(2) Control of load reduction buffer

The load reduction buffer's control signals are used as described below.

- HLDACK#: Buffer direction control used for address buses
  1: CPU → External I/O device
  0: External I/O device → CPU
- SYSDIR/GPIO6: Buffer direction control used for data buses (see **item (3)** below)
  1: External I/O device → CPU
  0: CPU → External I/O device
- CKE: Enables buffer operation (see **item (4)** below)
  1: Disabled
  0: Enabled

Remark When the bus hold function is not being used, the CPU always becomes the address bus's bus master. In such cases, HLDACK# is not used to control the buffer direction, and it can remain set as "CPU → External I/O device".

(3) Buffer control timing for data bus

The status of the SYSDIR signal is as follows, depending on the access type.

- High impedance: During bus hold
- High-level: During read access to one of the following spaces
  ISA system bus I/O space
  ISA system bus memory space
  LCD interface/high-speed system bus memory space, selected by LCDSEL(1:0)
- Low-level: Access other than the above

Even if the CPU is in standby mode due to the power mode, control continues as long as the system bus is operating.

(4) Caution points concerning connection of load reduction buffer

- The CKE signal can be used as an operation-enabling signal only when the SDDRAMMODREG's SCLK bit value is "0". When this SCLK bit is set to "0", the SDRAM/SROM clock is output only when SDRAM and SROM are used. The CKE signal also goes high at that time. When the SCLK bit's value is "1", the SDRAM/SROM clock is continually output and the CKE signal goes high.
  The CKE signal can be used to enable buffer operation only when SDRAM and SROM are not being used. This reduces power consumption when driving a high load.
- The load reduction buffer goes to high impedance state when its operation is disabled. Accordingly, be sure to take measures to protect equipment, such as when apply a pull-up or pull-down resistance to an external I/O device.

11.4.7 Illegal access notification

(1) Types of Illegal Access

Under the following circumstances, the VR4121 provides notification concerning illegal access of the CPU core, by the unmaskable interrupt request (bus error exception) during a read operation, and by the maskable interrupt request (Int0) set with the BCUERRSTREG's BERRST bit during a write operation.

- **Bus deadlock**

  A deadlock is judged when a bus timeout (DRAM refresh interval + α) occurs due to the non-return of a ready signal via the system bus or LCD controller interface, in which case notification of illegal access is given.
  (0 < α < DRAM refresh interval)

- **Address space reserved for future use**

  Notification of illegal access is given when the processor has accessed any of the following addresses.
  0x0FFF FFFF to 0x0D00 0000
  0x09FF FFFF to 0x0400 0000

(2) Notification Method for Illegal Access

The methods used to notify the CPU core are listed below.

Table 11-10. Illegal Access Notification Methods

Access Type	Illegal Access Notification Method
Processor read request	Notification by bus error caused by SysCmd
Processor write request	Notification by interrupt exception (Int0)

Remark To clear the interrupt source caused by a processor write request, write "1" to BCUERRSTREG's bit1 (BERRST).

11.5 Bus Operations

The bus operations of buses controlled by the BCU are described below.

The BCU's operating clock (TClock, internal) appears in the timing chart for each bus operation.

TClock is determined within a range of 26 to 56 MHz by CLKSEL(2:0) pin settings.

★ Note that the TClock can be output from the BUSCLK pin by means of a BCUCNTREG3 setting. In the initial state after RTC reset, however, BUSCLK outputs a 1/4-TClock frequency and the TClock is only used internally.

11.5.1 ROM access

★ The VR4121 supports the following three modes for ROM access other than SROM access. Use BCUCNTREG1's PAGEROM and ROMWEN bits to set the mode.

- Ordinary ROM read mode (ROMWEN, PAGEROM = 00)
- PageROM read mode (ROMWEN, PAGEROM = 01)
- Flash Memory mode (ROMWEN = 1)

Refer to **11.5.6 SROM interface** for details of the SROM access.

(1) Ordinary ROM Read Mode

Set ROMWEN = 0 and PAGEROM = 0.

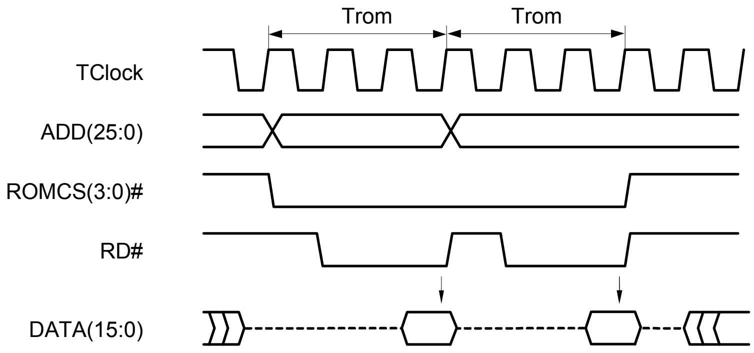
WROMA(2:0) (BCUSPEEDREG(2:0)) can be used to set the access time during the ordinary ROM read cycle (Trom), as shown in Table 11-11.

Figures 11-2 and 11-3 show 4-byte read timing chart data for when WROMA(2:0) is set to "110". If access uses a data size larger than 4 bytes, the Trom cycle is continued until the required access size is reached.

Table 11-11. Access Times During Ordinary ROM Read Mode

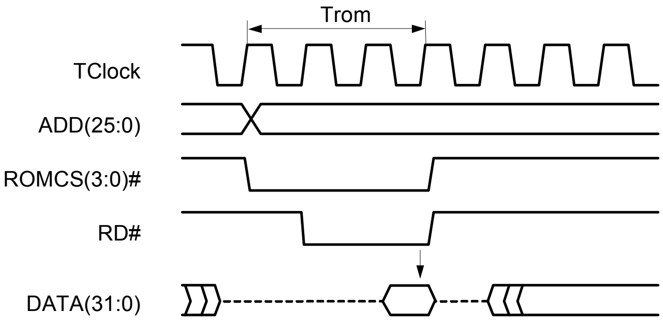
WROMA(2:0)	Trom (TClock)
000	9
001	8
010	7
011	6
100	5
101	4
110	3
111	2

Figure 11-2. ROM 4-Byte Read (16-Bit Mode, WROMA(2:0) = 110)



Remark Broken lines indicate high impedance.

Figure 11-3. ROM 4-Byte Read (32-Bit Mode, WROMA(2:0) = 110)



Remark Broken lines indicate high impedance.

Data is sampled at the rising edge of the TClock following the last Trom-state TClock.
The bus operation types for ordinary ROM are as follows.

1-byte read, 2-byte read, 3-byte read,
1-word read, 2-word read, and 4-word read (1 word = 4 bytes)

(2) PageROM Read Mode

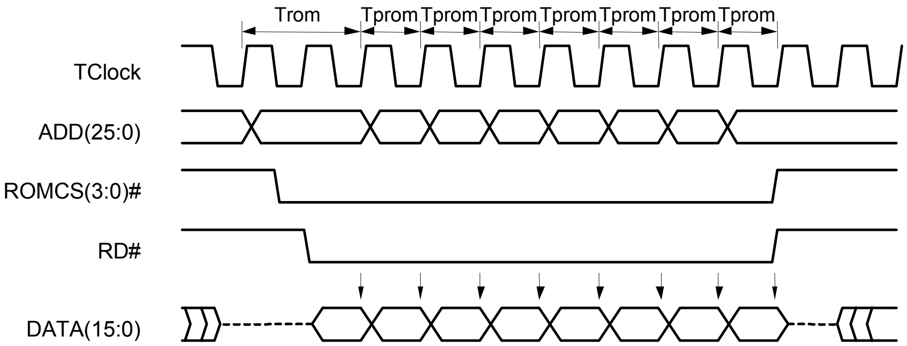
Set ROMWEN = 0 and PAGEROM = 1.
WROMA(2:0) (BCUSPEEDREG(2:0)) and WPROM(1:0) (BCUSPEEDREG(13:12)) can be used to set the access time of the Page ROM read cycle (Tprom).
Figures 11-4 and 11-5 show 4-word (16-byte) read timing charts for when WROMA(2:0) is set to "111" and WPROM(1:0) is set to "10". The ROMCS# and RD# pins are held at low level during Trom cycles.

Table 11-12. PageROM Read Mode Access Time

WROMA(2:0)	Trom (TClock)
000	9
001	8
010	7
011	6
100	5
101	4
110	3
111	2

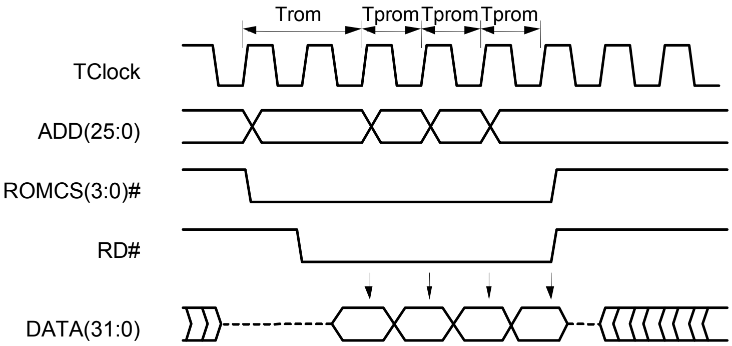
WPROM(1:0)	Tprom (TClock)
00	3
01	2
10	1
11	RFU

Figure 11-4. PageROM 4-Word Read, 16-Bit Mode (WROMA(2:0) = 111, WPROM(1:0) = 10)



Remark Broken lines indicate high impedance.

Figure 11-5. PageROM 4-Word Read, 32-Bit Mode (WROMA(2:0) = 111, WPROM(1:0) = 10)



Remark Broken lines indicate high impedance.

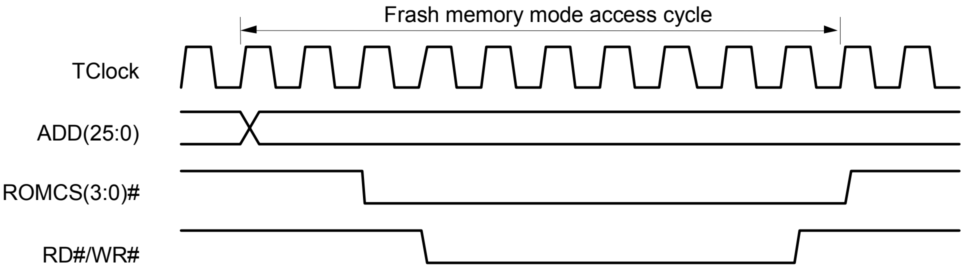
(3) Flash Memory Mode

Set ROMWEN = 1.

This mode is used to meet the characteristics required for writing to flash memory and for accessing the flash memory register. This mode can also be used to read to flash memory.

Note that the access time is constant when in this mode.

Figure 11-6. Flash Memory Mode, 2-Byte Access



11.5.2 System bus access

(1) Access to System Bus of ISA Memory Space and ISA I/O Space

The relationships between the data bus and SHB#, ADD0 signals during 16-/8-bit access to the system bus are shown below.

SHB#	ADD0	IOCS16# MEMCS16#	Write		Read		Remark
			DATA(15:8)	DATA(7:0)	DATA(15:8)	DATA(7:0)	
0	0	0	○	○	◎	◎	
1	0	0	×	○	–	◎	
0	1	0	○	Copy of DATA(15:8)	◎	–	
1	1	0	×	×	–	–	**Note 1**
–	0	1	×	○	–	◎	**Note 2**
–	1	1	×	○	–	◎	**Note 3**

Notes **1.** This combination is never output.

2. Byte access to even addresses

3. Byte access to odd addresses

Remarks ○: Indicates the system bus output valid data.

×: Indicates the system bus output invalid data.

◎ : Indicates the data sampled by system bus.

–: Indicates the data not sampled by system bus.

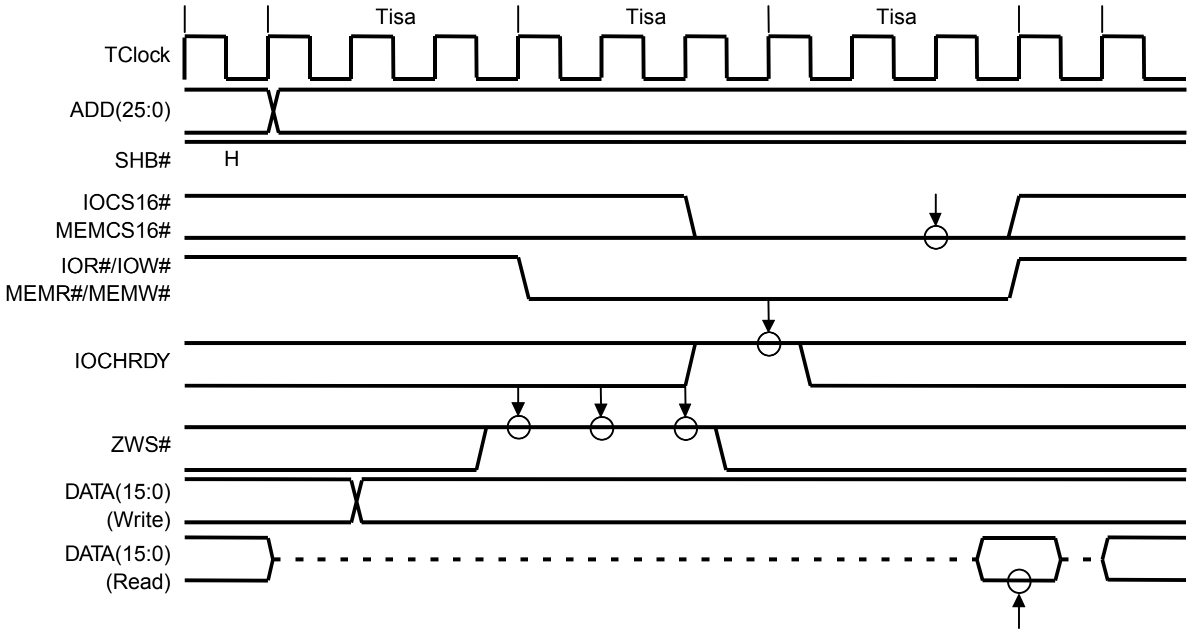
(2) Bus Operations in System Bus of ISA Memory Space and ISA I/O Space

Setting WISAA(2:0) (BCUSPEEDREG(6:4)) can be used to select the access time.

Table 11-13. System Bus Access Times

WISAA(2:0)	Tisa (TClock)
000	8
001	7
010	6
011	5
100	4
101	3
110	RFU
111	RFU

★ **Figure 11-7. 1-Byte Access to Even Address Using 16-Bit Bus (WISAA(2:0) = 101)**



Remarks 1. ○ indicates sampling timing.

2. Broken lines indicate high impedance.

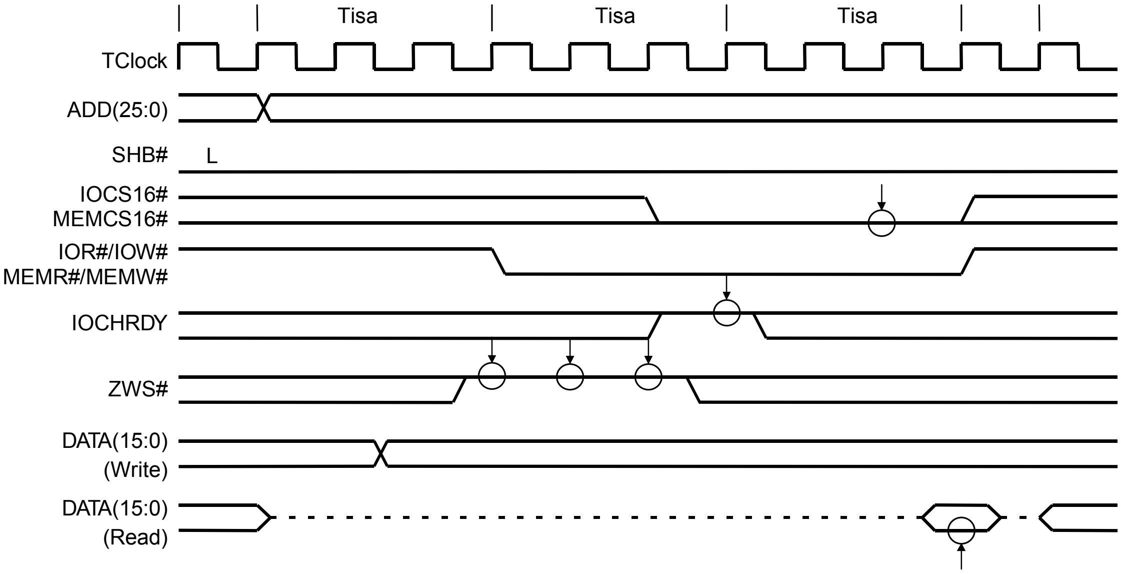
Figure 11-8 illustrates 2-byte access when sampling IOCHRDY at high level.

The IOCHRDY signal is sampled 1-Tisa cycle later from the falling edge of the IOR#/IOW# or MEMR#/MEMW# signal. When IOCHRDY is active, data becomes valid 1-Tisa cycle later. When IOCHRDY is inactive, one more wait of 1-Tisa cycle is inserted.

If the system bus access time has been set as 3 TClocks (WISAA(2:0) = 101), the bus cycle will end after waiting for at least 3 TClocks (1-Tisa cycle) after the ready signal is sampled using IOCHRDY.

If high-level IOCHRDY is sampled, the IOCS16# or MEMCS16# signal is sampled for 1-Tisa cycle from sampling IOCHRDY.

★ **Figure 11-8. 2-Byte Access When Sampling IOCHRDY at High Level Using 16-Bit Bus (WISAA(2:0) = 101)**

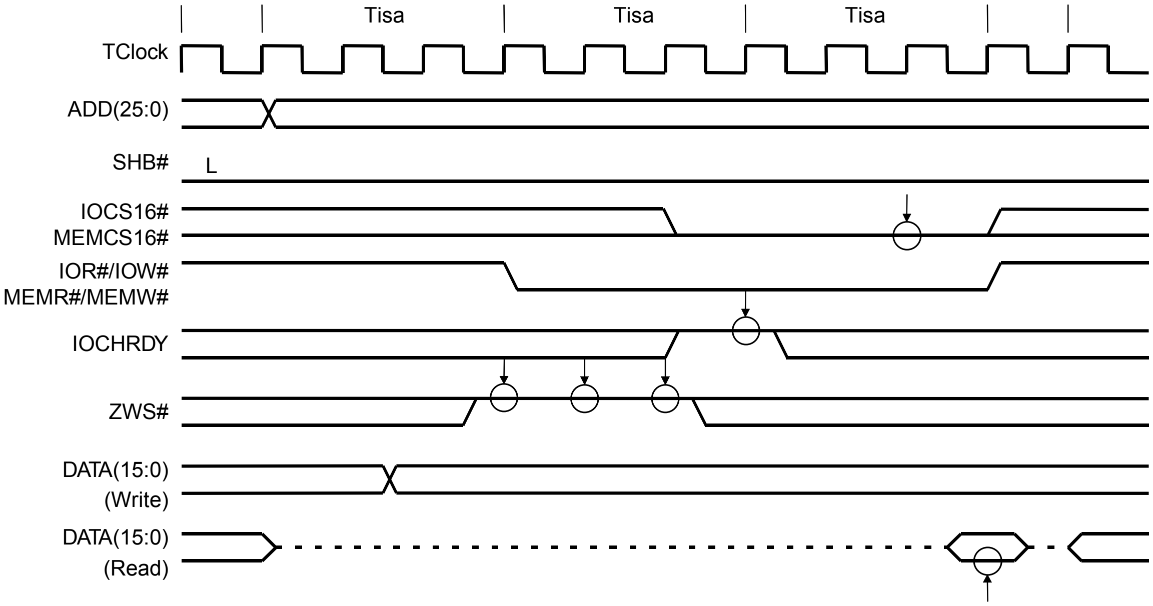


Remarks 1. ○ indicates sampling timing.

2. Broken lines indicate high impedance.

Figures 11-9 and 11-10 show timing charts for 1-byte access.

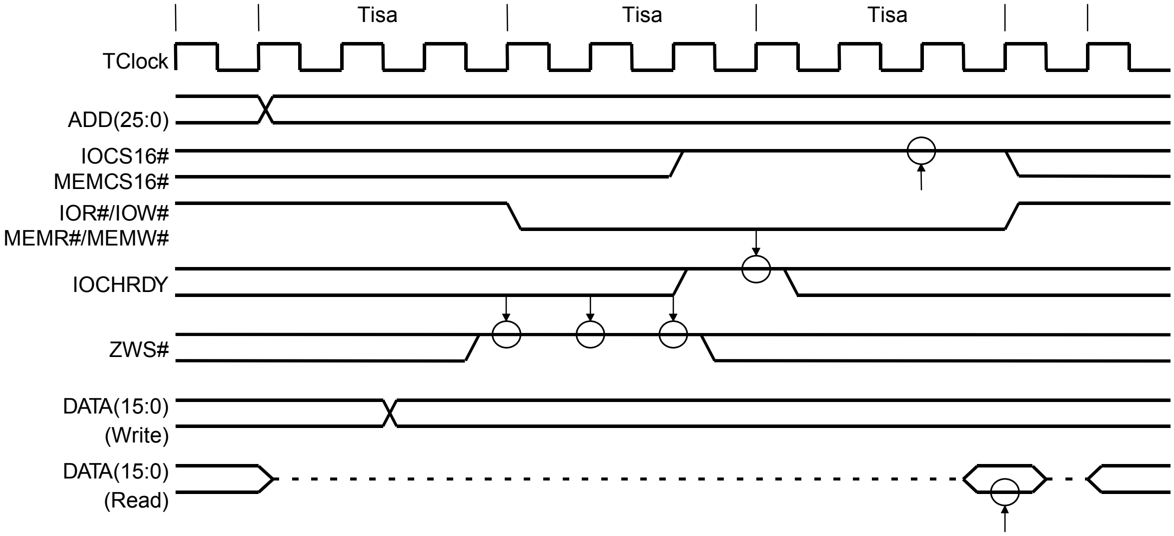
★ **Figure 11-9. 1-Byte Access to Odd Address Using 16-Bit Bus (WISAA(2:0) = 101)**



Remarks 1. ○ indicates sampling timing.

2. Broken lines indicate high impedance.

★ **Figure 11-10. 1-Byte Access to Odd Address Using 8-Bit Bus (WISAA(2:0) = 101)**



Remarks 1. ○ indicates sampling timing.

2. Broken lines indicate high impedance.

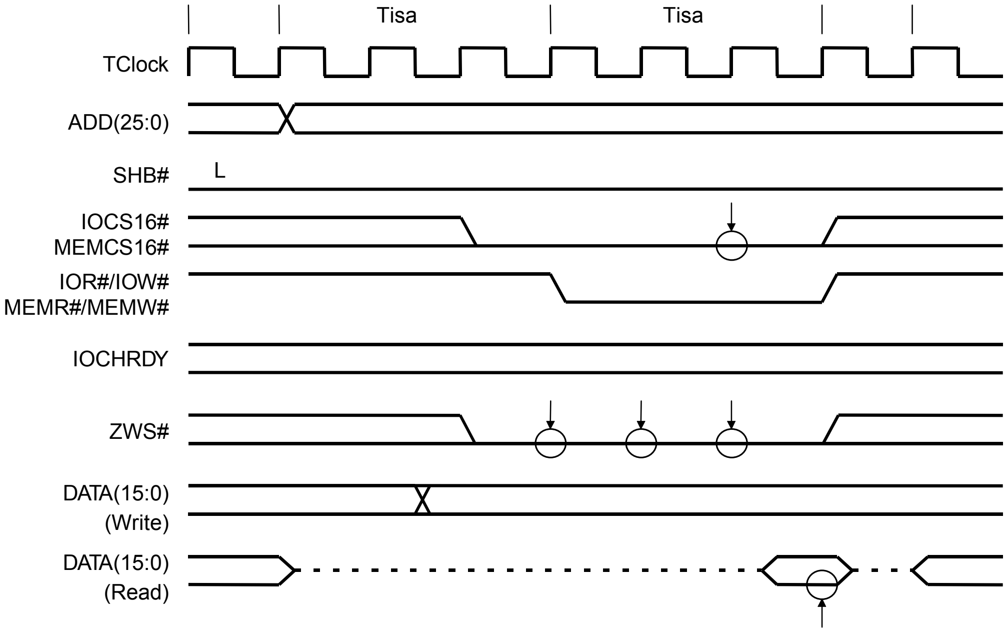
Figures 11-11 and 11-12 illustrate 2-byte access when sampling ZWS# at low level.

The bus cycle will end after waiting for at least 3 TClocks (1-Tisa cycle) after the ready signal is sampled using ZWS#. The ZWS# signal is sampled at every rising edge of TClock in the second and later Tisa periods.

If high-level ZWS# is sampled, one more wait of 1-Tisa cycle is inserted.

Caution Be sure not to change the level of the ZWS# signal in 1-Tisa cycle.

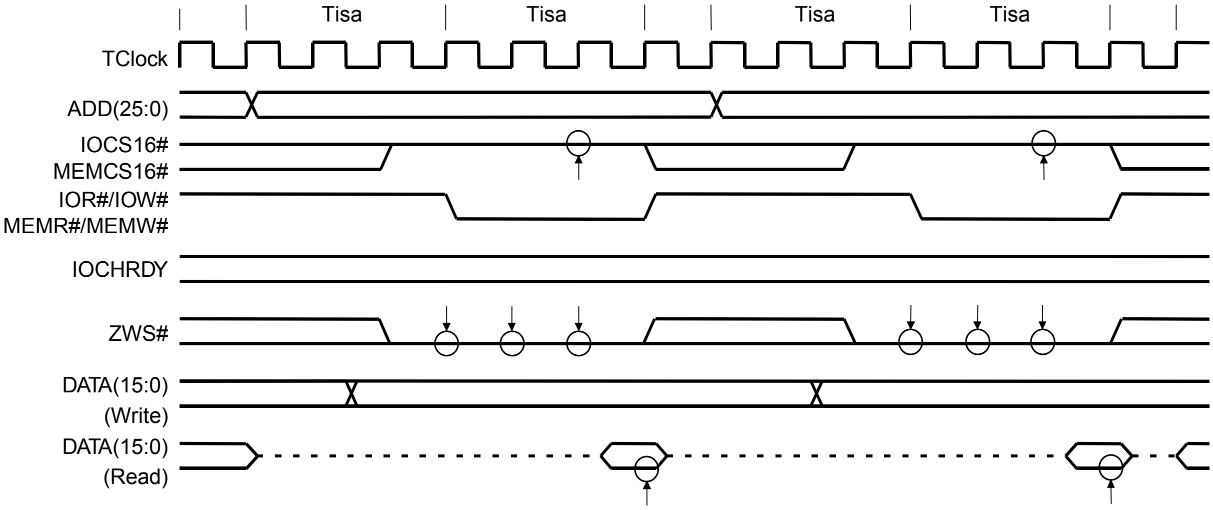
★ **Figure 11-11. 2-Byte Access When Sampling ZWS# at Low Level Using 16-Bit Bus (WISAA(2:0) = 101)**



Remarks 1. ○ indicates sampling timing.

2. Broken lines indicate high impedance.

★ **Figure 11-12. 2-Byte Access When Sampling ZWS# at Low Level Using 8-Bit Bus (WISAA(2:0) = 101)**



Remarks 1. ○ indicates sampling timing.

2. Broken lines indicate high impedance.

(3) Bus Operations in High-Speed System Bus

The high-speed system bus memory space can be selected by setting the ISAM/LCD bit of BCUCNTREG1 to "1". WLCD/M(2:0) (BCUSPEEDREG(10:8)) can be used to set the access time for access to this space, as shown in the table below.

★ The operation of the high-speed system bus is the same as that of system bus in the ISA memory space, except for access time settings and LCDCS# signal activation.

When a data transfer that exceeds the current bus width has been executed, the LCDCS# signal is held low while the next access is continued.

The VR4121 supports both the 16-bit mode and the 32-bit mode for the high-speed system bus. The BCUCNTREG3's LCD32/ISA32 bit is used to switch bus widths. Although 8 bit bus sizing is enabled while in 16-bit mode, bus sizing cannot be done while in 32-bit mode. The bus's operation is the same during 16-bit mode and 32-bit mode, but the respective byte enable signals differ: A0 and SHB# are used during 16-bit mode and UUCAS#/MRAS3#, ULCAS#/MRAS2#, UCAS#, and LCAS# are used during 32-bit mode.

Table 11-14. High-Speed System Bus Access Times

WLCD/M(2:0)	Tisa (TClock)
000	8
001	7
010	6
011	5
100	4
101	3
110	2
111	1

- Wait settings by software

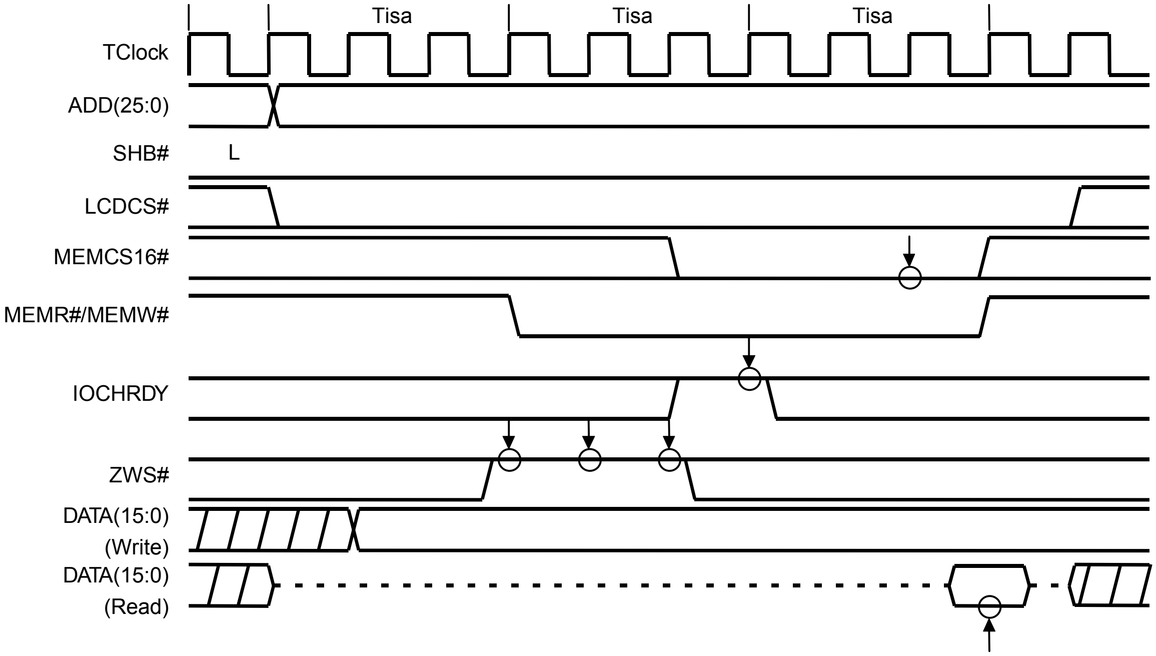
  The VR4121 enables wait settings by software during high-speed system bus access. When the BCUSPEEDREG's MINPLS(1:0) bit has been set, the MEMR#/MEMW# signal's minimum number of active cycles can be changed as Tisa cycles (select using the BCUSPEEDREG's WLCD/M(2:0) bit).

  During high-speed system bus access, the MEMR#/MEMW# signal becomes inactive after the number of wait cycles specified by the MINPLS(1:0) bit have elapsed and after the IOCHRDY signal becomes active, which ends the bus cycle. Accordingly, the minimum number of bus cycles can be set as follows.

  - MINPLS(1:0) = 00 (wait = 1 Tisa)
  - WLCD/M(2:0) = 111 (1Tisa = 1TClock)
  - Low level input starts at initial sampling of ZWS# signal.

  The above settings set a bus cycle count (period until LCDCS# becomes active) of 3 TClocks.

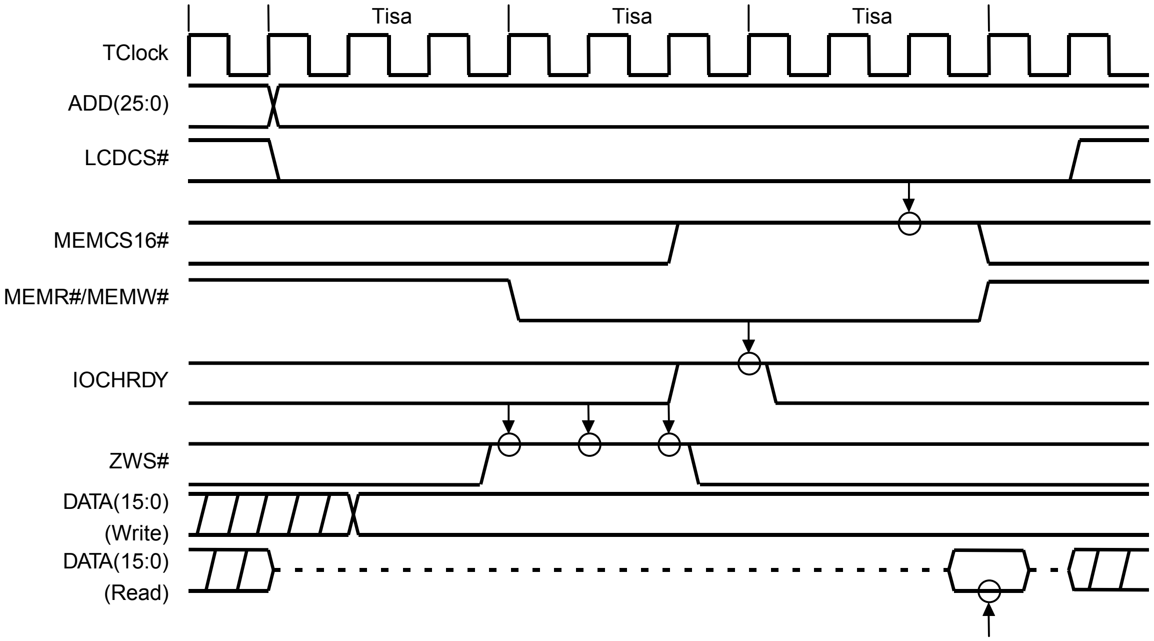
★ **Figure 11-13. 2-Byte Access Using 16-Bit Bus (WLCD/M(2:0) = 101, MINPLS(1:0) = 00, 01)**



Remarks 1. ○ indicates sampling timing.

2. Broken lines indicate high impedance.

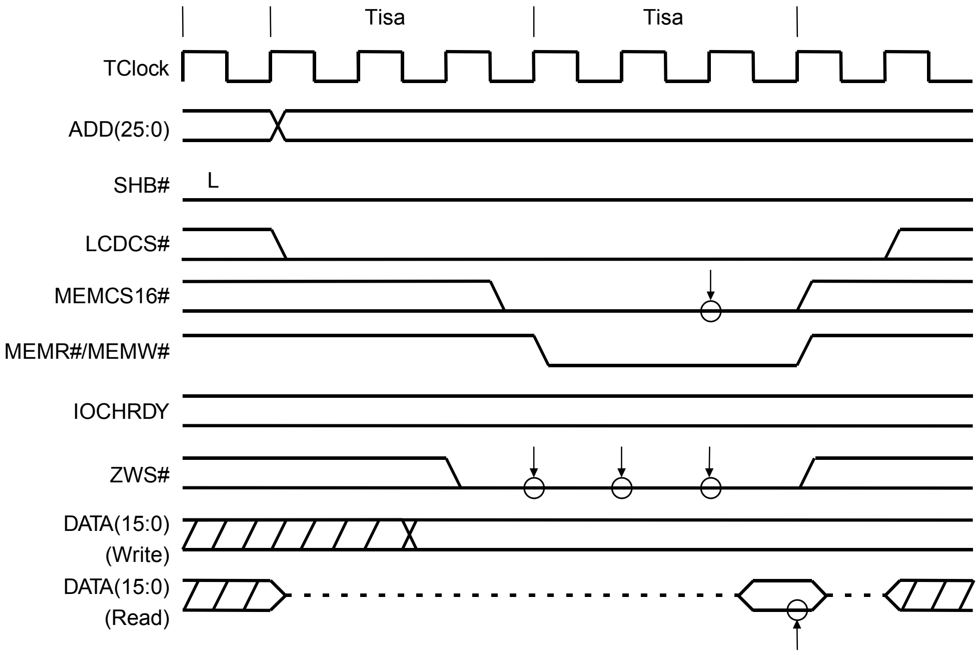
★ **Figure 11-14. 1-Byte Access Using 8-Bit Bus (WLCD/M(2:0) = 101, MINPLS(1:0) = 00, 01)**



Remarks 1. ○ indicates sampling timing.

2. Broken lines indicate high impedance.

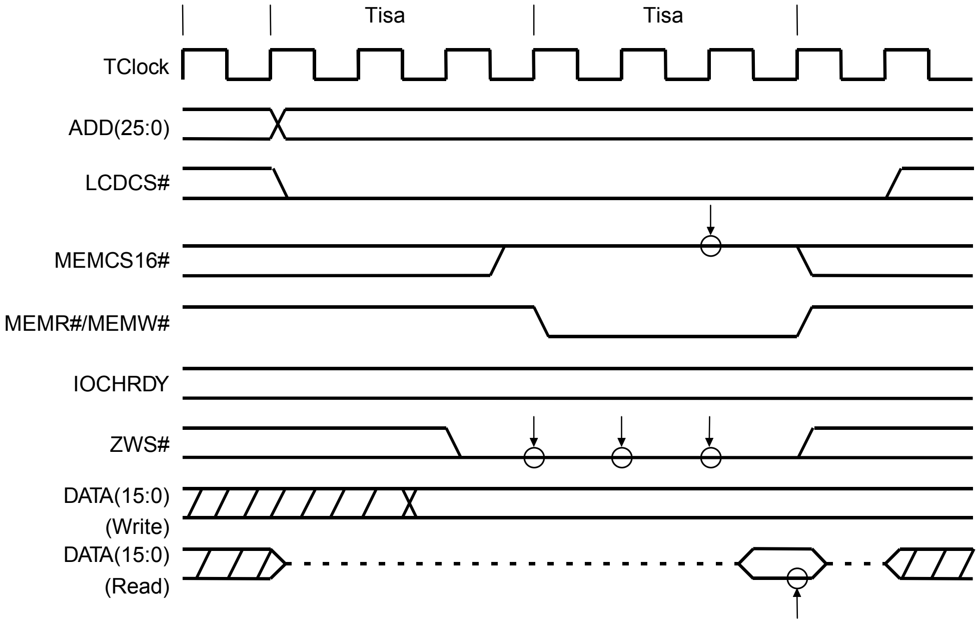
★ **Figure 11-15. 2-Byte Access When Sampling ZWS# at Low Level Using 16-Bit Bus (WLCD/M(2:0) = 101, MINPLS(1:0) = 00)**



Remarks 1. ○ indicates sampling timing.

2. Broken lines indicate high impedance.

★ **Figure 11-16. 1-Byte Access When Sampling ZWS# at Low Level Using 8-Bit Bus (WLCD/M(2:0) = 101, MINPLS(1:0) = 00)**

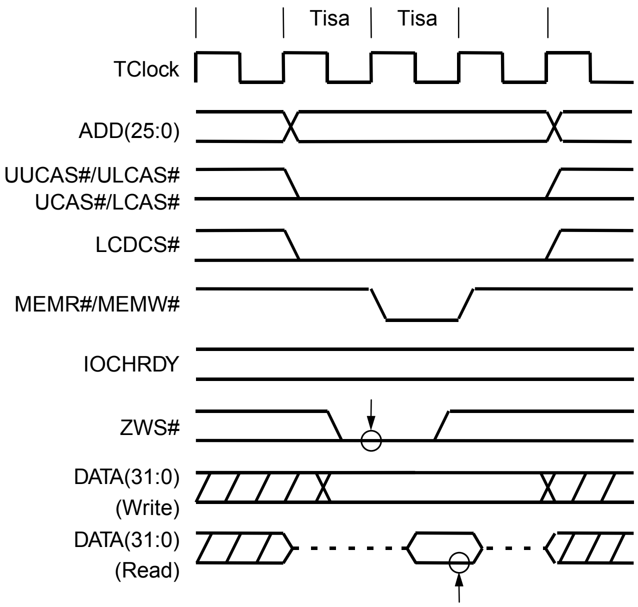


Remarks 1. ○ indicates sampling timing.

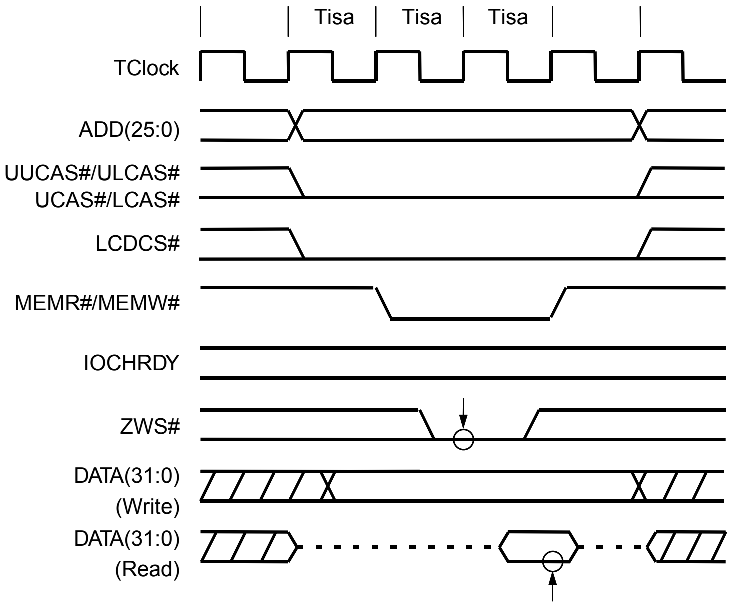
2. Broken lines indicate high impedance.

Figure 11-17. 4-Byte Access when ZWS# Is Sampled at Low Level during 32-Bit Bus Mode (WLCD/M(2:0) = 111) (1/2)

(a) When MINPLS(1:0) = 00



(b) When MINPLS(1:0) = 01

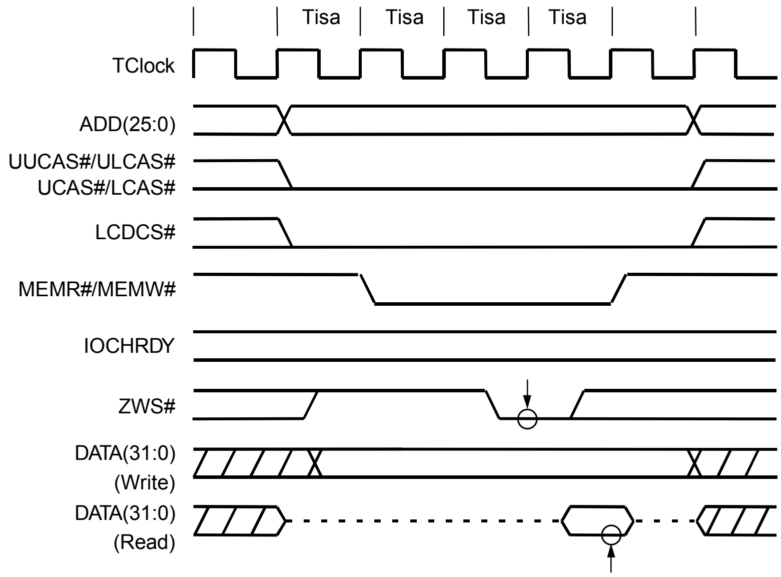


Remarks 1. ○ indicates sampling timing.

2. Broken lines indicate high impedance.

Figure 11-17. 4-Byte Access when ZWS# Is Sampled at Low Level during 32-Bit Bus Mode (WLCD/M(2:0) = 111) (2/2)

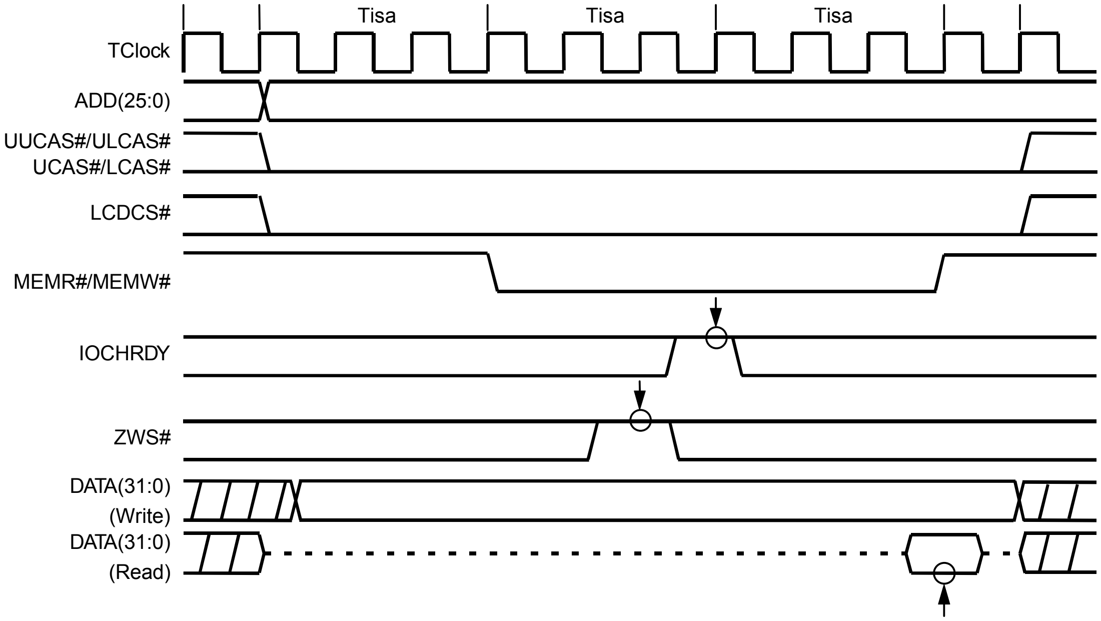
(c) When MINPLS(1:0) = 00, 01, or 10



Remarks 1. ○ indicates sampling timing.

2. Broken lines indicate high impedance.

Figure 11-18. 4-Byte Access when IOCHRDY# Is Sampled at High Level during 32-Bit Bus Mode (WLCD/M(2:0) = 101, MINPLS(1:0) = 00, 01)



Remarks 1. ○ indicates sampling timing.

2. Broken lines indicate high impedance.

11.5.3 LCD interface

The space of the physical address, from 0x0A00 0000 to 0x0AFF FFFF can be used as the LCD space by setting the ISM/LCD bit of the BCUCNTREG1 to 0. WLCD/M(2:0) (BCUSPEEDREG(10:8)) can be used to set the access time.

Table 11-15. Access Times for LCD Interface

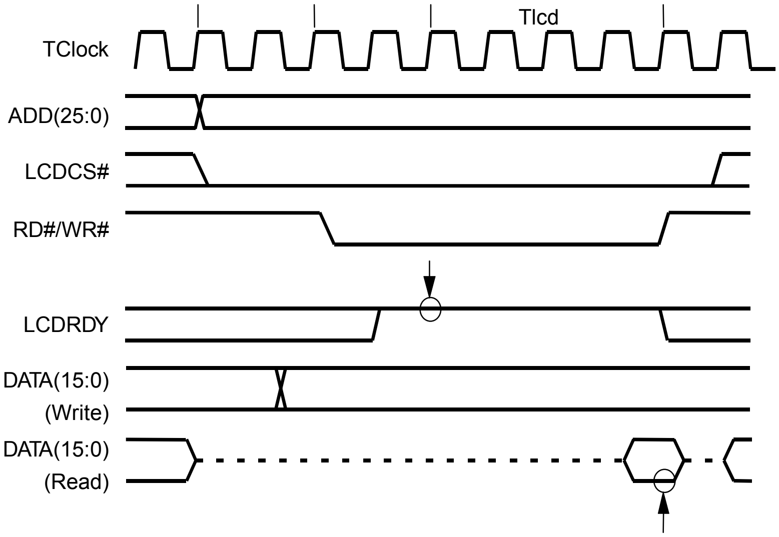
WLCD/M (2:0)	Tlcd (TClock)
000	8
001	6
010	4
011	2
100	5
101	4
110	3
111	2

When the WLCD/M2 bit's value is "0", the LCD interface operates using a VR4111 compatible access cycle. When its value is "1", it operates using a shorter access cycle.

When the LCD interface is used in 16-bit bus, SHB# and ADD0 are used to specify bytes. When the LCD interface is used in 32-bit bus, UUCAS#, ULCAS#, UCAS#, and LCAS# are used to specify bytes.

(1) VR4111-compatible access cycle mode

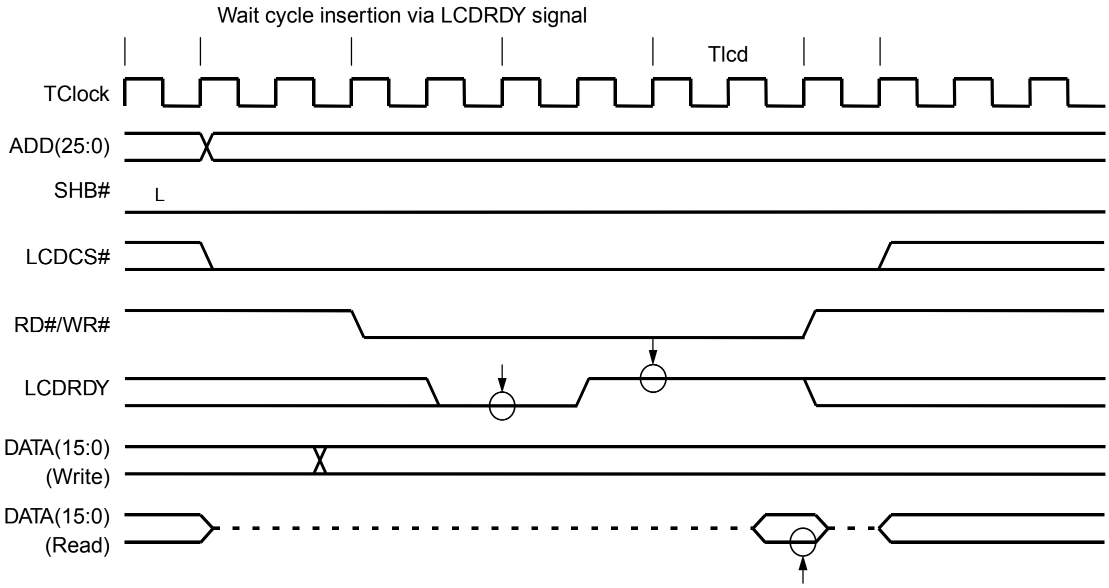
Figure 11-19. 2-Byte Access to LCD Controller during 16-Bit Mode (WLCD/M(2:0) = 010)



Remarks 1. ○ indicates sampling timing.

2. Broken lines indicate high impedance.

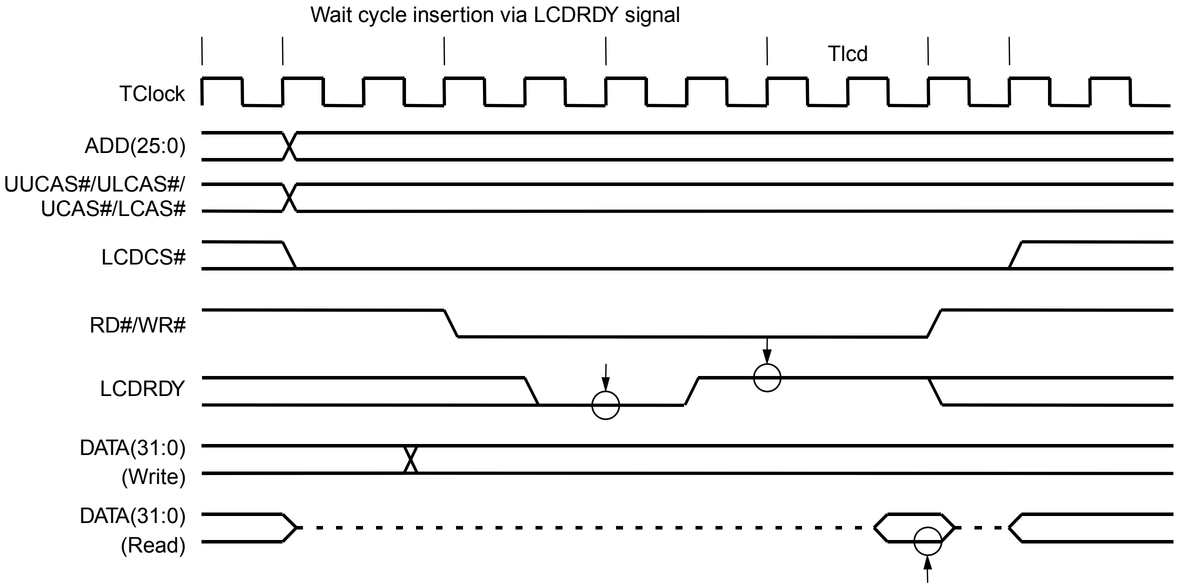
Figure 11-20. 2-Byte Access to LCD Controller during 16-Bit Mode (WLCD/M(2:0) = 011)



Remarks 1. ○ indicates sampling timing.

2. Broken lines indicate high impedance.

Figure 11-21. Access to LCD Controller during 32-Bit Mode (WLCD/M(2:0) = 011)

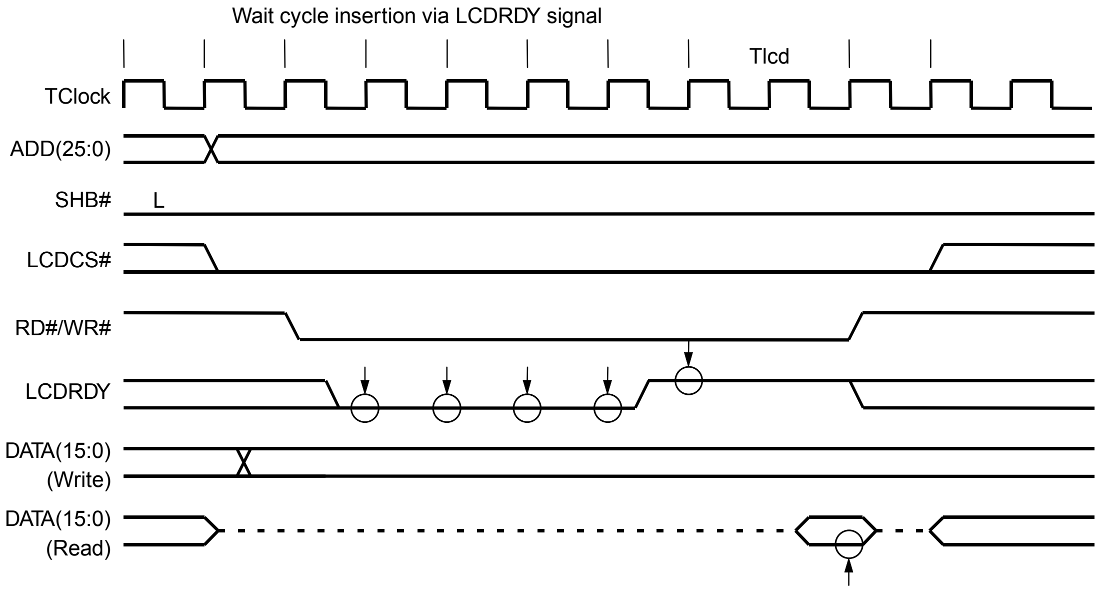


Remarks 1. ○ indicates sampling timing.

2. Broken lines indicate high impedance.

(2) Short access cycle mode

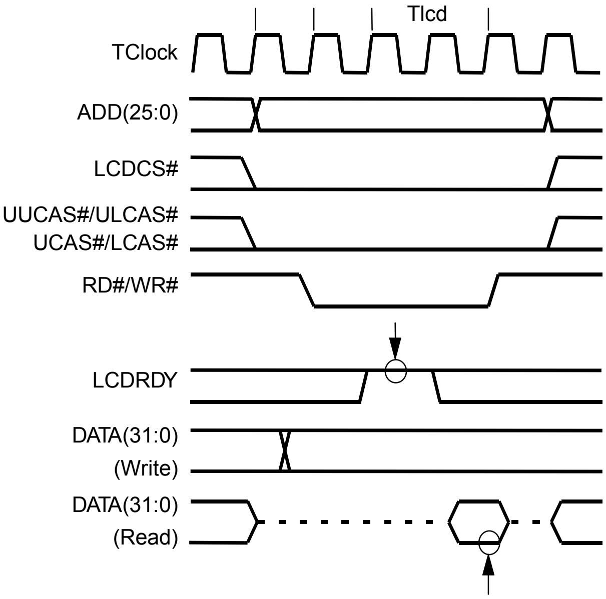
Figure 11-22. 2-Byte Access to LCD Controller during 16-Bit Mode (WLCD/M(2:0) = 100)



Remarks 1. ○ indicates sampling timing.

2. Broken lines indicate high impedance.

Figure 11-23. 4-Byte Access to LCD Controller during 32-Bit Mode (WLCD/M(2:0) = 100)



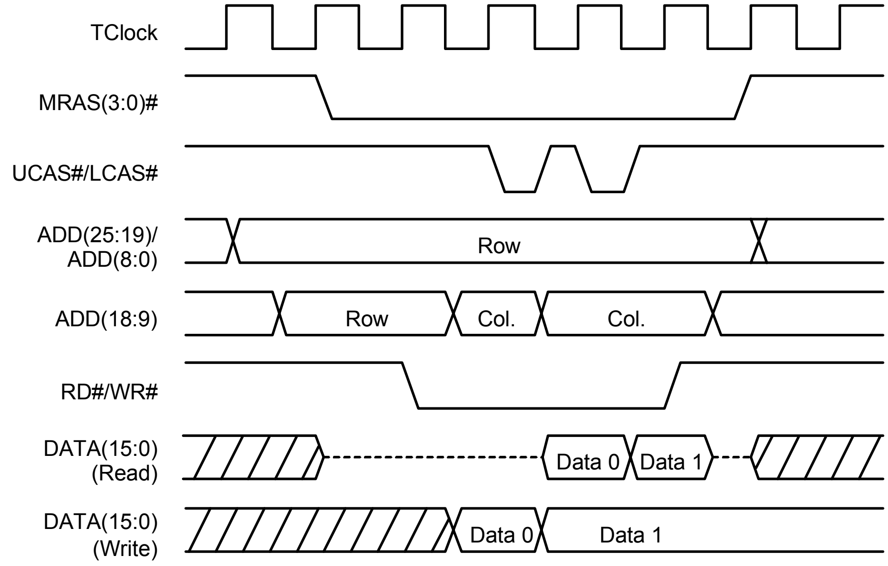
Remarks 1. ○ indicates sampling timing.

2. Broken lines indicate high impedance.

11.5.4 DRAM access (EDO Type)

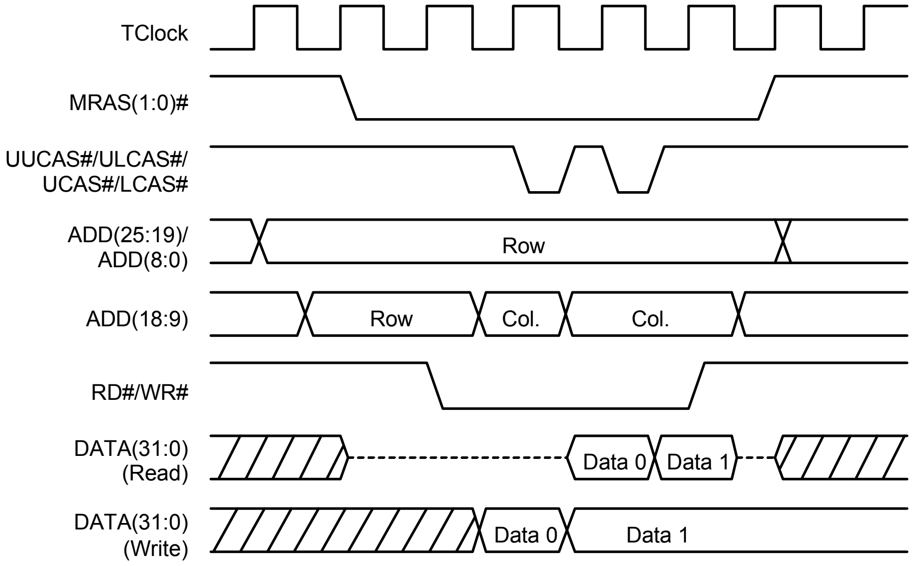
The access time is constant for DRAM.

Figure 11-24. 4-Byte Access to DRAM (16-Bit Mode)



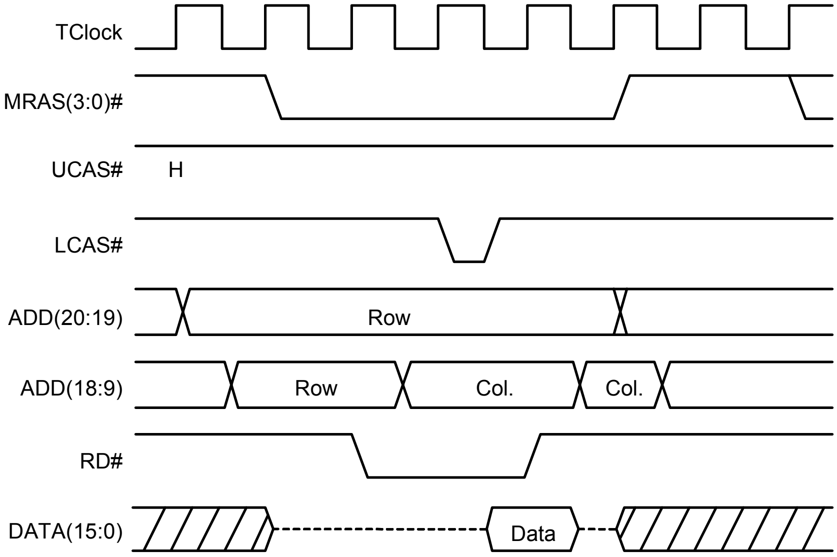
Remark Broken lines indicate high impedance.

Figure 11-25. 8-Byte Access to DRAM (32-Bit Mode)



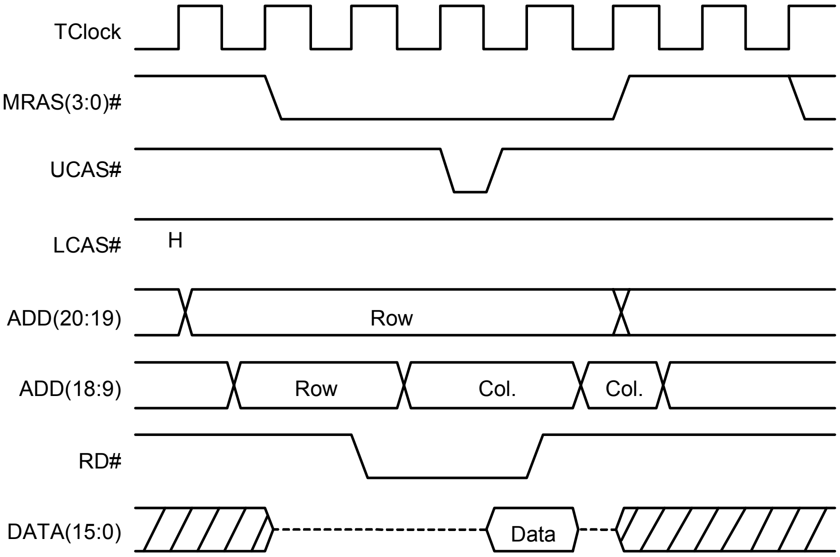
Remark Broken lines indicate high impedance.

Figure 11-26. Byte Read of Odd Address in DRAM (16-Bit Mode)



Remark Broken lines indicate high impedance.

Figure 11-27. Byte Read of Even Address in DRAM (16-Bit Mode)



Remark Broken lines indicate high impedance.

Figure 11-28. Byte Write to Odd Address in DRAM (16-Bit Mode)

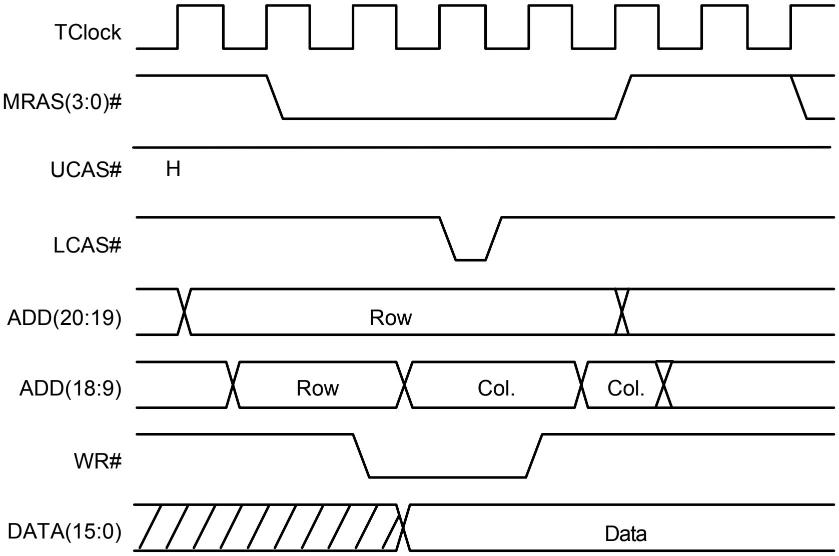


Figure 11-29. Byte Write to Even Address in DRAM (16-Bit Mode)

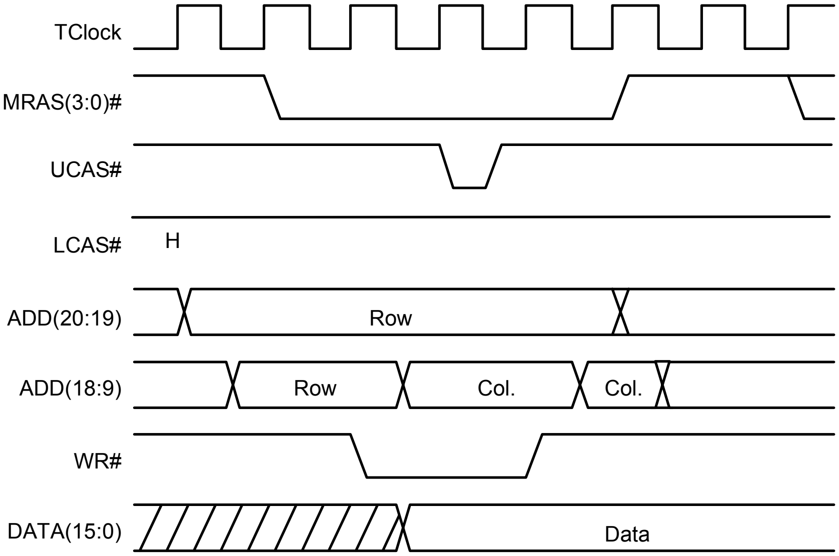
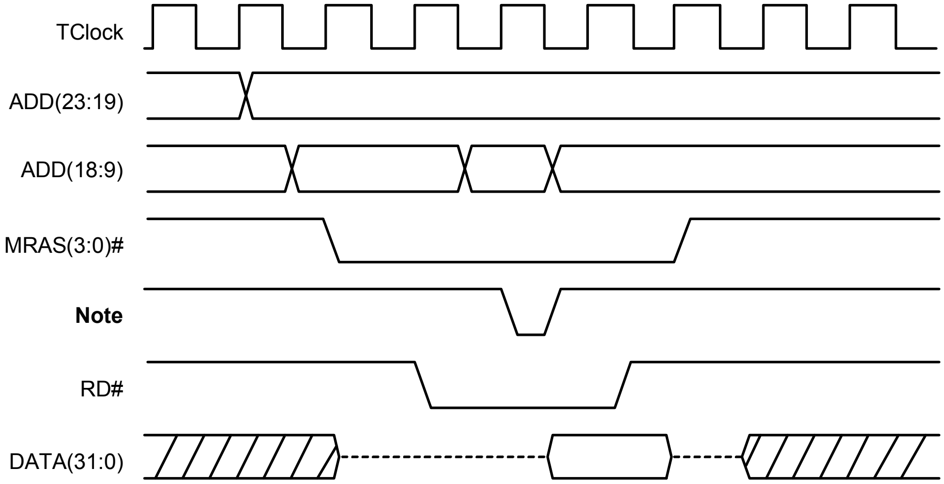


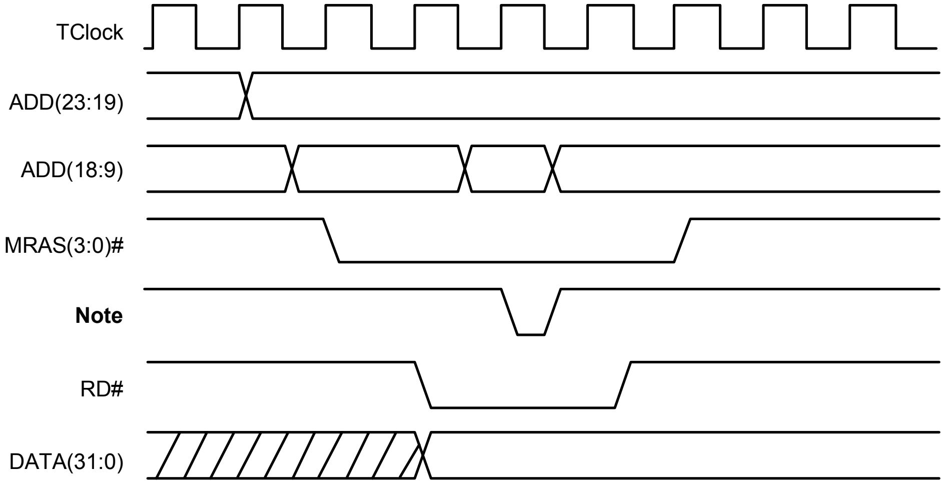
Figure 11-30. Read Cycle for DRAM (32-Bit Mode)



Note When accessing DATA(31:24): UUCAS#
When accessing DATA(23:16): ULCAS#
When accessing DATA(15:8): UCAS#
When accessing DATA(7:0): LCAS#

Remark Broken lines indicate high impedance.

Figure 11-31. Write Cycle for DRAM (32-Bit Mode)



Note When accessing DATA(31:24): UUCAS#
When accessing DATA(23:16): ULCAS#
When accessing DATA(15:8): UCAS#
When accessing DATA(7:0): LCAS#

11.5.5 SDRAM interface

The VR4121 supports bus operations for SDRAM. It includes the following features related to SDRAM access.

- Select from two /CAS latency settings ("2" or "3") (set via SDRAMMODREG).
- Burst mode setting is fixed as "interleave".
- Burst length setting is fixed as "2".
- A command with auto-precharge is issued at the end of each read/write cycle.

The two types of bus operations toward SDRAM are described below.

- Single read/write (1 byte, 2 bytes, 3 bytes, or 1 word (= 4 bytes))
- Block read/write (2 words or 4 words)

The SDRAM interface handles the following parameters. These parameters are set via the SDRAMCNTREG register.

- TRCD: Number of cycles between an active command and a read/write command
- TDLA: Number of cycles between the last write data and an active command
- TRC: Number of cycles between a refresh and another refresh or an active command, or between a self refresh and a refresh or an active command

(1) Single write cycle

The timing when a 1-word write operation is continued for two iterations is shown below. In the figure, <1> and <2> indicate when the commands for writing the first word are issued, and <3> and <4> indicate when the commands for writing the second word are issued.

(a) Writing of first word

<1> Active command is issued.

<2> Write command with auto precharge is issued. At the rising edge of the clock signal following issuance of this command, UUDQM#, ULDQM#, UDQM#, and LDQM# all go to high level to mask the write data.

(b) Writing of second word

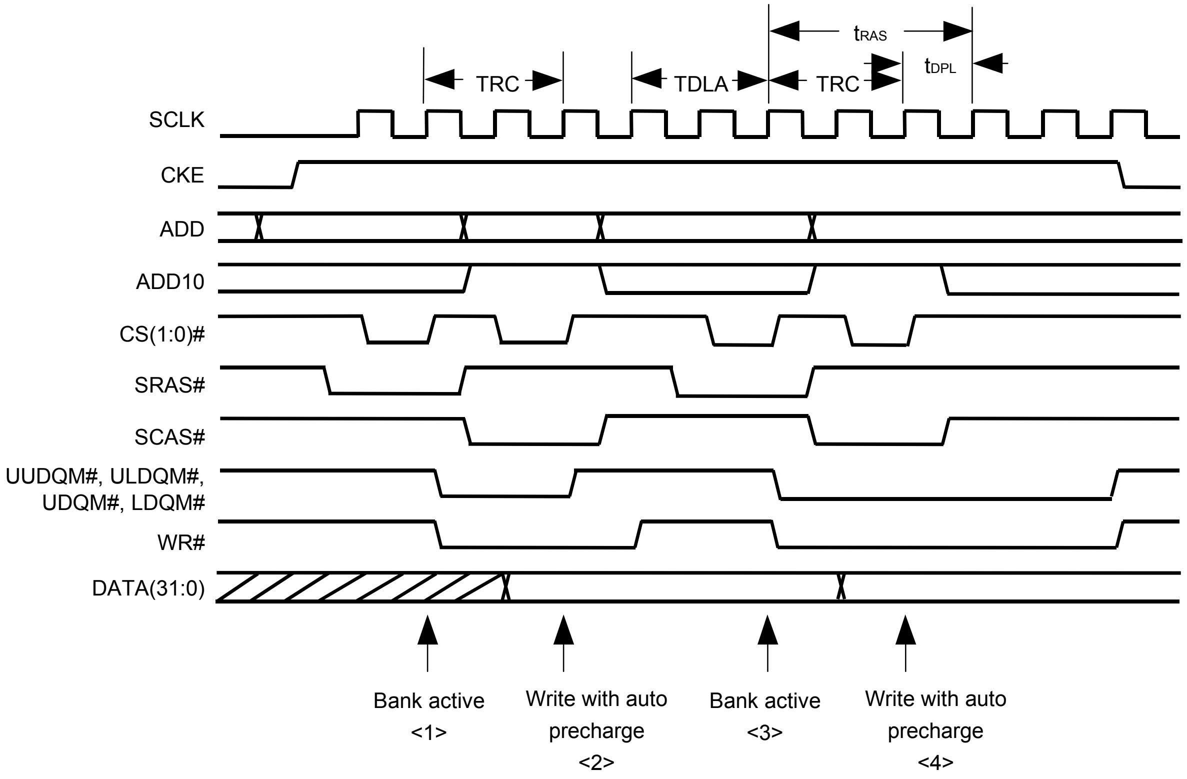
<3> Active command is issued.

<2> Command with auto precharge is issued.

(c) Caution points related to TRCD settings

When setting the value of TRCD in the SDRAMCNTREG register, consider not only the t_{RCD} value of the connected SDRAM but also its t_{RAS} (bank active → precharge) value.

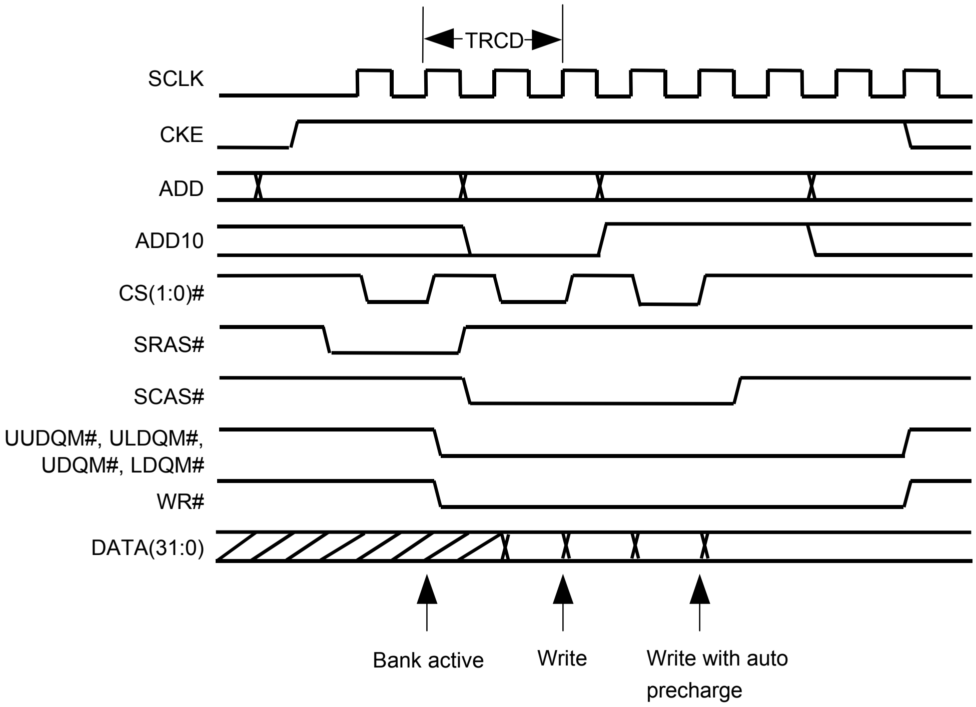
Figure 11-32. Write Cycles Continued during 32-Bit Bus Mode (TRCD(2:0) = 010, TDLA(2:0) = 010)



(2) Block write cycle

The timing of the block write cycle is shown below. At the end of the bus cycle, the VR4121 issues a write command with auto precharge to perform a precharge. Therefore, a write command is issued between the bank active command and the write command with auto precharge.

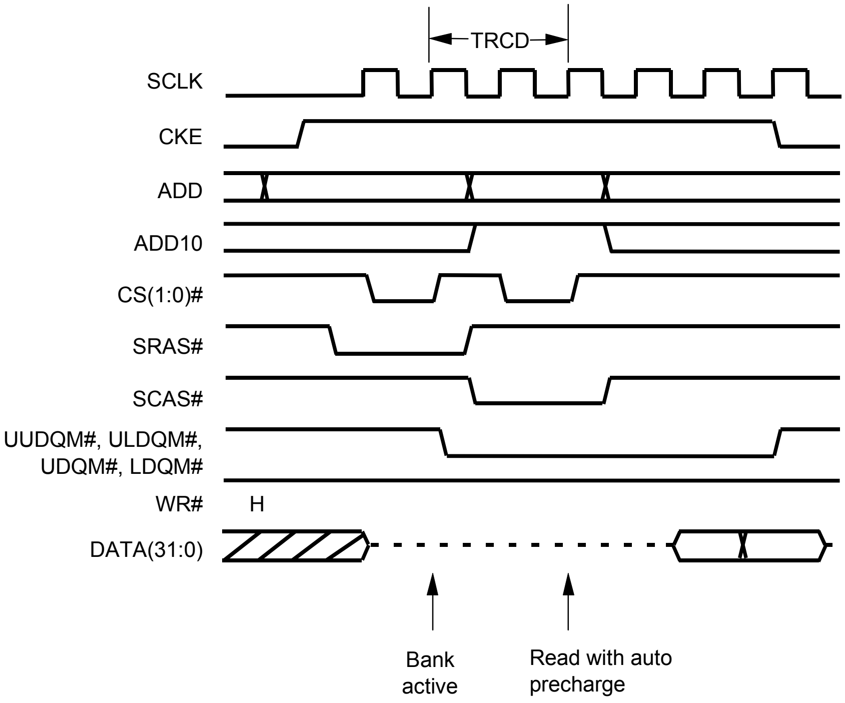
Figure 11-33. Block Write Cycle during 32-Bit Bus Mode (TRCD(2:0) = 010, /CAS latency = 2)



(3) Single read cycle

The timing of the single read cycle is shown below.

Figure 11-34. Single Read Cycle during 32-Bit Bus Mode (TRCD(2:0) = 010, TDLA(2:0) = 010, /CAS latency = 2)

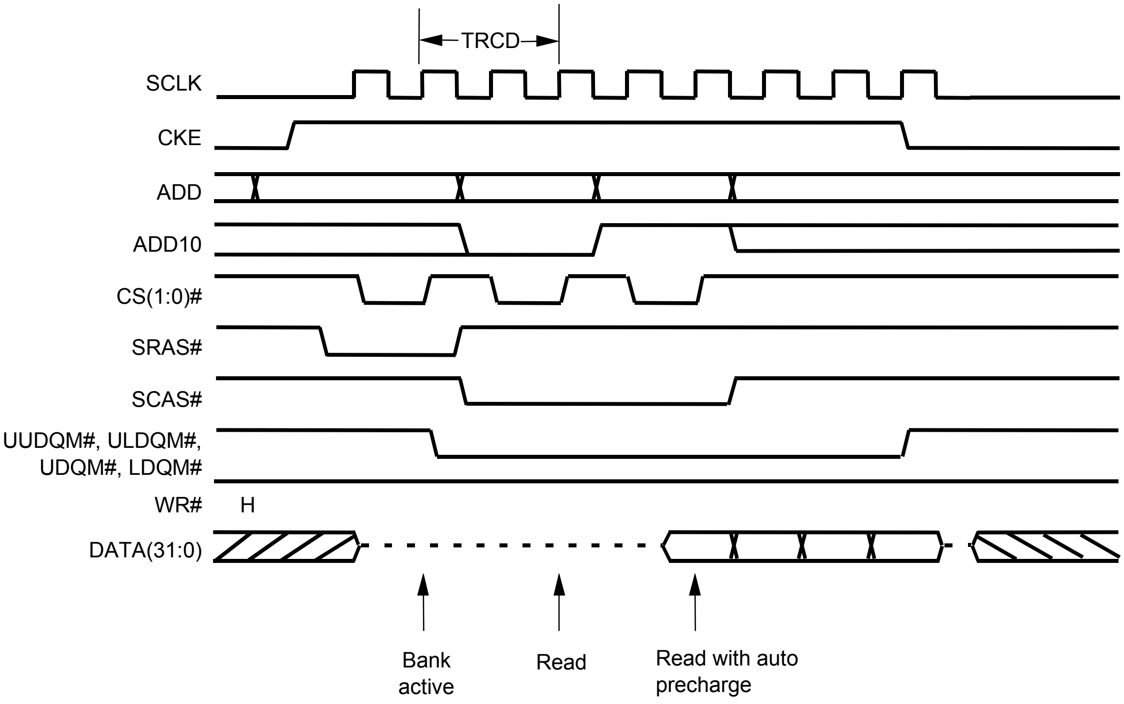


Remark Broken lines indicate high impedance.

(4) Block read cycle

The timing of the block read cycle is shown below. At the end of the bus cycle, the VR4121 issues a read command with auto precharge to perform a precharge. Therefore, a read command is issued between the bank active command and the read command with auto precharge.

Figure 11-35. Block Read Cycle during 32-Bit Bus Mode (TRCD(2:0) = 010, /CAS latency = 2)



Remark Broken lines indicate high impedance.

11.5.6 SROM interface

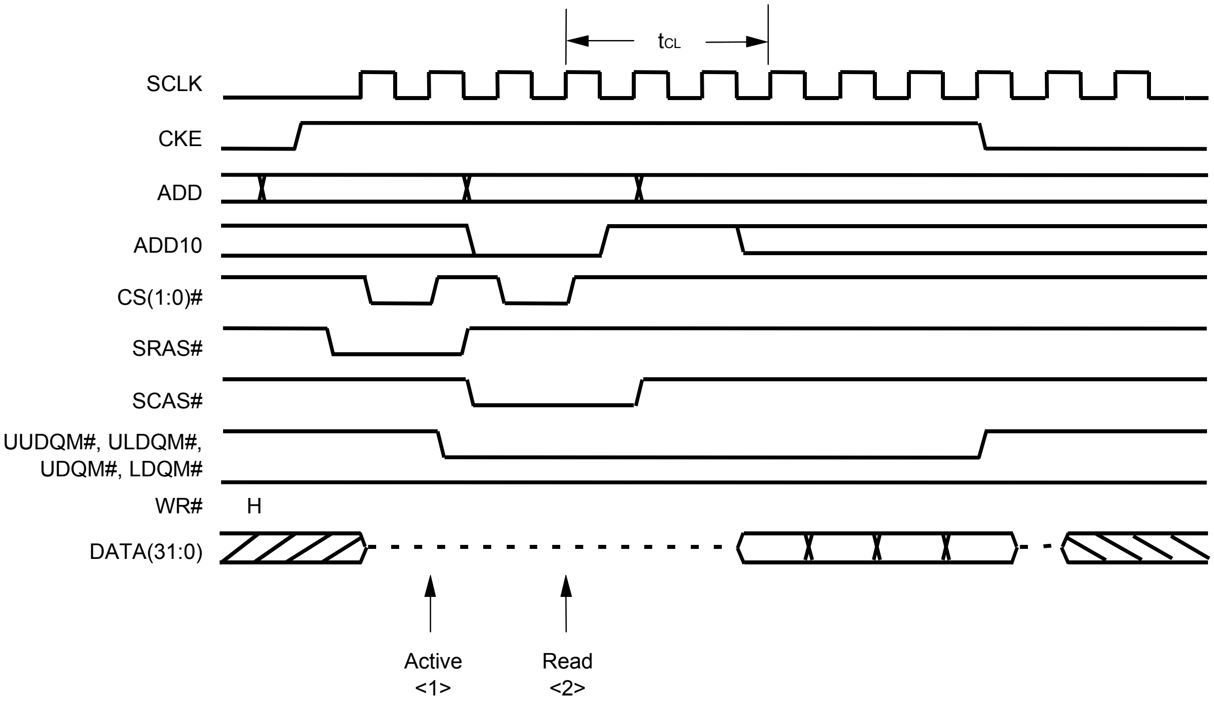
The VR4121 supports bus operations for SROM. It includes the following features related to SROM access.

- /RAS latency value is fixed as "2".
- Select from four /CAS latency settings ("3", "4", "5", or "6") (set via SROMMODREG).
- Burst mode setting is fixed as "interleave".
- Burst length setting is fixed as "4". When 4-word read access occurs during 16-bit mode, the burst access is continued a second time.
- Supports power modes.

The timing of SROM access during 32-bit bus mode is shown below.

Since the /RAS latency value is fixed at "2", the command indicated by <1> is issued two cycles before the command indicated by <2> is issued. Data transfer begins after the second command has been issued and the number of cycles specified by the /CAS latency value have elapsed.

Figure 11-36. 4-Word Read Cycle (/CAS latency = 3) during 32-Bit Bus Mode

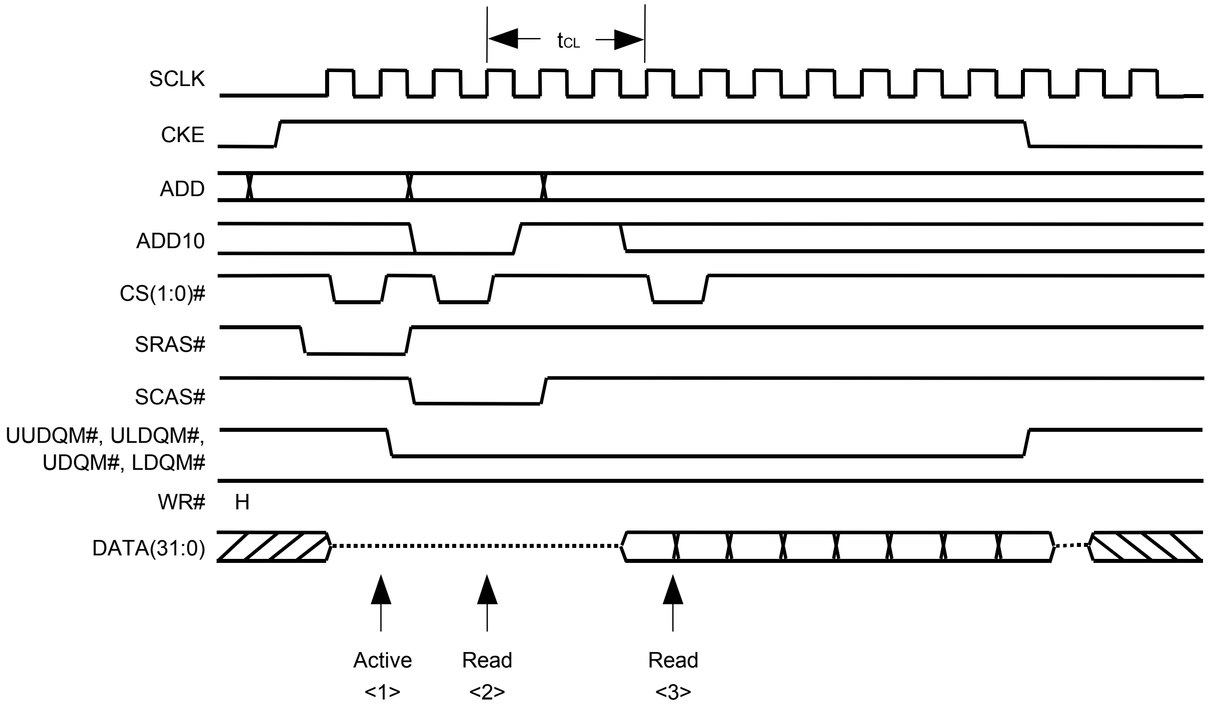


Remark Broken lines indicate high impedance.

The four-word access timing toward SROM during 16-bit bus mode is shown below.

Since the /RAS latency value is fixed at "2", the command indicated by <1> is issued two cycles before the command indicated by <2> is issued. Data transfer begins after the second command has been issued and the number of cycles specified by the /CAS latency value have elapsed. In addition, the read command is issued again (at <3>), four cycles after the command indicated by <2> is issued. The data of the command indicated by <3> is transferred following the data of the command indicated by <2>.

Figure 11-37. 4-Word Read Cycle (/CAS latency = 3) during 16-Bit Bus Mode



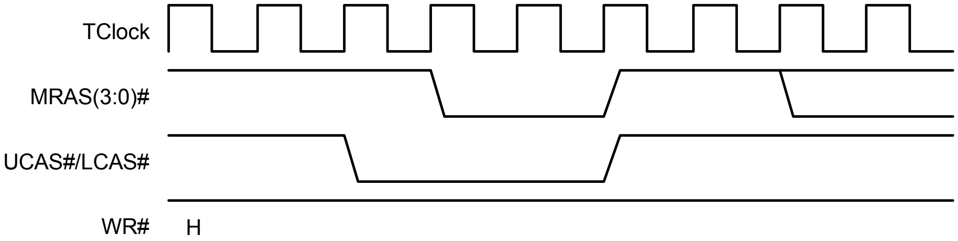
Remark Broken lines indicate high impedance.

11.5.7 Refresh

The VR4121 supports CBR refresh and self refresh.

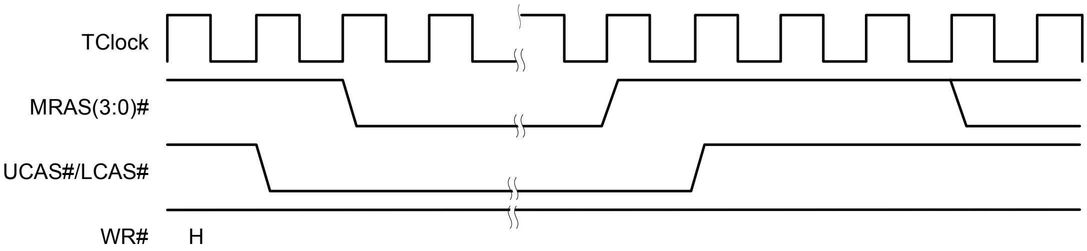
(1) CBR Refresh

Figure 11-38. CBR Refresh (16-Bit Mode)



(2) Self Refresh

Figure 11-39. Self Refresh (16-Bit Mode)

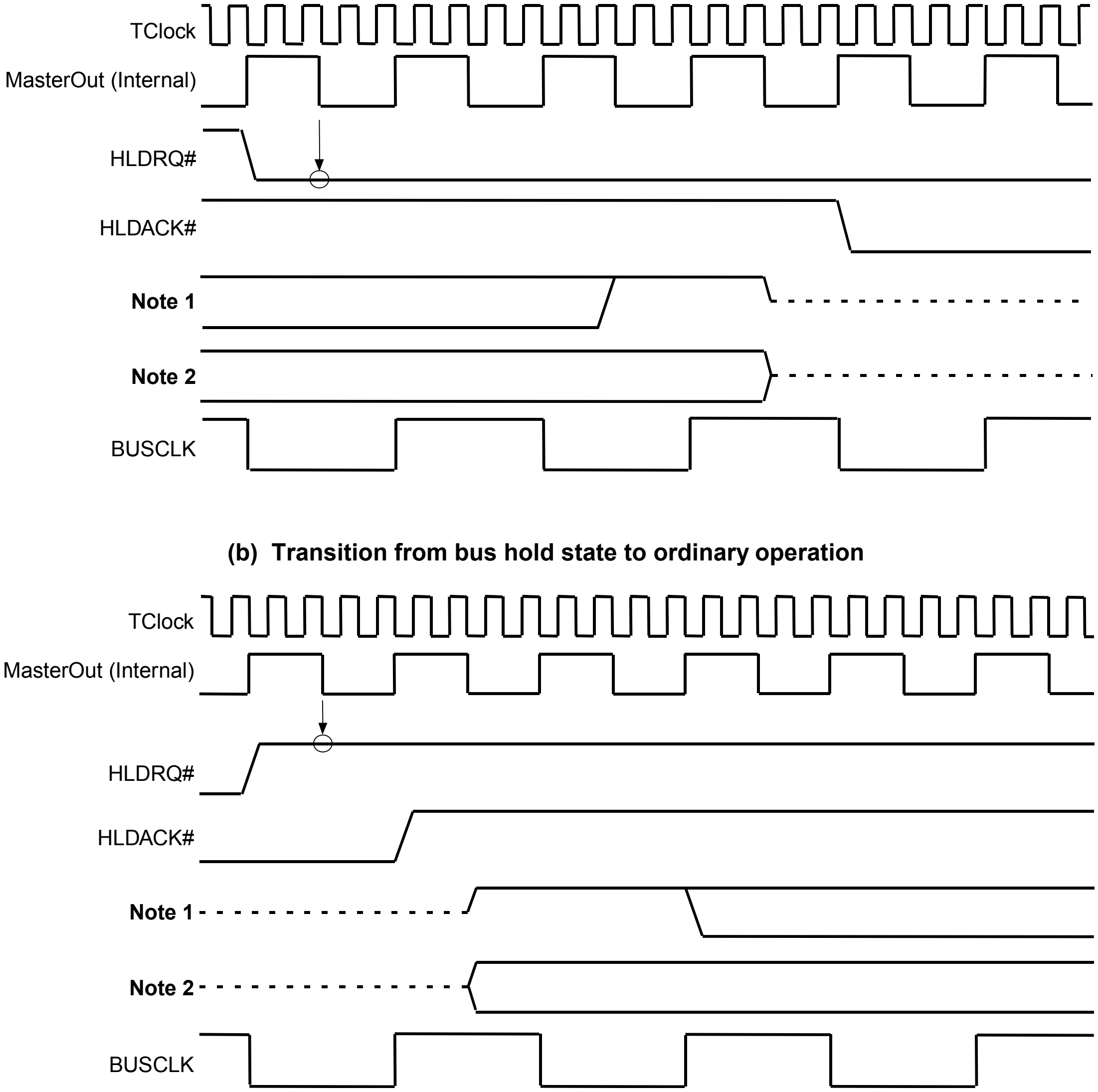


11.5.8 Bus hold

If the external bus master wants to obtain the system bus mastership during the Fullspeed/Standby/Suspend mode, it makes the HLDRQ# signal active to start the bus hold function. Once it is allowed, the VR4121 asserts the HLDACK# signal, makes system bus signals high-impedance state, then passes the bus mastership to the external bus master.

Note that the bus hold function is disabled during the Hibernate mode. Bus mastership requirement is not acknowledged in the Hibernate mode even the HLDRQ# signal is asserted.

★ **Figure 11-40. Bus Hold in Fullspeed Mode and Standby Mode (When Using EDO DRAM)**



(a) Transition from ordinary operation to bus hold state

(b) Transition from bus hold state to ordinary operation

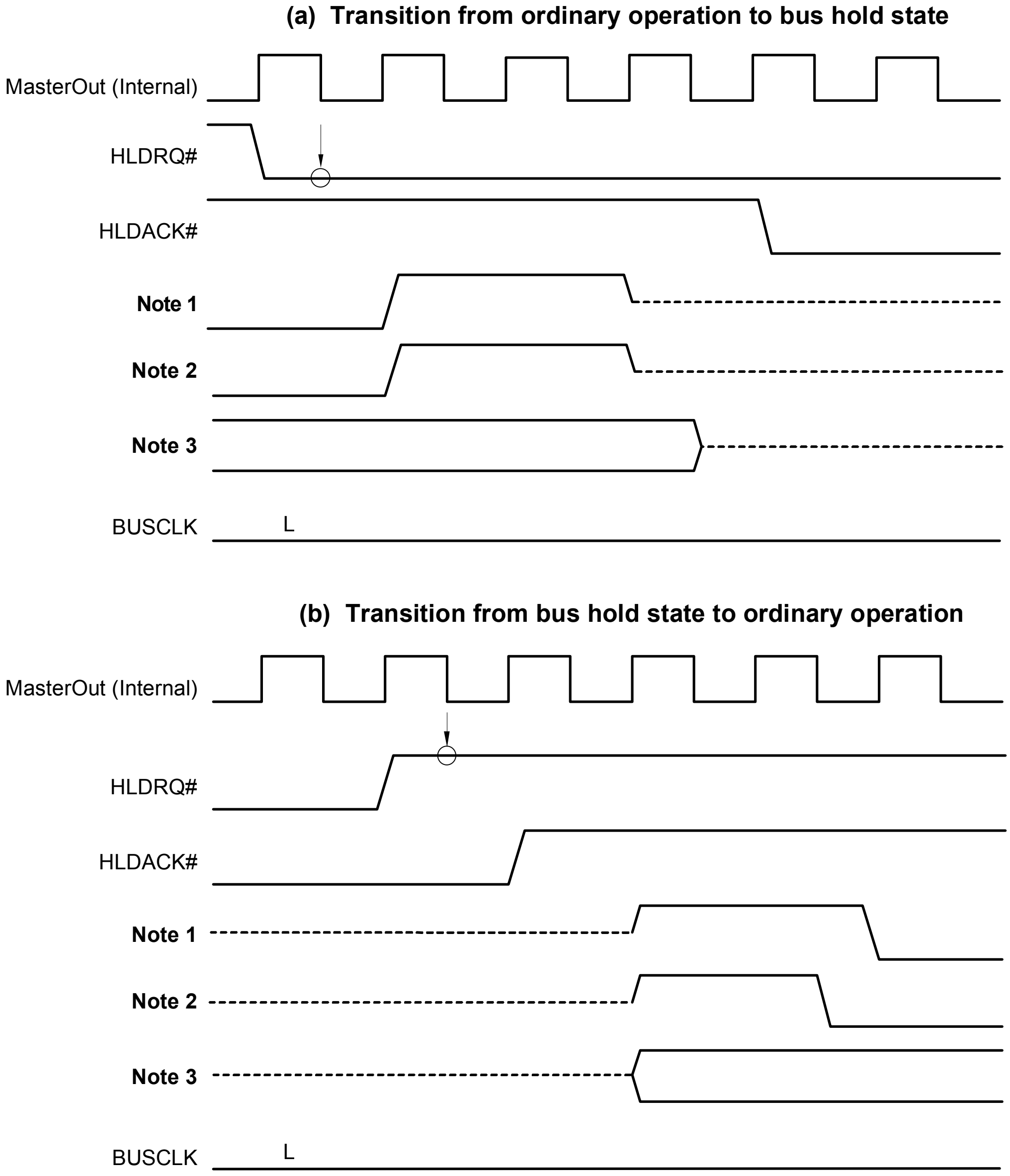
Notes 1. UUCAS#/MRAS3#, ULCAS#/MRAS2#, MRAS(1:0)#, UCAS#, LCAS#

2. SHB#, IOR#, IOW#, MEMR#, MEMW#, RD#, WR#, ADD25/SCLK, ADD(24:0), DATA(15:0), DATA(31:16)/GPIO(31:16) in 32-bit data bus mode

Remarks 1. ○ indicates sampling timing.

2. Broken lines indicate high impedance.

Figure 11-41. Bus Hold in Suspend Mode (When Using EDO DRAM)



(a) Transition from ordinary operation to bus hold state

(b) Transition from bus hold state to ordinary operation

Notes 1. UUCAS#/MRAS3#, ULCAS#/MRAS2#, MRAS(1:0)# (in 16-bit data bus mode)
MRAS(1:0)# (in 32-bit data bus mode)

2. UCAS#, LCAS# (in 16-bit data bus mode)
UUCAS#/MRAS3#, ULCAS#/MRAS2#, UCAS#, LCAS# (in 32-bit data bus mode)

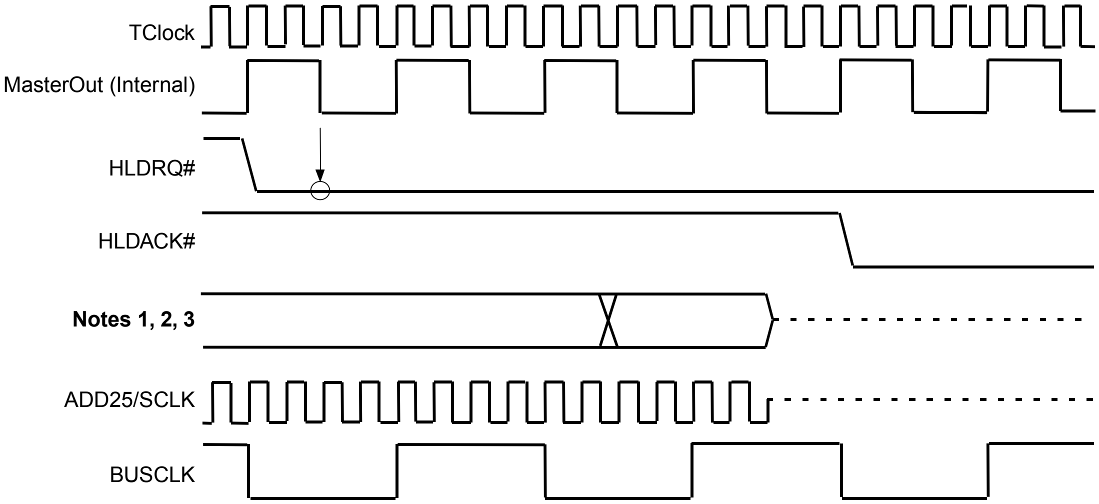
3. SHB#, IOR#, IOW#, MEMR#, MEMW#, RD#, WR#, ADD25/SCLK, ADD(24:0), DATA(15:0), DATA(31:16)/GPIO(31:16) in 32-bit data bus mode

Remarks 1. ○ indicates sampling timing.

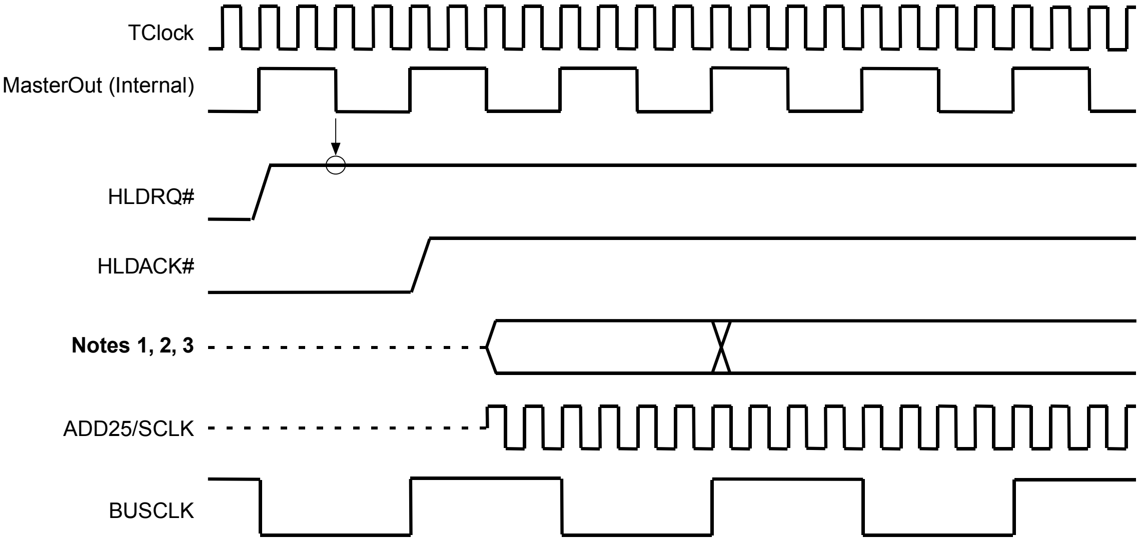
2. Broken lines indicate high impedance.

★ **Figure 11-42. Bus Hold in Fullspeed Mode and Standby Mode (When Using SDRAM)**

(a) Transition from ordinary operation to bus hold state



(b) Transition from bus hold state to ordinary operation



Notes 1. UUCAS#/MRAS3#, ULCAS#/MRAS2#, MRAS(1:0)#, UCAS#, LCAS# CKE, GPIO4/SRAS#, GPIO5/SCAS#, GPIO6/SYSDIR, SHB#, IOR#, IOW#, MEMR#, MEMW#, RD#, WR#, ADD(24:0), DATA(15:0)

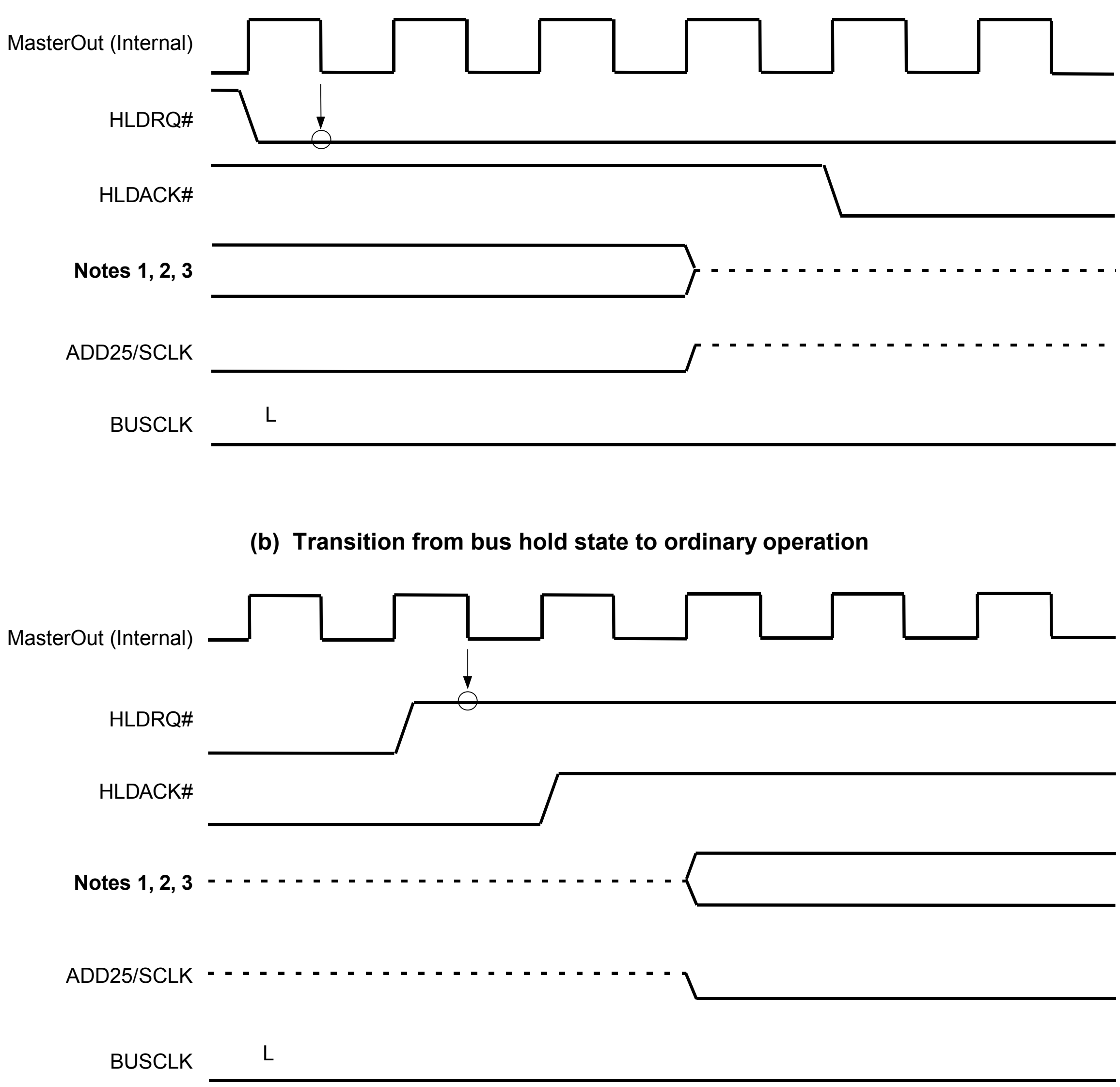
2. DATA(31:16)/GPIO(31:16) in 32-bit data bus mode

3. ROMCS(3:2)# (DRAM-expanded in 32-bit data bus mode)

Remarks 1. ○ indicates sampling timing.

2. Broken lines indicate high impedance.

★ **Figure 11-43. Bus Hold in Suspend Mode (When Using SDRAM)**



(a) Transition from ordinary operation to bus hold state

(b) Transition from bus hold state to ordinary operation

Notes 1. UUCAS#/MRAS3#, ULCAS#/MRAS2#, MRAS(1:0)#, UCAS3#, LCAS# CKE, GPIO4/SRAS#, GPIO5/SCAS#, GPIO6/SYSDIR, SHB#, IOR#, IOW#, MEMR#, MEMW#, RD#, WR#, ADD(24:0), DATA(15:0)

2. DATA(31:16)/GPIO(31:16) in 32-bit data bus mode

3. ROMCS(3:2)# (DRAM-expanded in 32-bit data bus mode)

Remarks 1. ○ indicates sampling timing.

2. Broken lines indicate high impedance.

[MEMO]

CHAPTER 12 DMAAU (DMA ADDRESS UNIT)

12.1 General

The DMAAU register controls the DMA addresses for the AIU and IrDA 4-Mbps communication module (FIR).

The DMA channel used for each unit can set a DMA start address as any half-word address in the physical address from 0x0000 0000 to 0x01FF FFFE, and is retained in DRAM as a 2-Kbyte block that starts at the address which is generated by masking the low-order 10 bits of the DMA start address.

Caution DMA operations are not guaranteed if an address overlaps with another DMA buffer.

After a DMA start address is set to the DMA base address register, the VR4121 performs DMA transfer using the registers of DMAAU as below.

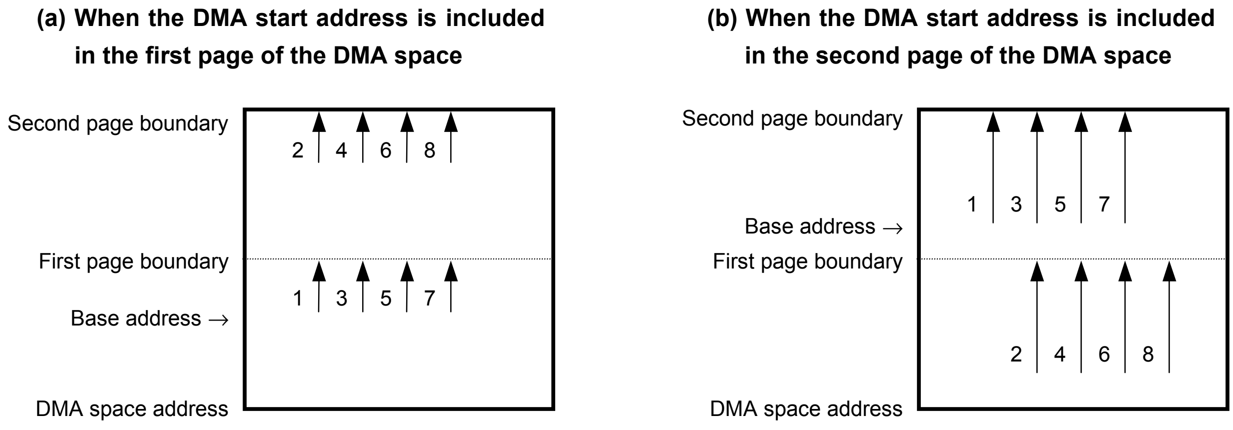
(1) When the DMA start address is included in the first page of the DMA space

1. The VR4121 starts a DMA transfer after writing the start address to the DMA address register.
2. When the DMA transfer reaches the first page boundary, the VR4121 adds 1 Kbyte to the contents of the DMA base address register, writes the value to the DMA address register, and continues the DMA transfer.
3. When the DMA transfer reaches the second page boundary, the VR4121 writes the contents of the DMA base address register to the DMA address register and continues the DMA transfer.
4. The VR4121 repeats 2. and 3. until all the data is transferred.

(2) When the DMA start address is included in the second page of the DMA space

1. The VR4121 starts a DMA transfer after writing the start address to the DMA address register.
2. When the DMA transfer reaches the second page boundary, the VR4121 subtracts 1 Kbyte from the contents of the DMA base address register, writes the value to the DMA address register, and continues the DMA transfer.
3. When the DMA transfer reaches the first page boundary, the VR4121 writes the contents of the DMA base address register to the DMA address register and continues the DMA transfer.
4. The VR4121 repeats 2. and 3. until all the data is transferred.

Figure 12-1. DMA Space Used in DMA Transfers



(a) When the DMA start address is included in the first page of the DMA space

(b) When the DMA start address is included in the second page of the DMA space

12.2 Register Set

The DMAAU registers are listed below.

Table 12-1. DMAAU Registers

Address	R/W	Register Symbols	Function
0x0B00 0020	R/W	AIUIBALREG	AIU IN DMA Base Address Register Low
0x0B00 0022	R/W	AIUIBAHREG	AIU IN DMA Base Address Register High
0x0B00 0024	R/W	AIUIALREG	AIU IN DMA Address Register Low
0x0B00 0026	R/W	AIUIAHREG	AIU IN DMA Address Register High
0x0B00 0028	R/W	AIUOBALREG	AIU OUT DMA Base Address Register Low
0x0B00 002A	R/W	AIUOBAHREG	AIU OUT DMA Base Address Register High
0x0B00 002C	R/W	AIUOALREG	AIU OUT DMA Address Register Low
0x0B00 002E	R/W	AIUOAHREG	AIU OUT DMA Address Register High
0x0B00 0030	R/W	FIRBALREG	FIR DMA Base Address Register Low
0x0B00 0032	R/W	FIRBAHREG	FIR DMA Base Address Register High
0x0B00 0034	R/W	FIRALREG	FIR DMA Address Register Low
0x0B00 0036	R/W	FIRAHREG	FIR DMA Address Register High

These registers are described in detail below.

12.2.1 AIU IN DMA base address registers

(1) AIUIBALREG (0x0B00 0020)

Bit	15	14	13	12	11	10	9	8
Name	AIUIBA15	AIUIBA14	AIUIBA13	AIUIBA12	AIUIBA11	AIUIBA10	AIUIBA9	AIUIBA8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	1	1	1	1	1	0	0	0
After reset	1	1	1	1	1	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	AIUIBA7	AIUIBA6	AIUIBA5	AIUIBA4	AIUIBA3	AIUIBA2	AIUIBA1	AIUIBA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:1	AIUIBA(15:1)	DMA base address 15:1 for AIU input
0	AIUIBA0	DMA base address 0 for AIU input Write 0 to this bit. 0 is returned after a read.

(2) AIUIBAHREG (0x0B00 0022)

Bit	15	14	13	12	11	10	9	8
Name	AIUIBA31	AIUIBA30	AIUIBA29	AIUIBA28	AIUIBA27	AIUIBA26	AIUIBA25	AIUIBA24
R/W	R	R	R	R	R	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	1
After reset	0	0	0	0	0	0	0	1

Bit	7	6	5	4	3	2	1	0
Name	AIUIBA23	AIUIBA22	AIUIBA21	AIUIBA20	AIUIBA19	AIUIBA18	AIUIBA17	AIUIBA16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	1	1	1	1	1	1	1	1
After reset	1	1	1	1	1	1	1	1

Bit	Name	Function
15:11	AIUIBA(31:27)	DMA base address 31:27 for AIU input Write 0 to these bits. 0 is returned after a read.
10:0	AIUIBA(26:16)	DMA base address 26:16 for AIU input

AIUIBALREG and AIUIBAHREG are used to set the base addresses for the DMA channel used for audio input (recording).

The addresses set to this register become DMA transfer start addresses.

The DMA channel used for audio input is retained in DRAM as a 2-Kbyte buffer that starts at the address which is generated by masking the lower 10 bits of the DMA start address.

12.2.2 AIU IN DMA address registers

(1) AIUIALREG (0x0B00 0024)

Bit	15	14	13	12	11	10	9	8
Name	AIUIA15	AIUIA14	AIUIA13	AIUIA12	AIUIA11	AIUIA10	AIUIA9	AIUIA8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	1	1	1	1	1	0	0	0
After reset	1	1	1	1	1	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	AIUIA7	AIUIA6	AIUIA5	AIUIA4	AIUIA3	AIUIA2	AIUIA1	AIUIA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:0	AIUIA(15:0)	Next DMA address 15:0 to be accessed for AIU input channel

(2) AIUIAHREG (0x0B00 0026)

Bit	15	14	13	12	11	10	9	8
Name	AIUIA31	AIUIA30	AIUIA29	AIUIA28	AIUIA27	AIUIA26	AIUIA25	AIUIA24
R/W	R	R	R	R	R	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	1
After reset	0	0	0	0	0	0	0	1

Bit	7	6	5	4	3	2	1	0
Name	AIUIA23	AIUIA22	AIUIA21	AIUIA20	AIUIA19	AIUIA18	AIUIA17	AIUIA16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	1	1	1	1	1	1	1	1
After reset	1	1	1	1	1	1	1	1

Bit	Name	Function
15:0	AIUIA(31:16)	Next DMA address 31:16 to be accessed for AIU input channel

12.2.3 AIU OUT DMA base address registers

(1) AIUOBALREG (0x0B00 0028)

Bit	15	14	13	12	11	10	9	8
Name	AIUOBA15	AIUOBA14	AIUOBA13	AIUOBA12	AIUOBA11	AIUOBA10	AIUOBA9	AIUOBA8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	1	1	1	1	1	0	0	0
After reset	1	1	1	1	1	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	AIUOBA7	AIUOBA6	AIUOBA5	AIUOBA4	AIUOBA3	AIUOBA2	AIUOBA1	AIUOBA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:1	AIUOBA(15:1)	DMA base address 15:1 for AIU output
0	AIUOBA0	DMA base address 0 for AIU output Write 0 to this bit. 0 is returned after a read.

(2) AIUOBAHREG (0x0B00 002A)

Bit	15	14	13	12	11	10	9	8
Name	AIUOBA31	AIUOBA30	AIUOBA29	AIUOBA28	AIUOBA27	AIUOBA26	AIUOBA25	AIUOBA24
R/W	R	R	R	R	R	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	1
After reset	0	0	0	0	0	0	0	1

Bit	7	6	5	4	3	2	1	0
Name	AIUOBA23	AIUOBA22	AIUOBA21	AIUOBA20	AIUOBA19	AIUOBA18	AIUOBA17	AIUOBA16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	1	1	1	1	1	1	1	1
After reset	1	1	1	1	1	1	1	1

Bit	Name	Function
15:11	AIUOBA(31:27)	DMA base address 31:27 for AIU output Write 0 to these bits. 0 is returned after a read.
10:0	AIUOBA(26:16)	DMA base address 26:16 for AIU output

AIUOBALREG and AIUOBAHREG are used to set the base addresses for the DMA channel used for audio output (playback).

The addresses set to this register become DMA transfer start addresses.

The DMA channel used for audio output is retained in DRAM as a 2-Kbyte buffer that starts at the address which is generated by masking the lower 10 bits of the DMA start address.

12.2.4 AIU OUT DMA address registers

(1) AIUOALREG (0x0B00 002C)

Bit	15	14	13	12	11	10	9	8
Name	AIUOA15	AIUOA14	AIUOA13	AIUOA12	AIUOA11	AIUOA10	AIUOA9	AIUOA8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	1	1	1	1	1	0	0	0
After reset	1	1	1	1	1	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	AIUOA7	AIUOA6	AIUOA5	AIUOA4	AIUOA3	AIUOA2	AIUOA1	AIUOA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:0	AIUOA(15:0)	Next DMA address 15:0 to be accessed for AIU output channel

(2) AIUOAHREG (0x0B00 002E)

Bit	15	14	13	12	11	10	9	8
Name	AIUOA31	AIUOA30	AIUOA29	AIUOA28	AIUOA27	AIUOA26	AIUOA25	AIUOA24
R/W	R	R	R	R	R	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	1
After reset	0	0	0	0	0	0	0	1

Bit	7	6	5	4	3	2	1	0
Name	AIUOA23	AIUOA22	AIUOA21	AIUOA20	AIUOA19	AIUOA18	AIUOA17	AIUOA16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	1	1	1	1	1	1	1	1
After reset	1	1	1	1	1	1	1	1

Bit	Name	Function
15:0	AIUOA(31:16)	Next DMA address 31:16 to be accessed for AIU output channel

12.2.5 FIR DMA base address registers

(1) FIRBALREG (0x0B00 0030)

Bit	15	14	13	12	11	10	9	8
Name	FIRBA15	FIRBA14	FIRBA13	FIRBA12	FIRBA11	FIRBA10	FIRBA9	FIRBA8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	1	1	1	1	1	0	0	0
After reset	1	1	1	1	1	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	FIRBA7	FIRBA6	FIRBA5	FIRBA4	FIRBA3	FIRBA2	FIRBA1	FIRBA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:0	FIRBA(15:0)	FIR DMA base address 15:0

(2) FIRBAHREG (0x0B00 0032)

Bit	15	14	13	12	11	10	9	8
Name	FIRBA31	FIRBA30	FIRBA29	FIRBA28	FIRBA27	FIRBA26	FIRBA25	FIRBA24
R/W	R	R	R	R	R	R	R/W	R/W
RTCRST	0	0	0	0	0	0	0	1
After reset	0	0	0	0	0	0	0	1

Bit	7	6	5	4	3	2	1	0
Name	FIRBA23	FIRBA22	FIRBA21	FIRBA20	FIRBA19	FIRBA18	FIRBA17	FIRBA16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	1	1	1	1	1	1	1	1
After reset	1	1	1	1	1	1	1	1

Bit	Name	Function
15:11	FIRBA(31:27)	FIR DMA base address 31:27 Write 0 to these bits. 0 is returned after a read.
10:0	FIRBA(26:16)	FIR DMA base address 26:16

FIRBALREG and FIRBAHREG are used to set the base addresses for the FIR DMA channel.

The addresses set to this register become DMA transfer start addresses.

The FIR DMA channel is retained in DRAM as a 2-Kbyte buffer that starts at the address that is generated by masking the lower 10 bits of the DMA start address.

12.2.6 FIR DMA address registers

(1) FIRALREG (0x0B00 0034)

Bit	15	14	13	12	11	10	9	8
Name	FIRA15	FIRA14	FIRA13	FIRA12	FIRA11	FIRA10	FIRA9	FIRA8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	1	1	1	1	1	0	0	0
After reset	1	1	1	1	1	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	FIRA7	FIRA6	FIRA5	FIRA4	FIRA3	FIRA2	FIRA1	FIRA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:0	FIRA(15:0)	Next DMA address 15:0 to be accessed by FIR channel

(2) FIRAHREG (0x0B00 0036)

Bit	15	14	13	12	11	10	9	8
Name	FIRA31	FIRA30	FIRA29	FIRA28	FIRA27	FIRA26	FIRA25	FIRA24
R/W	R	R	R	R	R	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	1
After reset	0	0	0	0	0	0	0	1

Bit	7	6	5	4	3	2	1	0
Name	FIRA23	FIRA22	FIRA21	FIRA20	FIRA19	FIRA18	FIRA17	FIRA16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	1	1	1	1	1	1	1	1
After reset	1	1	1	1	1	1	1	1

Bit	Name	Function
15:0	FIRA(31:16)	Next DMA address 31:16 to be accessed by FIR channel

CHAPTER 13 DCU (DMA CONTROL UNIT)

13.1 General

The DCU register is used for DMA control. Specifically, it controls acknowledgment from the BCU that handles bus arbitration and DMA requests from the on-chip peripheral I/O units (AIU and FIR). It also controls DMA enable/prohibit settings.

13.2 DMA Priority Control

When a conflict occurs between DMA requests sent from on-chip peripheral I/O units, the following priority levels are used to resolve the conflict. These priority levels cannot be changed.

Table 13-1. DMA Priority Levels

Priority Level	Type of DMA Operation
High	Audio input (recording)
↑	Audio output (playback)
Low	FIR transmission/reception

When a conflict occurs among a DMA request from on-chip peripheral I/O units, a DRAM refresh request, and a bus hold request, the processing is executed in the order of DRAM refresh, bus hold, and DMA. When a DRAM refresh request occurs during a DMA transfer, DRAM refresh is performed after one DMA transfer has been completed.

13.3 Register Set

The DCU register set is described below.

Table 13-2. DCU Registers

Address	R/W	Register Symbols	Function
0x0B00 0040	R/W	DMARSTREG	DMA Reset Register
0x0B00 0042	R	DMAIDLEREG	DMA Idle Register
0x0B00 0044	R/W	DMASENREG	DMA Sequencer Enable Register
0x0B00 0046	R/W	DMAMSKREG	DMA Mask Register
0x0B00 0048	R	DMAREQREG	DMA Request Register
0x0B00 004A	R/W	TDREG	Transfer Direction Register

These registers are described in detail below.

13.3.1 DMARSTREG (0x0B00 0040)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	DMARST
R/W	R	R	R	R	R	R	R	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:1	RFU	Write 0 to these bits. 0 is returned after a read.
0	DMARST	Reset DMA controller 0: Reset 1: Normal

This register is used to reset the DMA controller.

13.3.2 DMAIDLEREG (0x0B00 0042)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	DMAISTAT
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:1	RFU	Write 0 to these bits. 0 is returned after a read.
0	DMAISTAT	Display DMA sequencer status 1: Idle status 0: Sequencer busy

This register is used to display the DMA sequencer status.

13.3.3 DMASENREG (0x0B00 0044)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	DMASEN
R/W	R	R	R	R	R	R	R	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:1	RFU	Write 0 to these bits. 0 is returned after a read.
0	DMASEN	Enable DMA sequencer 1: Enable 0: Prohibit

This register is used to enable/prohibit the DMA sequencer.

13.3.4 DMAMSKREG (0x0B00 0046)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	DMAMSK AIN	DMAMSK AOUT	RFU	DMAMSK FOUT
R/W	R	R	R	R	R/W	R/W	R	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:4	RFU	Write 0 to these bits. 0 is returned after a read.
3	DMAMSKAIN	Audio input DMA transfer enable/prohibit 1: Enable 0: Prohibit
2	DMAMSKAOUT	Audio output DMA transfer enable/prohibit 1: Enable 0: Prohibit
1	RFU	Write 0 to this bit. 0 is returned after a read.
0	DMAMSKFOUT	FIR transmission DMA transfer enable/prohibit 1: Enable 0: Prohibit

This register is used to enable/prohibit various types of DMA transfers.

The DMA transfer enable bits should be set when the units that receive DMA service have been stopped or when there are no pending DMA requests. If any of the above bits are set to a unit while a DMA request is pending for that unit, the CPU's operation will be undefined.

13.3.5 DMAREQREG (0x0B00 0048)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	DRQAIN	DRQAOUT	RFU	DRQFOUT
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:4	RFU	Write 0 to these bits. 0 is returned after a read.
3	DRQAIN	Audio input DMA transfer request 1: Request pending 0: No request
2	DRQAOUT	Audio output DMA transfer request 1: Request pending 0: No request
1	RFU	Write 0 to this bit. 0 is returned after a read.
0	DRQFOUT	FIR transmission DMA transfer request 1: Request pending 0: No request

This register is used to indicate whether or not there are any DMA transfer requests.

13.3.6 TDREG (0x0B00 004A)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	FIR
R/W	R	R	R	R	R	R	R	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:1	RFU	Write 0 to these bits. 0 is returned after a read.
0	FIR	Transfer direction of DMA channel for FIR transmission 1: I/O → MEM 0: MEM → I/O

This register is used to set the data transfer direction of DMA channel for FIR transmission.

[MEMO]

CHAPTER 14 CMU (CLOCK MASK UNIT)

14.1 General

As various input clocks (ctclock, i_seclk, firclock) are supplied from the CPU to each unit, a masking method enables power consumption to be curtailed in units that are not used.

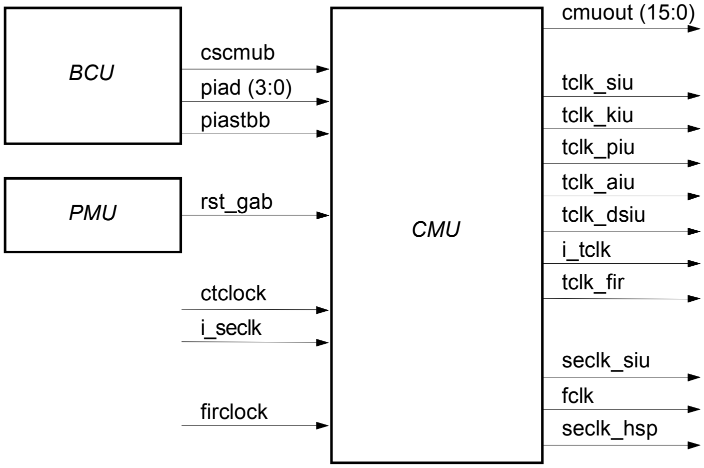
The units for which this masking method are used are the KIU, PIU, AIU, SIU, DSIU, FIR, and HSP (software modem interface) units.

The basic functions are described below.

1. Control of TClock supplied to PIU, AIU, SIU, KIU, DSIU, and FIR
2. Control of internal clock (18.432 MHz) supplied to SIU and HSP
3. Control of internal clock (48 MHz) supplied to FIR

The initial value is "0", which specifies masking all supplied clocks. No clock is supplied unless the CPU writes "1" to CMUCLKMSK register.

Figure 14-1. CMU Peripheral Block Diagram



14.2 Register Set

The CMU register is listed below.

Table 14-1. CMU Register

Address	R/W	Register Symbol	Function
0x0B00 0060	R/W	CMUCLKMSK	CMU Clock Mask Register

This register is described in detail below.

14.2.1 CMUCLKMSK (0x0B00 0060)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	MSKFFIR	MSKSHSP	MSKSSIU
R/W	R	R	R	R	R	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	MSKDSIU	MSKFIR	MSKKIU	MSKAIU	MSKSIU	MSKPIU
R/W	R	R	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:11	RFU	Write 0 to these bits. 0 is returned after a read.
10	MSKFFIR	Supply/mask 48-MHz clock to FIR unit 1: Supply 0: Mask
9	MSKSHSP	Supply/mask 18.432-MHz clock to HSP unit 1: Supply 0: Mask
8	MSKSSIU	Supply/mask 18.432-MHz clock to SIU unit 1: Supply 0: Mask
7:6	RFU	Write 0 to these bits. 0 is returned after a read.
5	MSKDSIU	Supply/mask TClock to DSIU unit 1: Supply 0: Mask
4	MSKFIR	Supply/mask TClock to FIR unit 1: Supply 0: Mask
3	MSKKIU	Supply/mask TClock to KIU unit 1: Supply 0: Mask
2	MSKAIU	Supply/mask TClock to AIU unit 1: Supply 0: Mask
1	MSKSIU	Supply/mask TClock to SIU unit 1: Supply 0: Mask
0	MSKPIU	Supply/mask TClock to PIU unit 1: Supply 0: Mask

This register is used to mask the clocks that are supplied to the KIU, PIU, AIU, SIU, DSIU, FIR, and HSP units.

[MEMO]

CHAPTER 15 ICU (INTERRUPT CONTROL UNIT)

15.1 General

The ICU collects interrupt requests from the various on-chip peripheral units and transfers these interrupt request signals (Int0, Int1, Int2, Int3, and NMI) to the CPU core.

The functions of the ICU's internal blocks are briefly described below.

- ADDECICU … Decodes read/write addresses from the CPU that are used for ICU registers.

- REGICU … This includes a register for interrupt masking. The initial value is "0", which specifies masking. No interrupt signal is supplied to CPU core unless the CPU writes "1" to this register.

- OUTICU … This block collects interrupt requests after masking them, and generates an interrupt request signal to output to the CPU core.
  During Suspend mode, it also controls the masking of interrupt requests, output of the int_all signal that is the general interrupt source signal, and the memdrv output timing signal that is used when returning from Suspend mode.

To acknowledge an interrupt requests to the CPU core, the following signals are used:

NMI: battint_intr only
Switching between NMI and Int0 is enabled according to this register's settings.
Because NMI's interrupt masking cannot be controlled by means of software, switch to Int0 to mask battint_Intr.

Int3: hsp_intr only

Int2: rtc_long2_intr only

Int1: rtc_long1_intr only
The IT (interval timer) and HSP interrupts require more responsiveness than do other interrupt sources.

Int0: All other interrupts
For details of the interrupt sources, see **15.2 Register Set**.

How an interrupt request is notified to the CPU core is shown below.

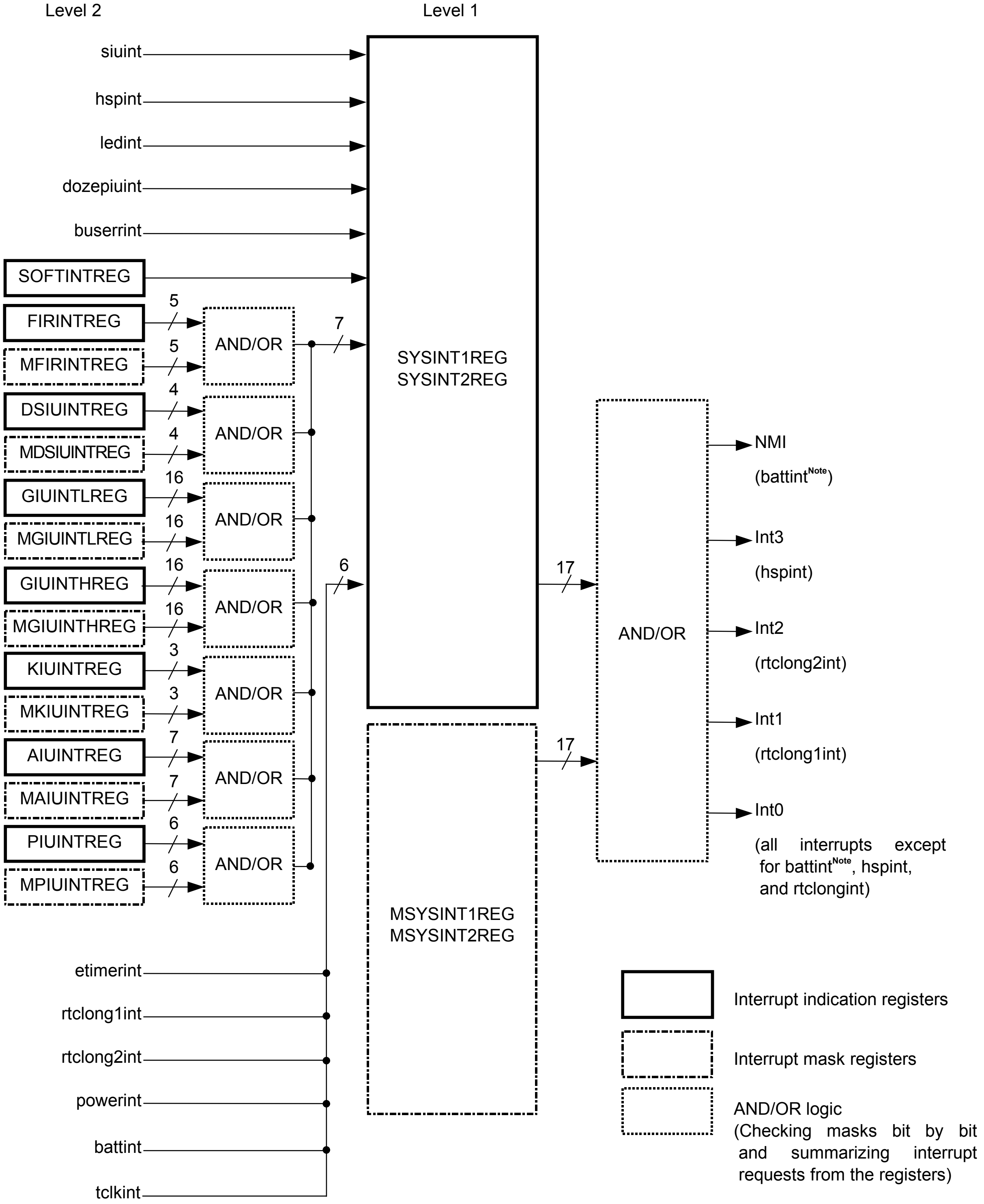
If an interrupt request occurs in the peripheral units, the corresponding bit in the interrupt indication register of Level 2 (xxxINTREG) is set to 1. The interrupt indication register is ANDed bit-wise with the corresponding interrupt mask register of Level 2 (MxxxINTREG). If the occurred interrupt request is enabled (set to 1) in the mask register, the interrupt request is notified to the interrupt indication register of Level 1 (SYSINTREG) and the corresponding bit is set to 1. At this time, the interrupt requests from the same register of Level 2 are notified to the SYSINTREG as a single interrupt request.

Interrupt requests from some units directly set their corresponding bits in the SYSINTREG.

The SYSINTREG is ANDed bit-wise with the interrupt mask register of Level 1 (MSYSINTREG). If the interrupt request is enabled by MSYSINTREG (set to 1), a corresponding interrupt request signal is output from the ICU to the CPU core. battint is connected to the NMI or Int0 signal of the CPU core (selected by setting of NMIREG). rtc_long signals are connected to the Int3 signal of the CPU core. The other interrupt requests are connected to the Int0 signal of the CPU core as a one interrupt request.

The following figure shows an outline of interrupt control in the ICU.

Figure 15-1. Interrupt Control Outline



Note Which of NMI or Int0 is used for battint is selected by setting NMIREG.
When selecting the NMI use, This interrupt cannot be masked in the CPU core.
Mask is set by MSYSINT1REG in ICU.

15.2 Register Set

The ICU registers are listed below.

Table 15-1. ICU Registers

Address	R/W	Register Symbols	Function
0x0B00 0080	R	SYSINT1REG	Level 1 System interrupt register 1
0x0B00 0082	R	PIUINTREG	Level 2 PIU interrupt register
0x0B00 0084	R	AIUINTREG	Level 2 AIU interrupt register
0x0B00 0086	R	KIUINTREG	Level 2 KIU interrupt register
0x0B00 0088	R	GIUINTLREG	Level 2 GIU interrupt register Low
0x0B00 008A	R	DSIUINTREG	Level 2 DSIU interrupt register
0x0B00 008C	R/W	MSYSINT1REG	Level 1 mask system interrupt register 1
0x0B00 008E	R/W	MPIUINTREG	Level 2 mask PIU interrupt register
0x0B00 0090	R/W	MAIUINTREG	Level 2 mask AIU interrupt register
0x0B00 0092	R/W	MKIUINTREG	Level 2 mask KIU interrupt register
0x0B00 0094	R/W	MGIUINTLREG	Level 2 mask GIU interrupt register Low
0x0B00 0096	R/W	MDSIUINTREG	Level 2 mask DSIU interrupt register
0x0B00 0098	R/W	NMIREG	NMI register
0x0B00 009A	R/W	SOFTINTREG	Software interrupt register
0x0B00 0200	R	SYSINT2REG	Level 1 System interrupt register 2
0x0B00 0202	R	GIUINTHREG	Level 2 GIU interrupt register High
0x0B00 0204	R	FIRINTREG	Level 2 FIR interrupt register
0x0B00 0206	R/W	MSYSINT2REG	Level 1 mask system interrupt register 2
0x0B00 0208	R/W	MGIUINTHREG	Level 2 mask GIU interrupt register High
0x0B00 020A	R/W	MFIRINTREG	Level 2 mask FIR interrupt register

These registers are described in detail below.

15.2.1 SYSINT1REG (0x0B00 0080)

(1/2)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	DOZE PIUINTR	RFU	SOFT INTR	WRBERR INTR	SIUINTR	GIUINTR
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	KIUINTR	AIUINTR	PIUINTR	RFU	ETIMER INTR	RTCL1 INTR	POWER INTR	BATINTR
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:14	RFU	Write 0 to these bits. 0 is returned after a read.
13	DOZEPIUINTR	PIU interrupt during Suspend mode 1: Occurred 0: Normal
12	RFU	Write 0 to this bit. 0 is returned after a read.
11	SOFTINTR	Software interrupt (occurs by setting the SOFTINTREG) 1: Occurred 0: Normal
10	WRBERRINTR	Bus error interrupt 1: Occurred 0: Normal
9	SIUINTR	SIU interrupt 1: Occurred 0: Normal
8	GIUINTR	GIU (lower) interrupt 1: Occurred 0: Normal
7	KIUINTR	KIU interrupt 1: Occurred 0: Normal
6	AIUINTR	AIU interrupt 1: Occurred 0: Normal

(2/2)

Bit	Name	Function
5	PIUINTR	PIU interrupt 1: Occurred 0: Normal
4	RFU	Write 0 to this bit. 0 is returned after a read.
3	ETIMERINTR	ETIMER interrupt 1: Occurred 0: Normal
2	RTCL1INTR	RTCLong1 interrupt 1: Occurred 0: Normal
1	POWERINTR	PowerSW interrupt 1: Occurred 0: Normal
0	BATINTR	Battery interrupt 1: Occurred 0: Normal

This register indicates when various interrupts occur in the VR4121 system.

15.2.2 PIUINTREG (0x0B00 0082)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	PADCMD INTR	PADADP INTR	PADPAGE1 INTR	PADPAGE0 INTR	PADDLO STINTR	RFU	PENCHG INTR
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:7	RFU	Write 0 to these bits. 0 is returned after a read.
6	PADCMDINTR	PIU command scan interrupt. This interrupt occurs when command scan found valid data. 1: Occurred 0: Normal
5	PADADPINTR	PIU AD port scan interrupt. This interrupt occurs when AD port scan found a set of valid data. 1: Occurred 0: Normal
4	PADPAGE1INTR	PIU data buffer page 1 interrupt. This interrupt occurs when a set of valid data is stored in page 1 of data buffer. 1: Occurred 0: Normal
3	PADPAGE0INTR	PIU data buffer page 0 interrupt. This interrupt occurs when a set of valid data is stored in page 0 of data buffer. 1: Occurred 0: Normal
2	PADDLOSTINTR	A/D data timeout interrupt. This interrupt occurs when a set of data did not found within specified time. 1: Occurred 0: Normal
1	RFU	Write 0 to this bit. 0 is returned after a read.
0	PENCHGINTR	Touch panel contact status change interrupt 1: Change has occurred 0: No change

This register indicates when various PIU-related interrupts occur.

15.2.3 AIUINTREG (0x0B00 0084)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	INTMEND	INTM	INTMIDLE	INTMST
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	INTSEND	INTS	INTSIDLE	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:12	RFU	Write 0 to these bits. 0 is returned after a read.
11	INTMEND	Audio input (MIC) DMA buffer 2 page interrupt 1: Occurred 0: Normal
10	INTM	Audio input (MIC) DMA buffer 1 page interrupt 1: Occurred 0: Normal
9	INTMIDLE	Audio input (MIC) idle interrupt (received data is lost). This interrupt occurs if valid data exists in MIDATREG when data was received from A/D converter. 1: Occurred 0: Normal
8	INTMST	Audio input (MIC) receive completion interrupt. This interrupt occurs when 10-bit converted data was received from the A/D converter. 1: Occurred 0: Normal
7:4	RFU	Write 0 to these bits. 0 is returned after a read
3	INTSEND	Audio output (speaker) DMA buffer 2 page interrupt 1: Occurred 0: Normal
2	INTS	Audio output (speaker) DMA buffer 1 page interrupt 1: Occurred 0: Normal
1	INTSIDLE	Audio output (speaker) idle interrupt (mute). This interrupt occurs if there is no valid data in SODATREG when data was transferred to D/A. 1: Occurred 0: Normal
0	RFU	Write 0 to this bit. 0 is returned after a read

This register indicates when various AIU-related interrupts occur.

15.2.4 KIUINTREG (0x0B00 0086)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	KDATLOST	KDATRDY	SCANINT
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:3	RFU	Write 0 to these bits. 0 is returned after a read.
2	KDATLOST	Key scan data lost interrupt 1: Occurred 0: Normal
1	KDATRDY	Key scan data complete interrupt 1: Occurred 0: Normal
0	SCANINT	Key input detect interrupt 1: Occurred 0: Normal

This register indicates when various KIU-related interrupts occur.

15.2.5 GIUINTLREG (0x0B00 0088)

Bit	15	14	13	12	11	10	9	8
Name	INTS15	INTS14	INTS13	INTS12	INTS11	INTS10	INTS9	INTS8
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	INTS7	INTS6	INTS5	INTS4	INTS3	INTS2	INTS1	INTS0
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:0	INTS(15:0)	Interrupt to GPIO(15:0) pin 1: Occurred 0: Normal

This register indicates when various GIU-related interrupts occur.

15.2.6 DSIUINTREG (0x0B00 008A)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	INTDCTS	INTSER0	INTSR0	INTST0
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	1
After reset	0	0	0	0	0	0	0	1

Bit	Name	Function
15:12	RFU	Write 0 to these bits. 0 is returned after a read.
11	INTDCTS	DCTS# change interrupt 1: Occurred 0: Normal
10	INTSER0	Debug serial receive error interrupt 1: Occurred 0: Normal
9	INTSR0	Debug serial receive complete interrupt 1: Occurred 0: Normal
8	INTST0	Debug serial transmit complete interrupt 1: Occurred 0: Normal
7:1	RFU	Write 0 to these bits. 0 is returned after a read.
0	RFU	Write 1 to this bit. 1 is returned after a read.

This register indicates when various DSIU-related interrupts occur.

15.2.7 MSYSINT1REG (0x0B00 008C)

(1/2)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	DOZE PIUINTR	RFU	SOFTINTR	WRBERR INTR	SIUINTR	GIUINTR
R/W	R	R	R/W	R	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	KIUINTR	AIUINTR	PIUINTR	RFU	ETIMER INTR	RTCL1 INTR	POWER INTR	BATINTR
R/W	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:14	RFU	Write 0 to these bits. 0 is returned after a read.
13	DOZEPIUINTR	PIU interrupt enable during suspend mode 1: Enable 0: Prohibit
12	RFU	Write 0 to this bit. 0 is returned after a read.
11	SOFTINTR	Software interrupt (occurs by setting the SOFTINTREG) enable 1: Enable 0: Prohibit
10	WRBERRINTR	Bus error interrupt enable 1: Enable 0: Prohibit
9	SIUINTR	SIU interrupt enable 1: Enable 0: Prohibit
8	GIUINTR	GIU (lower) interrupt enable 1: Enable 0: Prohibit
7	KIUINTR	KIU interrupt enable 1: Enable 0: Prohibit
6	AIUINTR	AIU interrupt enable 1: Enable 0: Prohibit

(2/2)

Bit	Name	Function
5	PIUINTR	PIU interrupt enable 1: Enable 0: Prohibit
4	RFU	Write 0 when writing. 0 is returned after a read.
3	ETIMERINTR	ETIMER interrupt enable 1: Enable 0: Prohibit
2	RTCL1INTR	RTCLong1 timer interrupt enable 1: Enable 0: Prohibit
1	POWERINTR	PowerSW interrupt enable 1: Enable 0: Prohibit
0	BATINTR	Battery interrupt enable 1: Enable 0: Prohibit

This register is used to mask various interrupts that occur in the VR4121 system.

15.2.8 MPIUINTREG (0x0B00 008E)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	PADCMD INTR	PADADP INTR	PADPAGE1 INTR	PADPAGE0 INTR	PADDLO STINTR	RFU	PENCHG INTR
R/W	R	R/W	R/W	R/W	R/W	R/W	R	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:7	RFU	Write 0 to these bits. 0 is returned after a read.
6	PADCMDINTR	PIU command scan interrupt enable 1: Enable 0: Prohibit
5	PADADPINTR	PIU A/D port scan interrupt enable 1: Enable 0: Prohibit
4	PADPAGE1INTR	PIU data buffer page 1 interrupt enable 1: Enable 0: Prohibit
3	PADPAGE0INTR	PIU data buffer page 0 interrupt enable 1: Enable 0: Prohibit
2	PADDLOSTINTR	A/D data timeout interrupt enable 1: Enable 0: Prohibit
1	RFU	Write 0 to this bit. 0 is returned after a read.
0	PENCHGINTR	Touch panel contact status change interrupt enable 1: Enable 0: Prohibit

This register is used to mask various PIU-related interrupts.

15.2.9 MAIUINTREG (0x0B00 0090)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	INTMEND	INTM	INTMIDLE	INTMST
R/W	R	R	R	R	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	INTSEND	INTS	INTSIDLE	RFU
R/W	R	R	R	R	R/W	R/W	R/W	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:12	RFU	Write 0 to these bits. 0 is returned after a read.
11	INTMEND	Audio input (MIC) DMA buffer 2 page interrupt enable 1: Enable 0: Prohibit
10	INTM	Audio input (MIC) DMA buffer 1 page interrupt enable 1: Enable 0: Prohibit
9	INTMIDLE	Audio input (MIC) idle interrupt (received data is lost) enable 1: Enable 0: Prohibit
8	INTMST	Audio input (MIC) receive complete interrupt 1: Enable 0: Prohibit
7:4	RFU	Write 0 to these bits. 0 is returned after a read.
3	INTSEND	Audio output (speaker) DMA buffer 2 page interrupt enable 1: Enable 0: Prohibit
2	INTS	Audio output (speaker) DMA buffer 1 page interrupt enable 1: Enable 0: Prohibit
1	INTSIDLE	Audio output (speaker) idle interrupt (mute) enable 1: Enable 0: Prohibit
0	RFU	Write 0 to this bit. 0 is returned after a read.

This register is used to mask various AIU-related interrupts.

15.2.10 MKIUINTREG (0x0B00 0092)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	KDAT LOST	KDAT RDY	SCAN INT
R/W	R	R	R	R	R	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:3	RFU	Write 0 to these bits. 0 is returned after a read.
2	KDATLOST	Key data scan lost interrupt enable 1: Enable 0: Prohibit
1	KDATRDY	Key scan data complete interrupt enable 1: Enable 0: Prohibit
0	SCANINT	Key input detect interrupt enable 1: Enable 0: Prohibit

This register is used to mask various KIU-related interrupts.

15.2.11 MGIUINTLREG (0x0B00 0094)

Bit	15	14	13	12	11	10	9	8
Name	INTS15	INTS14	INTS13	INTS12	INTS11	INTS10	INTS9	INTS8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	INTS7	INTS6	INTS5	INTS4	INTS3	INTS2	INTS1	INTS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:0	INTS(15:0)	GPIO(15:0) pin interrupt enable 1: Enable 0: Prohibit

This register is used to mask various GIU-related interrupts.

15.2.12 MDSIUINTREG (0x0B00 0096)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	INTDCTS	INTSER0	INTSR0	INTST0
R/W	R	R	R	R	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:12	RFU	Write 0 to these bits. 0 is returned after a read.
11	INTDCTS	DCTS# change interrupt enable 1: Enable 0: Prohibit
10	INTSER0	Debug serial data receive error interrupt enable 1: Enable 0: Prohibit
9	INTSR0	Debug serial data receive complete interrupt enable 1: Enable 0: Prohibit
8	INTST0	Debug serial data transmit complete interrupt enable 1: Enable 0: Prohibit
7:0	RFU	Write 0 to these bits. 0 is returned after a read.

This register is used to mask various DSIU-related interrupts.

15.2.13 NMIREG (0x0B00 0098)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	NMIOR INT
R/W	R	R	R	R	R	R	R	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:1	RFU	Write 0 to these bits. 0 is returned after a read.
0	NMIORINT	Low battery detect interrupt type setting 1: Int0 0: NMI

This register is used to set the type of interrupt used to notify the VR4120 CPU core when a low battery detect interrupt has occurred.

15.2.14 SOFTINTREG (0x0B00 009A)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	SOFT INTR3	SOFT INTR2	SOFT INTR1	SOFT INTR0
R/W	R	R	R	R	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:4	RFU	Write 0 to these bits. 0 is returned after a read.
3:0	SOFTINTR(3:0)	Set/clear software interrupt 1: Set 0: Clear

This register is used to set software interrupts. Each bit can be set separately, and can cause four types of interrupts.

15.2.15 SYSINT2REG (0x0B00 0200)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	DSIU INTR	FIR INTR	TCLK INTR	HSP INTR	LEDINTR	RTCL2 INTR
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:6	RFU	Write 0 to these bits. 0 is returned after a read.
5	DSIUINTR	DSIU interrupt 1: Occurred 0: Normal
4	FIRINTR	FIR interrupt 1: Occurred 0: Normal
3	TCLKINTR	TClock counter interrupt 1: Occurred 0: Normal
2	HSPINTR	HSP interrupt 1: Occurred 0: Normal
1	LEDINTR	LED interrupt 1: Occurred 0: Normal
0	RTCL2INTR	RTCLong2 timer interrupt 1: Occurred 0: Normal

This register indicates when various interrupts occur in the VR4121 system.

15.2.16 GIUINTHREG (0x0B00 0202)

Bit	15	14	13	12	11	10	9	8
Name	INTS31	INTS30	INTS29	INTS28	INTS27	INTS26	INTS25	INTS24
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	INTS23	INTS22	INTS21	INTS20	INTS19	INTS18	INTS17	INTS16
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:0	INTS(31:16)	GPIO(31:16) pin interrupt 1: Occurred 0: Normal

This register indicates when various GIU-related interrupts occur.

15.2.17 FIRINTREG (0x0B00 0204)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	FIRINT	FDPINT4	FDPINT3	FDPINT2	FDPINT1
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:5	RFU	Write 0 to these bits. 0 is returned after a read.
4	FIRINT	Interrupt from FIR unit 1: Occurred 0: Normal
3	FDPINT4	FIR DMA buffer (receive side) 2 page interrupt 1: Occurred 0: Normal
2	FDPINT3	FIR DMA buffer (transmit side) 2 page interrupt 1: Occurred 0: Normal
1	FDPINT2	FIR DMA buffer (receive side) 1 page interrupt 1: Occurred 0: Normal
0	FDPINT1	FIR DMA buffer (transmit side) 1 page interrupt 1: Occurred 0: Normal

This register indicates when various FIR-related interrupts occur.

When FDPINT4 or FDPINT3 is set to 1, the VR4121 stops the DMA requests. When FDPINT2 or FDPINT1 is set to 1 during the FDPCNT bit of the DPCNTR register (0x0C00 0044) is set to 1 (DMA buffer 1 page interrupt is enabled), the VR4121 stops the DMA requests.

15.2.18 MSYSINT2REG (0x0B00 0206)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	DSIU INTR	FIR INTR	TCLK INTR	HSPINTR	LEDINTR	RTCL2 INTR
R/W	R	R	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:6	RFU	Write 0 to these bits. 0 is returned after a read.
5	DSIUINTR	DSIU interrupt enable 1: Enable 0: Prohibit
4	FIRINTR	FIR interrupt enable 1: Enable 0: Prohibit
3	TCLKINTR	TClock counter interrupt enable 1: Enable 0: Prohibit
2	HSPINTR	HSP interrupt enable 1: Enable 0: Prohibit
1	LEDINTR	LED interrupt enable 1: Enable 0: Prohibit
0	RTCL2INTR	RTCLong2 timer interrupt enable 1: Enable 0: Prohibit

This register is used to mask various interrupts in the VR4121 system.

15.2.19 MGIUINTHREG (0x0B00 0208)

Bit	15	14	13	12	11	10	9	8
Name	INTS31	INTS30	INTS29	INTS28	INTS27	INTS26	INTS25	INTS24
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	INTS23	INTS22	INTS21	INTS20	INTS19	INTS18	INTS17	INTS16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:0	INTS(31:16)	Enable GPIO(31:16) pin interrupt 1: Enable 0: Prohibit

This register is used to mask various GIU-related interrupts.

15.2.20 MFIRINTREG (0x0B00 020A)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	FIRINT	FDPINT4	FDPINT3	FDPINT2	FDPINT1
R/W	R	R	R	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:5	RFU	Write 0 to these bits. 0 is returned after a read.
4	FIRINT	FIR unit interrupt enable 1: Enable 0: Prohibit
3	FDPINT4	FIR DMA buffer 2 page interrupt (receive side) enable 1: Enable 0: Prohibit
2	FDPINT3	FIR DMA buffer 2 page interrupt (transmit side) enable 1: Enable 0: Prohibit
1	FDPINT2	FIR DMA buffer 1 page interrupt (receive side) enable 1: Enable 0: Prohibit
0	FDPINT1	FIR DMA buffer 1 page interrupt (transmit side) enable 1: Enable 0: Prohibit

This register is used to mask various FIR-related interrupts.

15.3 Notes for Register Setting

There is no register setting flow in relation to the ICU.

With regard to the interrupt mask registers, the initial setting is "initial = 0 = mask" after setting. Therefore, enough masks must be cleared to provide sufficient interrupts for the CPU's start-up processing. This is always necessary when battint_intr = NMI.

The initial setting for battint_intr is "initial = 0 = NMI". A "1" must be written to the register to switch this setting to Int0.

soft_intr is a software interrupt request that is output to Int0 by setting 1 to the SOFTINTREG register. Writing a "0" clears the interrupt.

[MEMO]

CHAPTER 16 PMU (POWER MANAGEMENT UNIT)

16.1 General

The PMU performs the following power management within the VR4121 and controls the power supply throughout the system which includes the VR4121.

- Reset control
- Shutdown control
- Power-on control
- Low-power consumption mode control

The PMU also performs settings to use the GPIO(12:9) and GPIO(3:0) signals as a startup factor.

16.1.1 Reset control

The operations of the RTC, peripheral units, CPU core, and PMUINTREG bit settings during a reset are listed below.

Table 16-1. Bit Operations During Reset

Reset Type	RTC	Peripheral Units	CPU Core	PMUINTREG
RTC reset	Reset	Reset	Cold reset	RTCRST=1
RSTSW reset	Active	Reset	Cold reset	RSTSW=1

(1) RTC reset

When the RTCRST# signal becomes active, the PMU resets all peripheral units including the RTC unit. It also makes the ccoldresetb and creset signals (internal) active and resets the CPU core.

In addition, the RTCRST bit in PMUINTREG is set to 1. After the CPU is restarted, the RTCRST bit must be checked and cleared to 0 by means of software.

For details of the timing of RTC reset, refer to **8.1.1 RTC reset**.

(2) RSTSW reset

When the RSTSW# signal becomes active, the PMU resets all peripheral units except for RTC and PMU. It also makes the ccoldresetb and creset signals (internal) active and resets the CPU core.

In addition, the RSTSW bit in PMUINTREG is set to 1. After the CPU is restarted, the RSTSW bit must be checked and cleared to 0 by means of software.

For details of the timing of RSTSW reset, refer to **8.1.2 RSTSW**.

16.1.2 Shutdown control

The operations of the RTC, peripheral units, CPU core, and PMUINTREG bit settings during a reset are listed below.

Table 16-2. Bit Operations During Shutdown

Shutdown Type	RTC	Peripheral Units	CPU Core	PMUINTREG
HALTimer shutdown	Active	Reset	Cold reset	HALTIMERRST=1
Deadman's SW shutdown	Active	Reset	Cold reset	TIMOUTRST=1
Software shutdown	Active	Reset	Cold reset	–
BATTINH shutdown	Active	Reset	Cold reset	BATTINH = 1

(1) HALTimer shutdown

After the CPU is activated (following the mode change from Shutdown or Hibernate mode to Fullspeed mode), the software must write "1" to PMUCNTREG’s HALTIMERRST bit within about four seconds to clear the HALTimer.

If the HALTimer is not cleared within about four seconds after the CPU is activated, the PMU asserts the rst_gab signal (internal) and resets all peripheral units except for RTC and PMU. Next, it asserts the ccoldresetb and creset signals (internal) and resets the CPU core.

In addition, the TIMOUTRST bit in PMUINTREG is set to 1. After the CPU is restarted, the TIMOUTRST bit must be checked and cleared to 0 by means of software.

For details of the timing of HALTimer shutdown, refer to **8.1.5 HALTimer shutdown**.

(2) Deadman's SW shutdown

When the Deadman's SW function is enabled, the software must write "1" to DSUCLRREG's DSWCLR bit each specified time, to clear the Deadman's SW counter (for details, see **CHAPTER 18 DSU (Deadman's SW UNIT)**).

If the Deadman's SW counter is not cleared within a specified time, the PMU asserts the rst_gab signal (internal) and resets all peripheral units except for RTC and PMU. Next, it asserts the ccoldresetb and creset signals (internal) and resets the CPU core.

In addition, the DMSRST bit in PMUINTREG is set to 1. After the CPU is restarted, the DMSRST bit must be checked and cleared to 0 by means of software.

For details of the timing of Deadman's SW shutdown, refer to **8.1.3 Deadman's SW**.

(3) Software shutdown

When the HIBERNATE instruction is executed, the PMU checks for currently pending interrupts. If there are no pending interrupts, it asserts the cclockstopen (internal) signal to stop the CPU clock. It then asserts the rst_gab signal (internal) and resets all peripheral units except for RTC and PMU.

The PMU register contents do not change.

For details of the timing of software shutdown, refer to **8.1.4 Software shutdown**.

(4) BATTINH shutdown

If the BATTINH/BATTINT# signal is low when the CPU is going to be low level, PMU stops CPU activation, and resets all peripheral units except for RTC and PMU. Next, it asserts the ccoldresetb and creset signals (internal) and resets the CPU core.

In addition, the BATTINH bit in PMUINTREG is set to 1. After the CPU is restarted, the BATTINH bit must be checked and cleared to 0 by means of software.

For details of the timing of BATTINH shutdown, see **16.1.3 Power-on control** below.

16.1.3 Power-on control

The causes of CPU activation (mode change from shutdown mode or Hibernate mode to Fullspeed mode) are called startup factors. There are twelve startup factors: a power switch interrupt (POWER), eight types of GPIO activation interrupts (GPIO(12:9), (3:0)), a DCD interrupt (DCD#), a touch panel interrupt, and an ElapsedTime timer interrupt.

Battery low detection (BATTINH/BATTINT# pin check) is a factor that prevents CPU activation.

The period (power-on wait time), in which the POWERON pin is active at power-on, can be specified by using PMUWAITREG. After RTCRST, by which the CPU is initialized, the period is 343.75 ms. Power-on wait time can be specified when activation is caused by sources other than RTCRST.

During CPU activation, supplying voltage to the 2.5-V power-supply systems ($V_{DD}2$, $V_{DD}P$, $V_{DD}PD$) can be stopped to reduce the leak current. This means that these power supplies become 0 V while the MPOWER pin is inactive. The following operation will not be affected by supplying voltage of 2.3 V or more to these power supplies within the period from when the MPOWER pin becomes active to when PLL starts oscillation.

Caution When the CPU moves to the Hibernate mode by execution of the HIBERNATE instruction and an activation factor occurs simultaneously, the CPU may be activated without the POWERON signal being active after the MPOWER signal is once inactive. Moreover, if RSTSW, which is not an activation factor from the Hibernate mode, is active at the same time as the activation factor occurs, the CPU may be activated without the POWERON signal being active after the MPOWER signal is inactive once.

(1) Activation via power switch interrupt

When the POWER signal becomes active, the PMU makes the POWERON signal active and provides external notification that the CPU is being activated. After making the POWERON signal active, the PMU checks the BATTINH/BATTINT# signal and then makes the POWERON signal inactive.

If the BATTINH/BATTINT# signal is high, the PMU cancels peripheral unit reset, then starts the Cold Reset sequence to activate the CPU core.

If the BATTINH/BATTINT# signal is low, the PMU sets "1" to PMUINTREG's BATTINH bit and then performs another shutdown. After the CPU is restarted, the BATTINH bit must be checked and cleared by means of software.

Activation via power switch interrupt never sets PMUINTREG's POWERSWINTR bit to 1.

Figure 16-1. Activation via Power Switch Interrupt (BATTINH/BATTINT# = 1)

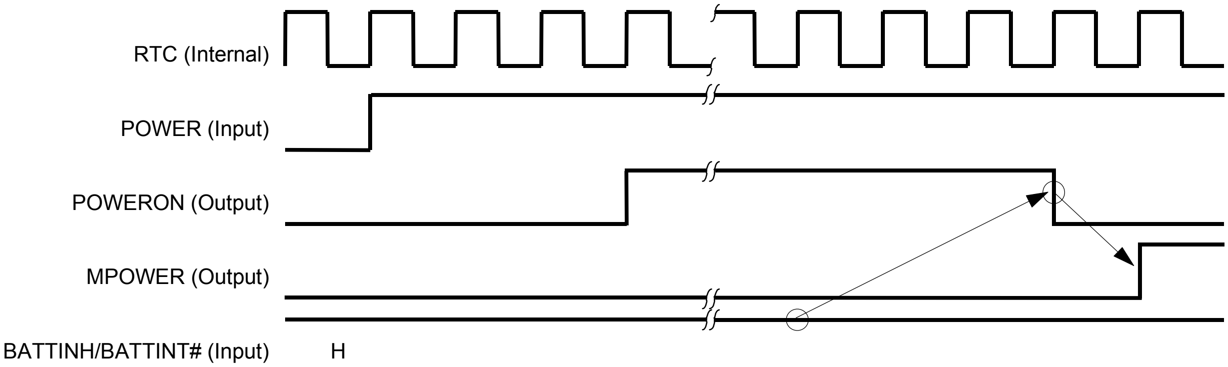
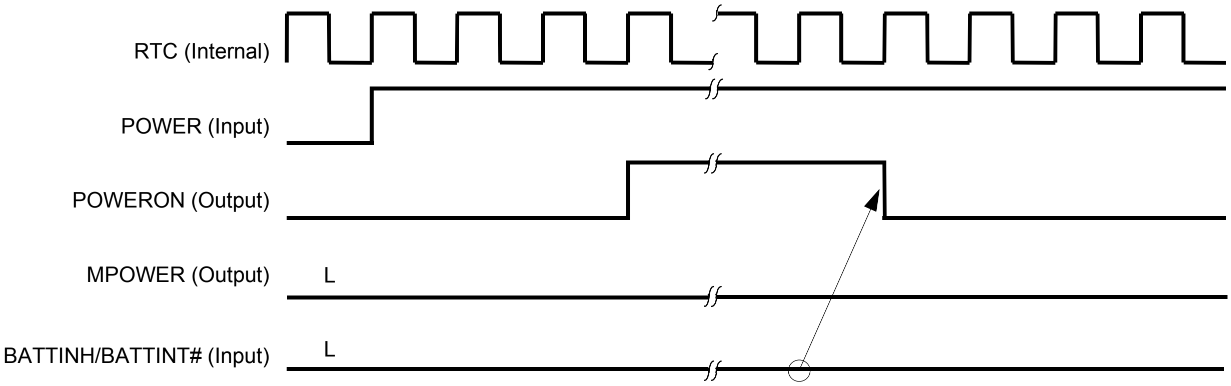


Figure 16-2. Activation via Power Switch Interrupt (BATTINH/BATTINT# = 0)



(2) Activation via GPIO activation interrupt

When the GPIO(12:9), (3:0) signal becomes active, the PMU checks the GPIO(12:9), (3:0)'s activation interrupt enable bit. If GPIO(12:9), (3:0) activation interrupts are enabled, the PMU makes the POWERON signal active and provides external notification that the CPU is being activated (since the GPIO(12:9), (2:0) activation enable interrupt bit is cleared after an RTC is reset, the GPIO(12:9), (2:0) signal cannot be used for activation immediately after an RTC reset. However, activation can occur at the falling edge of the GPIO3 signal immediately after an RTC reset for GPIO3 only). The PMU makes the POWERON signal active, then checks the BATTINH/BATTINT# signal and makes the POWERON signal inactive.

When the BATTINH/BATTINT# signal is high, the PMU cancels peripheral unit reset, then starts the Cold Reset sequence to activate the CPU core.

When the BATTINH/BATTINT# signal is low, the PMU sets "1" to PMUINTREG's BATTINH bit and then performs another shutdown. After the CPU is restarted, the BATTINH bit must be checked and cleared by means of software.

The CPU sets "1" to the corresponding GPIOINTR bit in the PMUINTREG or PMUINT2REG regardless of whether activation succeeds or fails.

Caution The changes in the GPIO signal are ignored while POWER signal is active.

Figure 16-3. Activation via GPIO Activation Interrupt (BATTINH/BATTINT# = 1)

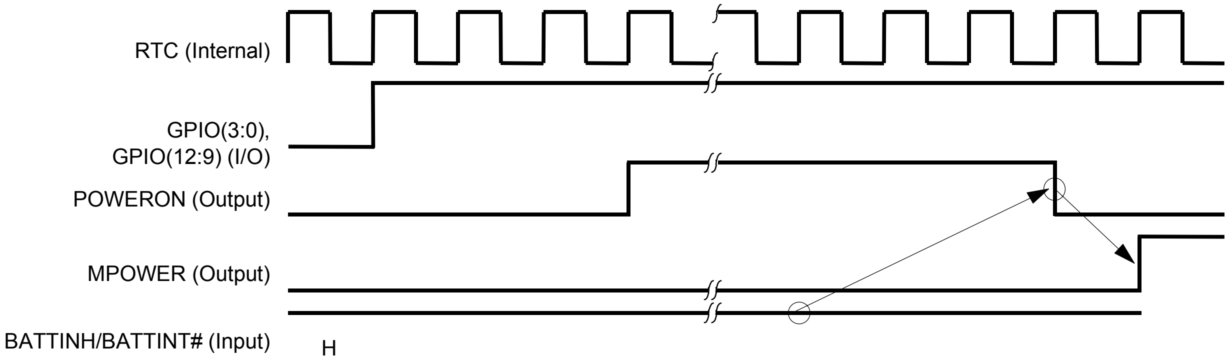
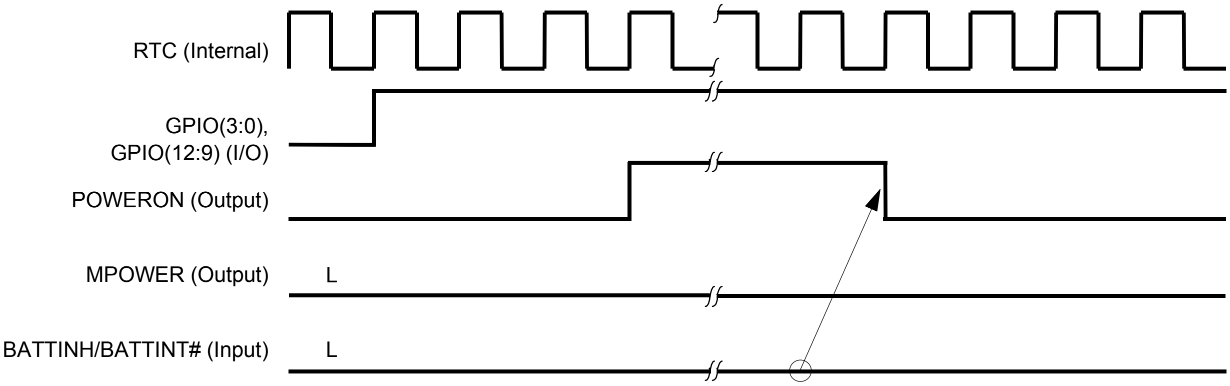


Figure 16-4. Activation via GPIO Activation Interrupt (BATTINH/BATTINT# = 0)



(3) Activation via DCD interrupt

When the DCD# signal becomes active (it means, the falling edge of DCD# signal is detected by PMU), the PMU makes the POWERON signal on and provides external notification that the CPU is being activated. After making the POWERON signal active, the PMU checks the BATTINH/BATTINT# signal and then making the POWERON signal inactive.

If the BATTINH/BATTINT# signal is high, the PMU cancels peripheral unit reset, then starts the Cold Reset sequence to activate the CPU core.

If the BATTINH/BATTINT# signal is low, the PMU sets "1" to PMUINTREG's BATTINH bit and then performs another shutdown. After the CPU is restarted, the BATTINH bit must be checked and cleared by means of software.

The PMUINTREG's DCDST bit in PMU does not indicate whether a DCD interrupt has occurred, but instead reflects the current status of the DCD# pin.

Cautions 1. Once POWERON has been active, the PMU cannot recognize changes in the DCD# signal. If the DCD# state when POWERON is active is different from the DCD# state when POWERON is inactive, the change in the DCD# signal is detected only after POWERON is inactive. However, if the DCD# state when POWERSW is active is the same as the DCD# state when POWERON is inactive, any changes in the DCD# signal that occur while POWERON is active are not detected.

2. The changes in the DCD# signal are ignored while POWER signal is active.

Figure 16-5. Activation via DCD Interrupt (BATTINH/BATTINT# = 1)

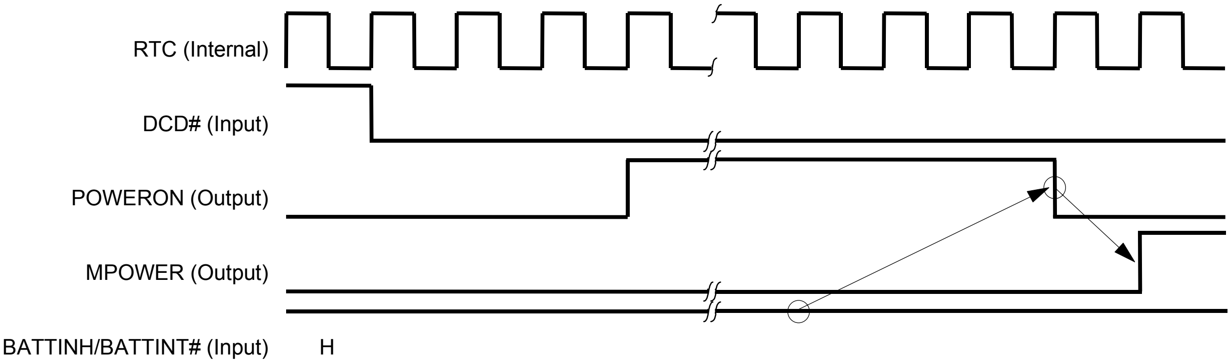
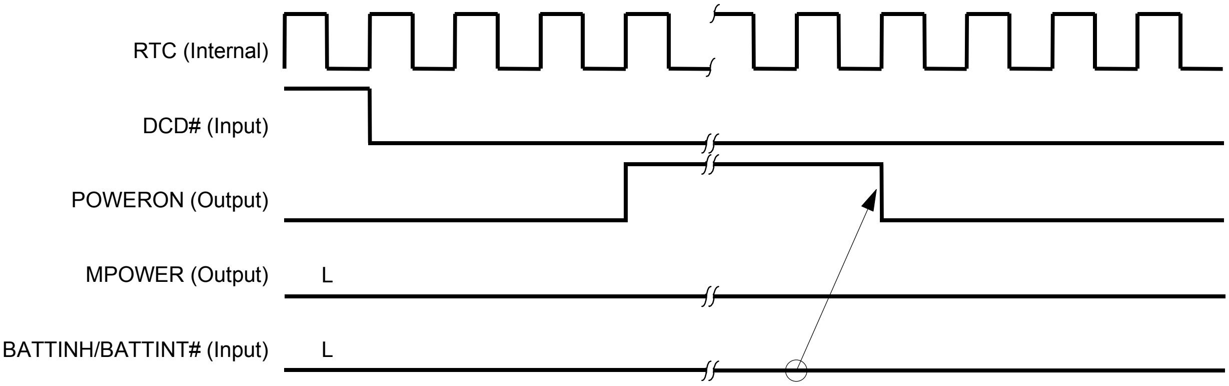


Figure 16-6. Activation via DCD Interrupt (BATTINH/BATTINT# = 0)



(4) Activation via ElapsedTime timer interrupt

When the alarm interrupt (alarm_intr) signal generated from the ElapsedTime timer becomes active, the PMU makes the POWERON signal active and provides external notification that the CPU is being activated. After making the POWERON signal active, the PMU checks the BATTINH/BATTINT# signal and then make the POWERON signal inactive.

If the BATTINH/BATTINT# signal is high, the PMU cancels peripheral unit reset, then starts the Cold Reset sequence to activate the CPU core.

If the BATTINH/BATTINT# signal is low, the PMU sets "1" to PMUINTREG's BATTINH bit and then performs another shutdown. After the CPU is restarted, the BATTINH bit must be checked and cleared by means of software.

Caution ElapsedTime timer interrupt is ignored while the POWER signal is active. After the POWER signal becomes inactive, it is notified to PMU.

Figure 16-7. Activation via Elapsed Timer Interrupt (BATTINH/BATTINT# = 1)

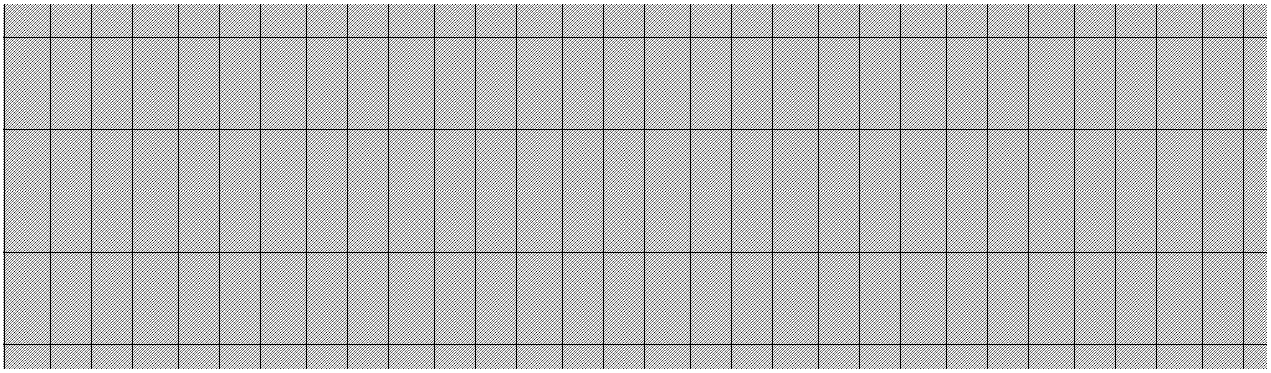
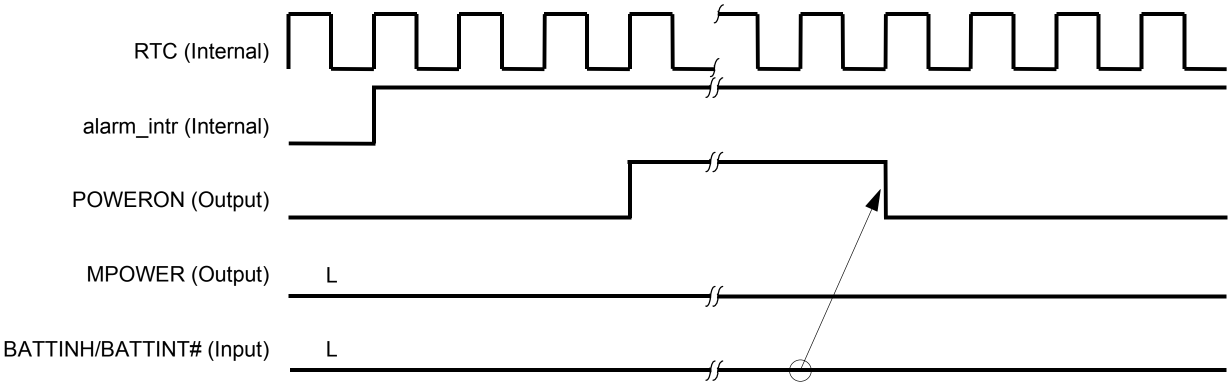


Figure 16-8. Activation via Elapsed Timer Interrupt (BATTINH/BATTINT# = 0)



16.1.4 Power mode

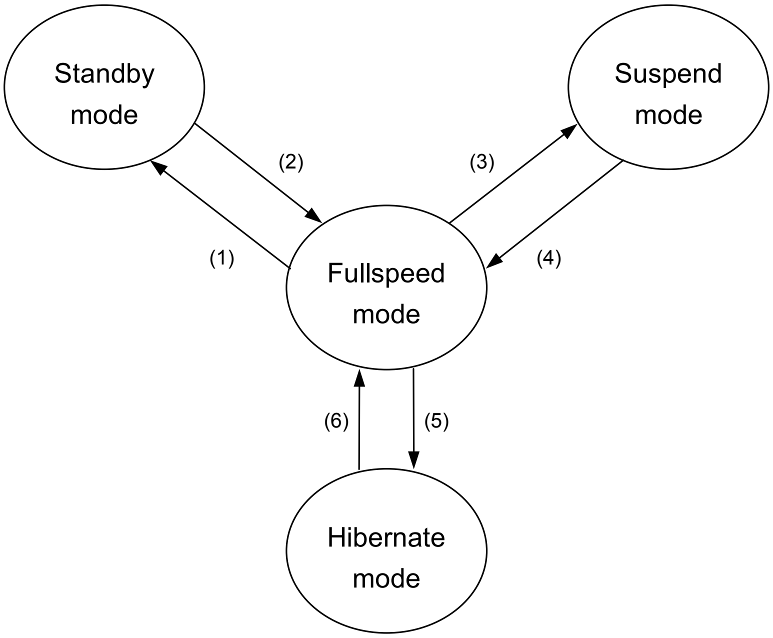
The VR4121 supports the following four power modes.

- Fullspeed mode
- Standby mode
- Suspend mode
- Hibernate mode

To set Standby, Suspend, or Hibernate mode from Fullspeed mode, execute a STANDBY, SUSPEND, or HIBERNATE instruction, respectively. RTCRST is always valid in every mode, and initializes (resets) units in the CPU including RTC. However, the CPU does not restart by RTCRST.

The power mode state transition and its outlines are shown in the next.

Figure 16-9. Power Mode State Transition



(1)	(2)	(3)	(4)	(5)	(6)
STANDBY instruction, pipeline flash, SysAD idle , and PClock high	All interrupts	SUSPEND instruction, pipeline flash, SysAD idle, PClock high, TClock high, and DRAM self refresh	POWER RSTSW Elapsed Time RTCLong1 RTCLong2 KeyTouch PenTouch GPIO(14:9) GPIO(3:0) DCD# (GPIO, SIU) BATTINTR	HIBERNATE instruction, pipeline flash, SysAD idle, PClock high, TClock high, MasterOut high, and DRAM self refresh	POWER ElapsedTime DCD# GPIO(12:9) GPIO(3:0)

Table 16-3. Power Mode

Mode	Internal Peripheral Unit				CPU Core
	RTC	ICU	DCU	Others	
Fullspeed	On	On	On	Selectable[Note]	On
Standby	On	On	On	Selectable[Note]	Off
Suspend	On	On	Off	Off	Off
Hibernate	On	Off	Off	Off	Off
Off	Off	Off	Off	Off	Off

Note See **CHAPTER 14 CMU (CLOCK MASK UNIT)** for details.

(1) Fullspeed mode

In Fullspeed mode, all internal clocks and the bus clock operate. In this mode, all the functions of the VR4121 can be executed.

(2) Standby mode

In Standby mode, all internal clocks, other than those provided to the internal peripheral units and the internal timer/interrupt unit of the CPU core, are fixed to high level.

To switch to Standby mode from Fullspeed mode, first execute the STANDBY instruction. The VR4121 waits until the SysAD bus (internal) enters idle status after the completion of the WB stage of the STANDBY instruction. Then, the internal clock is shut down, and the pipeline stops. PLL, timer/interrupt clock, internal bus clocks (TClock, MasterOut), and RTC continue to operate.

In Standby mode, the processor returns to Fullspeed mode when an interrupt occurs. At this time, the contents of bits indicating the states of pins in the peripheral unit's registers are undefined. The contents of other fields are retained.

(3) Suspend mode

In Suspend mode, all internal clocks (including TClock) other than those supplied to the RTC/ICU/PMU internal peripheral units and the internal timer/interrupt unit of the CPU core are fixed to high level.

To switch to Suspend mode from Fullspeed mode, first execute the SUSPEND instruction. The VR4121 waits until the SysAD bus (internal) enters idle status after the completion of the WB stage of the SUSPEND instruction and DRAM has entered self-refresh mode. Then, the internal clocks (including TClock) are shut down, and the pipeline stops. PLL, timer interrupt clock, MasterOut, and RTC continue to operate.

If the SUSPEND instruction is executed during DMA transfer, the DRAM transfer is suspended, and the operation is undefined.

In Suspend mode, the processor returns to Fullspeed mode when an interrupt request from the peripheral units or any resets occur. At this time, the contents of bits indicating the states of pins in the peripheral unit's registers are undefined. The contents of other fields are retained.

(4) Hibernate mode

In Hibernate mode, all the clocks supplied to internal peripheral units other than RTC/ICU/PMU and to the CPU core are fixed to high level.

To switch to Hibernate mode from Fullspeed mode, first execute the HIBERNATE instruction. The VR4121 waits until the SysAD bus (internal) enters idle status after the completion of the WB stage of the HIBERNATE instruction, DRAM has entered self-refresh mode, and the MPOWER pin has been made inactive. Then, the internal clocks (including TClock and MasterOut) are shut down, and the pipeline stops. PLL also stops, but RTC continue to operate.

In Hibernate mode, the processor returns to Fullspeed mode when it is alarmed from the RTC, the power-on switch is pressed, or DCD# pin is active. At this time, the contents of bits indicating the states of pins in the peripheral unit's registers and caches in the CPU core are undefined. The contents of other fields are retained.

- GPIO pins as an activation factors
  The GPIO(12:9) pin and GPIO(3:0) pin can be used not only as general-purpose ports and interrupt request inputs but also as activation factors from Hibernate mode. In addition, the GPIO3 pin can be used as an activation factor that follows an RTC reset. The selection of these pins as activation factors, as well as the settings for triggering, levels, etc., are set via the PMU rather than via the GIU or ICU (see **16.2 Register Settings**).
  The settings for using GPIO pins for activation after a reset or from another power mode are listed below.

Table 16-4. Activation via GPIO Pins

Pin	RTC Reset		Recovery from Hibernate Mode		Recovery (Interrupt Request) from Suspend or Standby Mode, or Interrupt Request during Fullspeed Mode	
	Synchronization Clock	Trigger	Synchronization Clock	Trigger	Synchronization Clock	Trigger
GPIO(12:9)	–	–	RTC	Rising edge, falling edge	MasterOut	Set via GIU
GPIO3	RTC	Falling edge	RTC	Rising edge, falling edge	RTC	Set via GIU
GPIO(2:0)	–	–	RTC	Rising edge, falling edge	RTC	Set via GIU

16.2 Register Set

The PMU registers are listed below.

Table 16-5. PMU Registers

Address	R/W	Register Symbols	Function
0x0B00 00A0	R/W	PMUINTREG	PMU Interrupt/Status Register
0x0B00 00A2	R/W	PMUCNTREG	PMU Control Register
0x0B00 00A4	R/W	PMUINT2REG	PMU Interrupt/Status 2 Register
0x0B00 00A6	R/W	PMUCNT2REG	PMU Control 2 Resister
0x0B00 00A8	R/W	PMUWAITREG	PMU Wait Counter Register
0x0B00 00AC	R/W	PMUDIVREG	PMU Div Mode Register

Each register is described in detail below.

16.2.1 PMUINTREG (0x0B00 00A0)

(1/2)

Bit	15	14	13	12	11	10	9	8
Name	GPIO3 INTR	GPIO2 INTR	GPIO1 INTR	GPIO0 INTR	RFU	DCDST	RTCINTR	BATTINH
R/W	R/W	R/W	R/W	R/W	R	R	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	memo1	memo0	TIMOUT RST	RTCRST	RSTSW	DMSRST	BATTINTR	POWER SWINTR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	1	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15	GPIO3INTR	GPIO3 activation interrupt detection. Cleared to 0 when 1 is written. 1: Detected 0: Not detected
14	GPIO2INTR	GPIO2 activation interrupt detection. Cleared to 0 when 1 is written. 1: Detected 0: Not detected
13	GPIO1INTR	GPIO1 activation interrupt detection. Cleared to 0 when 1 is written. 1: Detected 0: Not detected
12	GPIO0INTR	GPIO0 activation interrupt detection. Cleared to 0 when 1 is written. 1: Detected 0: Not detected
11	RFU	Write 0 to this bit. 0 is returned after a read.
10	DCDST	DCD# pin state 1: High 0: Low
9	RTCINTR	RTC alarm interrupt detection. Cleared to 0 when 1 is written. 1: Detected 0: Not detected
8	BATTINH	Battery low (BATTINH/BATTINT# signal low level) detection during activation. Cleared to 0 when 1 is written. 1: Detected 0: Not detected

(2/2)

Bit	Name	Function
7:6	memo(1:0)	These bits are readable/writable, and can be used by users freely.
5	TIMOUTRST	HALTimer reset detection. Cleared to 0 when 1 is written. 1: Detected 0: Not detected
4	RTCRST	RTC reset detection. Cleared to 0 when 1 is written. 1: Detected 0: Not detected
3	RSTSW	RESET switch interrupt detection or Deadman's SW interrupt request detection. Cleared to 0 when 1 is written. 1: Detected 0: Not detected
2	DMSRST	Deadman's switch interrupt detection. Cleared to 0 when 1 is written. 1: Detected 0: Not detected
1	BATTINTR	Battery low detection during normal operation. Cleared to 0 when 1 is written. 1: Detected 0: Not detected
0	POWERSWINTR	POWER switch interrupt detection. Cleared to 0 when 1 is written. 1: Detected 0: Not detected

This register is used to set whether the CPU detects a power-on factor and reset.

It also indicates the status of the DCD# pin.

When a deadman's switch interrupt request has occurred, the DMSRST bit and the RSTSW bit are both set.

The BATTINTR bit is set to 1 when the BATTINH/BATTINT# signal becomes low and a battery-low interrupt request occurs in modes other than the Hibernate mode (MPOWER = 1).

The POWERSWINTR bit is set to 1 when the POWER signal becomes high and a power switch request occurs in modes other than the Hibernate mode. However, this bit is not set to 1 when the POWER signal becomes high in the Hibernate mode (MPOWER = 0).

16.2.2 PMUCNTREG (0x0B00 00A2)

(1/2)

Bit	15	14	13	12	11	10	9	8
Name	GPIO3MSK	GPIO2MSK	GPIO1MSK	GPIO0MSK	GPIO3TRG	GPIO2TRG	GPIO1TRG	GPIO0TRG
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	1	0	0	0	1	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	7	6	5	4	3	2	1	0
Name	STANDBY	RFU	RFU	RFU	RFU	HALTIMER RST	RFU	RFU
R/W	R/W	R	R	R	R	R/W	R	R
RTCRST	0	0	0	0	0	0	1	0
After reset	0	0	0	0	0	0	1	0

Bit	Name	Function
15	GPIO3MSK	GPIO3 activation enable 1: Enable 0: Prohibit
14	GPIO2MSK	GPIO2 activation enable 1: Enable 0: Prohibit
13	GPIO1MSK	GPIO1 activation enable 1: Enable 0: Prohibit
12	GPIO0MSK	GPIO0 activation enable 1: Enable 0: Prohibit
11	GPIO3TRG	GPIO3 activation interrupt type 1: Falling edge detection 0: Rising edge detection
10	GPIO2TRG	GPIO2 activation interrupt type 1: Falling edge detection 0: Rising edge detection
9	GPIO1TRG	GPIO1 activation interrupt type 1: Falling edge detection 0: Rising edge detection
8	GPIO0TRG	GPIO0 activation interrupt type 1: Falling edge detection 0: Rising edge detection
7	STANDBY	Standby mode setting. This setting is performed only for software, and does not affect hardware in any way. 1: Standby mode 0: Normal mode

Note Holds the value before reset.

(2/2)

Bit	Name	Function
6:3	RFU	Write 0 to these bits. 0 is returned after a read.
2	HALTIMERRST	HALTimer reset 1: Reset 0: Set
1	RFU	Write 1 to this bit. 1 is returned after a read.
0	RFU	Write 0 to this bit. 0 is returned after a read.

This register is used to set CPU shutdown and overall system management operations.

The HALTIMERRST bit must be reset within about four seconds after activation. Resetting of the HALTIMERRST bit indicates that the VR4121 itself has been activated normally. If the HALTIMERRST bit is not reset within about four seconds after activation, program execution is regarded as abnormal (possibly due to a runaway) and an automatic shutdown is performed.

The GPIO(3:0)MSK bits are used to set enable/prohibit for activation from Hibernate mode when the corresponding interrupt (GPIO(3:0)) occurs. The GPIO3MSK bit is set to 1 by RTCRST, and the other bits are cleared to "0" (prohibit). Accordingly, the GPIO(2:0) cannot be used for activation immediately after an RTCRST reset. The GPIO activation interrupt is valid only when the CPU is in the Hibernate mode.

16.2.3 PMUINT2REG (0x0B00 00A4)

Bit	15	14	13	12	11	10	9	8
Name	GPIO12 INTR	GPIO11 INTR	GPIO10 INTR	GPIO9 INTR	RFU	RFU	RFU	RFU
R/W	R/W	R/W	R/W	R/W	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15	GPIO12INTR	GPIO12 activation interrupt detection. Cleared to 0 when 1 is written. 1: Detected 0: Not detected
14	GPIO11INTR	GPIO11 activation interrupt detection. Cleared to 0 when 1 is written. 1: Detected 0: Not detected
13	GPIO10INTR	GPIO10 activation interrupt detection. Cleared to 0 when 1 is written. 1: Detected 0: Not detected
12	GPIO9INTR	GPIO9 activation interrupt detection. Cleared to 0 when 1 is written. 1: Detected 0: Not detected
11:0	RFU	Write 0 to these bits. 0 is returned after a read.

This register is used to specify whether the GPIO(12:9) interrupts are detected as power-on factors.

16.2.4 PMUCNT2REG (0x0B00 00A6)

Bit	15	14	13	12	11	10	9	8
Name	GPIO12 MSK	GPIO11 MSK	GPIO10 MSK	GPIO9 MSK	GPIO12 TRG	GPIO11 TRG	GPIO10 TRG	GPIO9 TRG
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15	GPIO12MSK	GPIO12 activation enable 1: Enable 0: Prohibit
14	GPIO11MSK	GPIO11 activation enable 1: Enable 0: Prohibit
13	GPIO10MSK	GPIO10 activation enable 1: Enable 0: Prohibit
12	GPIO9MSK	GPIO9 activation enable 1: Enable 0: Prohibit
11	GPIO12TRG	GPIO12 activation interrupt type 1: Falling edge detection 0: Rising edge detection
10	GPIO11TRG	GPIO11 activation interrupt type 1: Falling edge detection 0: Rising edge detection
9	GPIO10TRG	GPIO10 activation interrupt type 1: Falling edge detection 0: Rising edge detection
8	GPIO9TRG	GPIO9 activation interrupt type 1: Falling edge detection 0: Rising edge detection
7:0	RFU	Write 0 to these bits. 0 is returned after a read.

Note Hold the value before reset.

This register is used to set the CPU activation by means of the GPIO(12:9) interrupt.

The GPIO(12:9)MSK bits are used to set enable/prohibit for activation from Hibernate mode when the corresponding interrupt (GPIO(12:9)) occurs. All mask bits are cleared to 0 (prohibit) after RTCRST reset. Therefore, GPIO(12:9) interrupts cannot be used for activation immediately after RTCRST reset. Additionally, the GPIO activation interrupt is valid only when the CPU is in the Hibernate mode.

16.2.5 PMUWAITREG (0x0B00 00A8)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	WCOUNT 13	WCOUNT 12	WCOUNT 11	WCOUNT 10	WCOUNT 9	WCOUNT 8
R/W	R	R	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	1	0	1	1	0	0
After reset	0	0	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	7	6	5	4	3	2	1	0
Name	WCOUNT 7	WCOUNT 6	WCOUNT 5	WCOUNT 3	WCOUNT 3	WCOUNT 2	WCOUNT 1	WCOUNT 0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15:14	RFU	Write 0 to these bits. 0 is returned after a read.
13:0	WCOUNT (13:0)	Activation wait time timer count value Activation wait time = WCOUNT(13:0) × (1/32.768) ms

Note Hold the value before reset.

This register is used to set the activation wait time when the CPU is activated.

This register is set to 0x2C00 (it sets 343.75-ms activation wait time) after RTC reset. Therefore, the 343.75-ms wait time is always inserted as an activation wait time, when the CPU is activated immediately after RTC reset. The activation wait time can be changed by setting this register for the CPU activation from the Hibernate mode.

When this register is set to 0x0, 0x1, 0x2, 0x3, or 0x4, the operation is not guaranteed. Be sure to set a value greater than or 0x5 to this register.

16.2.6 PMUDIVREG (0x0B00 00AC)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	DIV3	DIV2	DIV1	DIV0
R/W	R	R	R	R	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15:13	RFU	Write a "0" here. When this bit is read, "0" will be returned.
3:0	DIV(3:0)	Div mode setting 1111: RFU : 1011: RFU 1010: Div2.5 mode 1001: Div1.5 mode 1000: RFU 0111: RFU 0110: Div6 mode 0101: Div5 mode 0100: Div4 mode 0011: Div3 mode 0010: Div2 mode 0001: Div1 mode 0000: Same mode as set via CLKSEL(2:0) during RTC reset

Note Hold the value before reset.

This register is used to set the CPU core's Div mode. The Div mode setting determines the division rate of the TClock and VTClock in relation to the pipeline clock (PClock) frequency.

Since the contents of this register are cleared to "0" during an RTC reset, the Div mode setting immediately after an RTC reset is set as shown in Table 16-6, based on the CLKSEL(2:0) status.

Once the Div mode has been set via this register, the setting does not become effective immediately in the processor's operations. It becomes valid after a reset other than an RTC reset occurs.

The relation between CLKSEL settings and various clock frequency values is shown in Table 16-6 below.

Table 16-6. CLKSEL and Frequency of PClock, TClock, and VTClock

CLKSEL(2:0)	Div Mode	PClock Frequency	TClock, VTClock Frequency
111	RFU	RFU	RFU
110	Div6	168.5 MHz	28.1 MHz
101	Div5	147.5 MHz	29.5 MHz
100	Div4	131.1 MHz	32.8 MHz
011	Div4	118.0 MHz	29.5 MHz
010	Div3	98.3 MHz	32.8 MHz
001	Div3	90.7 MHz	30.2 MHz
000	Div3	78.6 MHz	26.2 MHz

[MEMO]

CHAPTER 17 RTC (REALTIME CLOCK UNIT)

17.1 General

The RTC unit has a total of four timers, including the following three types.

- RTCLong .... This is a 24-bit programmable counter that counts down using 32.768-kHz frequency. Cycle interrupts occur for up to 512 seconds.

- TClockCount ..... This is a 25-bit programmable counter that counts down using TClock cycles. Cycle interrupts occur for up to 1 to 2 seconds. This counter is used for performance evaluation.

- ElapsedTime ..... This is a 48-bit up counter that counts up using 32.768-kHz frequency. It counts up to 272 years before returning to zero. It includes 48-bit comparators (ECMPHREG, ECMPLREG, and ECMPMREG) and 48-bit alarm time registers (ETIMELREG, ETIMEMREG, and ETIMEHREG) to enable interrupts to occur at specified times.

17.2 Register Set

The RTC registers are listed below.

Table 17-1. RTC Registers

Address	R/W	Register Symbols	Function
0x0B00 00C0	R/W	ETIMELREG	Elapsed Time L Register
0x0B00 00C2	R/W	ETIMEMREG	Elapsed Time M Register
0x0B00 00C4	R/W	ETIMEHREG	Elapsed Time H Register
0x0B00 00C8	R/W	ECMPLREG	Elapsed Compare L Register
0x0B00 00CA	R/W	ECMPMREG	Elapsed Compare M Register
0x0B00 00CC	R/W	ECMPHREG	Elapsed Compare H Register
0x0B00 00D0	R/W	RTCL1LREG	RTC Long 1 L Register
0x0B00 00D2	R/W	RTCL1HREG	RTC Long 1 H Register
0x0B00 00D4	R	RTCL1CNTLREG	RTC Long 1 Count L Register
0x0B00 00D6	R	RTCL1CNTHREG	RTC Long 1 Count H Register
0x0B00 00D8	R/W	RTCL2LREG	RTC Long 2 L Register
0x0B00 00DA	R/W	RTCL2HREG	RTC Long 2 H Register
0x0B00 00DC	R	RTCL2CNTLREG	RTC Long 2 Count L Register
0x0B00 00DE	R	RTCL2CNTHREG	RTC Long 2 Count H Register
0x0B00 01C0	R/W	TCLKLREG	TCLK L Register
0x0B00 01C2	R/W	TCLKHREG	TCLK H Register
0x0B00 01C4	R	TCLKCNTLREG	TCLK Count L Register
0x0B00 01C6	R	TCLKCNTHREG	TCLK Count H Register
0x0B00 01DE	R/W	RTCINTREG	RTC Interrupt Register

Each register is described in detail below.

17.2.1 Elapsed time registers

(1) ETIMELREG (0x0B00 00C0)

Bit	15	14	13	12	11	10	9	8
Name	ETIME15	ETIME14	ETIME13	ETIME12	ETIME11	ETIME10	ETIME9	ETIME8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	7	6	5	4	3	2	1	0
Name	ETIME7	ETIME6	ETIME5	ETIME4	ETIME3	ETIME2	ETIME1	ETIME0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15:0	ETIME(15:0)	ElapsedTime bit 15:0

Note Continues counting.

(2) ETIMEMREG (0x0B00 00C2)

Bit	15	14	13	12	11	10	9	8
Name	ETIME31	ETIME30	ETIME29	ETIME28	ETIME27	ETIME26	ETIME25	ETIME24
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	7	6	5	4	3	2	1	0
Name	ETIME23	ETIME22	ETIME21	ETIME20	ETIME19	ETIME18	ETIME17	ETIME16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15:0	ETIME(31:16)	ElapsedTime bit 31:16

Note Continues counting.

(3) ETIMEHREG (0x0B00 00C4)

Bit	15	14	13	12	11	10	9	8
Name	ETIME47	ETIME46	ETIME45	ETIME44	ETIME43	ETIME42	ETIME41	ETIME40
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	7	6	5	4	3	2	1	0
Name	ETIME39	ETIME38	ETIME37	ETIME36	ETIME35	ETIME34	ETIME33	ETIME32
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15:0	ETIME(47:32)	ElapsedTime bit 47:32

Note Continues counting

These registers indicate the elapsed timer's value. They count up using a 32.768-kHz frequency and when a match occurs with the elapsed compare registers, an alarm (elapsed time interrupt) occurs (and the count-up continues). The setting is valid once values have been written to all registers (ETIMELREG, ETIMEMREG, and ETIMEHREG).

These registers have no buffers for read. Therefore, an illegal data may be read if the counter value changes during a read operation. When using a read data, be sure to read a value twice and check that two read vales are the same.

When setting these registers again, wait until at least 100 μs (32.768-kHz clock × 3) have elapsed before doing so.

17.2.2 Elapsed time compare registers

(1) ECMPLREG (0x0B00 00C8)

Bit	15	14	13	12	11	10	9	8
Name	ECMP15	ECMP14	ECMP13	ECMP12	ECMP11	ECMP10	ECMP9	ECMP8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	7	6	5	4	3	2	1	0
Name	ECMP7	ECMP6	ECMP5	ECMP4	ECMP3	ECMP2	ECMP1	ECMP0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15:0	ECMP(15:0)	Value to be compared with ElapsedTime bit 15:0

Note Previous value is retained.

(2) ECMPMREG (0x0B00 00CA)

Bit	15	14	13	12	11	10	9	8
Name	ECMP31	ECMP30	ECMP29	ECMP28	ECMP27	ECMP26	ECMP25	ECMP24
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	7	6	5	4	3	2	1	0
Name	ECMP23	ECMP22	ECMP21	ECMP20	ECMP19	ECMP18	ECMP17	ECMP16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15:0	ECMP(31:16)	Value to be compared with ElapsedTime bit 31:16

Note Previous value is retained.

(3) ECMPHREG (0x0B00 00CC)

Bit	15	14	13	12	11	10	9	8
Name	ECMP47	ECMP46	ECMP45	ECMP44	ECMP43	ECMP42	ECMP41	ECMP40
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	7	6	5	4	3	2	1	0
Name	ECMP39	ECMP38	ECMP37	ECMP36	ECMP35	ECMP34	ECMP33	ECMP32
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15:0	ECMP(47:32)	Value to be compared with ElapsedTime bit 47:32

Note Previous value is retained.

Use these registers to set the values to be compared with values in the elapsed time registers.

The setting is valid once values have been written to all registers (ECMPLREG, ECMPMREG, and ECMPHREG).

When setting these registers again, wait until at least 100 μs (32.768-kHz clock × 3) have elapsed before doing so.

17.2.3 RTC long 1 registers

(1) RTCL1LREG (0x0B00 00D0)

Bit	15	14	13	12	11	10	9	8
Name	RTCL1P15	RTCL1P14	RTCL1P13	RTCL1P12	RTCL1P11	RTCL1P10	RTCL1P9	RTCL1P8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	7	6	5	4	3	2	1	0
Name	RTCL1P7	RTCL1P6	RTCL1P5	RTCL1P4	RTCL1P3	RTCL1P2	RTCL1P1	RTCL1P0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15:0	RTCL1P(15:0)	15:0 for RTCLong1 counter cycle

Note Previous value is retained.

(2) RTCL1HREG (0x0B00 00D2)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	7	6	5	4	3	2	1	0
Name	RTCL1P23	RTCL1P22	RTCL1P21	RTCL1P20	RTCL1P19	RTCL1P18	RTCL1P17	RTCL1P16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15:8	RFU	Write 0 when writing. 0 is returned after a read.
7:0	RTCL1P(23:16)	23:16 for RTCLong1 counter cycle

Note Previous value is retained.

Use these registers to set the RTCLong1 counter cycle. The RTCLong1 counter begins its countdown at the value written to these registers.

The setting is valid once values have been written to both registers (RTCL1LREG and RTCL1HREG).

When setting these registers again, wait until at least 100 μs (32.768-kHz clock × 3) have elapsed before doing so.

Cautions 1. The RTC unit is stopped when all zeros are written.

2. Any combined setting of "RTCL1HREG = 0x0000" and "RTCL1LREG = 0x0001, 0x0002, 0x0003, 0x0004" is prohibited.

17.2.4 RTC long 1 count registers

(1) RTCL1CNTLREG (0x0B00 00D4)

Bit	15	14	13	12	11	10	9	8
Name	RTCL1C15	RTCL1C14	RTCL1C13	RTCL1C12	RTCL1C11	RTCL1C10	RTCL1C9	RTCL1C8
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	7	6	5	4	3	2	1	0
Name	RTCL1C7	RTCL1C6	RTCL1C5	RTCL1C4	RTCL1C3	RTCL1C2	RTCL1C1	RTCL1C0
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15:0	RTCL1C(15:0)	RTCLong1 counter bit 15:0

Note Continues counting.

(2) RTCL1CNTHREG (0x0B00 00D6)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	7	6	5	4	3	2	1	0
Name	RTCL1C23	RTCL1C22	RTCL1C21	RTCL1C20	RTCL1C19	RTCL1C18	RTCL1C17	RTCL1C16
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15:8	RFU	Write 0 to these bits. 0 is returned after a read.
7:0	RTCL1C(23:16)	RTCLong1 counter bit 23:16

Note Continues counting.

These registers indicate the RTCLong1 counter's values. The countdown uses a 32.768-kHz frequency and begins at the value set to the RTCLong1 registers. An RTCLong1 interrupt occurs when the counter reaches 0x00 0001 (at which point the counter returns to the start value and continues counting).

These registers have no buffers for read. Therefore, an illegal data may be read if the counter value changes during a read operation. When using a read data, be sure to read a value twice and check that two read vales are the same.

When setting these registers again, wait until at least 100 μs (32.768-kHz clock × 3) have elapsed before doing so.

17.2.5 RTC long 2 registers

(1) RTCL2LREG (0x0B00 00D8)

Bit	15	14	13	12	11	10	9	8
Name	RTCL2P15	RTCL2P14	RTCL2P13	RTCL2P12	RTCL2P11	RTCL2P10	RTCL2P9	RTCL2P8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	7	6	5	4	3	2	1	0
Name	RTCL2P7	RTCL2P6	RTCL2P5	RTCL2P4	RTCL2P3	RTCL2P2	RTCL2P1	RTCL2P0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15:0	RTCL2P(15:0)	15:0 for RTCLong2 counter cycle

Note Previous value is retained.

(2) RTCL2HREG (0x0B00 00DA)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	7	6	5	4	3	2	1	0
Name	RTCL2P23	RTCL2P22	RTCL2P21	RTCL2P20	RTCL2P19	RTCL2P18	RTCL2P17	RTCL2P16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15:8	RFU	Write 0 to these bits. 0 is returned after a read.
7:0	RTCL2P(23:16)	23:16 for RTCLong2 counter cycle

Note Previous value is retained.

Use these registers to set the RTCLong2 counter cycle. The RTCLong2 counter begins its countdown at the value written to these registers.

The setting is valid once values have been written to both registers (RTCL2LREG and RTCL2HREG).

When setting these registers again, wait until at least 100 μs (32.768-kHz clock × 3) have elapsed before doing so.

Cautions 1. The RTC unit is stopped when all zeros are written.

2. Any combined setting of "RTCL2HREG = 0x0000" and "RTCL2LREG = 0x0001, 0x0002, 0x0003, 0x0004" is prohibited.

17.2.6 RTC long 2 count registers

(1) RTCL2CNTLREG (0x0B00 00DC)

Bit	15	14	13	12	11	10	9	8
Name	RTCL2C15	RTCL2C14	RTCL2C13	RTCL2C12	RTCL2C11	RTCL2C10	RTCL2C9	RTCL2C8
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	7	6	5	4	3	2	1	0
Name	RTCL2C7	RTCL2C6	RTCL2C5	RTCL2C4	RTCL2C3	RTCL2C2	RTCL2C1	RTCL2C0
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15:0	RTCL2C(15:0)	RTCLong2 counter bit 15:0

Note Continues counting.

(2) RTCL2CNTHREG (0x0B00 00DE)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	7	6	5	4	3	2	1	0
Name	RTCL2C23	RTCL2C22	RTCL2C21	RTCL2C20	RTCL2C19	RTCL2C18	RTCL2C17	RTCL2C16
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15:8	RFU	Write 0 to these bits. 0 is returned after a read.
7:0	RTCL2C(23:16)	RTCLong2 counter bit 23:16

Note Continues counting.

These registers indicate the RTCLong2 counter's values. The countdown uses a 32.768-kHz frequency and begins at the value set to the RTCLong2 registers. An RTCLong2 interrupt occurs when the counter reaches 0x00 0001 (at which point the counter returns to the start value and continues counting).

These registers have no buffers for read. Therefore, an illegal data may be read if the counter value changes during a read operation. When using a read data, be sure to read a value twice and check that two read vales are the same.

17.2.7 TClock counter registers

(1) TCLKLREG (0x0B00 01C0)

Bit	15	14	13	12	11	10	9	8
Name	TCLKP15	TCLKP14	TCLKP13	TCLKP12	TCLKP11	TCLKP10	TCLKP9	TCLKP8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	TCLKP7	TCLKP6	TCLKP5	TCLKP4	TCLKP3	TCLKP2	TCLKP1	TCLKP0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:0	TCLKP(15:0)	15:0 for TClock counter cycle

(2) TCLKHREG (0x0B00 01C2)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	TCLKP24
R/W	R	R	R	R	R	R	R	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	TCLKP23	TCLKP22	TCLKP21	TCLKP20	TCLKP19	TCLKP18	TCLKP17	TCLKP16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:9	RFU	Write 0 to these bits. 0 is returned after a read.
8:0	TCLKP(24:16)	24:16 for TClock counter cycle

Use these registers to set the TCLK counter cycle. The TCLK counter begins its countdown at the value written to these registers.

The setting is valid once values have been written to both registers (TCLKLREG and TCLKHREG).

Caution The TCLK unit is stopped when all zeros are written.

17.2.8 TClock counter count registers

(1) TCLKCNTLREG (0x0B00 01C4)

Bit	15	14	13	12	11	10	9	8
Name	TCLKC15	TCLKC14	TCLKC13	TCLKC12	TCLKC11	TCLKC10	TCLKC9	TCLKC8
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	TCLKC7	TCLKC6	TCLKC5	TCLKC4	TCLKC3	TCLKC2	TCLKC1	TCLKC0
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:0	TCLKC(15:0)	TClock counter 15:0

(2) TCLKCNTHREG (0x0B00 01C6)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	TCLKC24
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	TCLKC23	TCLKC22	TCLKC21	TCLKC20	TCLKC19	TCLKC18	TCLKC17	TCLKC16
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:9	RFU	Write 0 to these bits. 0 is returned after a read.
8:0	TCLKC(24:16)	TClock counter 24:16

Use these registers to set the TCLK counter value. The TCLKCNT counter begins its countdown at the value written to the TCLK counter registers. An TCLK counter interrupt occurs when the counter reaches 0x000 0001 (at which point the counter returns to the start value and continues counting).

These registers have no buffers for read. Therefore, an illegal data may be read if the counter value changes during a read operation. When using a read data, be sure to read a value twice and check that two read vales of the higher 9 bits are the same.

17.2.9 RTC interrupt register

(1) RTCINTREG (0x0B00 01DE)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RTCINTR3	RTCINTR2	RTCINTR1	RTCINTR0
R/W	R	R	R	R	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	**Note**	**Note**	**Note**

Bit	Name	Function
15:4	RFU	Write 0 to these bits. 0 is returned after a read.
3	RTCINTR3	TClock counter interrupt. Cleared to 0 when 1 is written. 1: Occurred 0: Normal
2	RTCINTR2	RTCLong2 interrupt. Cleared to 0 when 1 is written. 1: Occurred 0: Normal
1	RTCINTR1	RTCLong1 interrupt. Cleared to 0 when 1 is written. 1: Occurred 0: Normal
0	RTCINTR0	Status bit for elapsed time interrupt. Cleared to 0 when 1 is written. 1: Occurred 0: Normal

Note Previous value is retained.

This register is used to set/indicate the occurrences of interrupt requests of RTC.

[MEMO]

CHAPTER 18 DSU (DEADMAN'S SWITCH UNIT)

18.1 General

The DSU detects when the VR4121 is in runaway (endless loop) state and resets the VR4121 to minimize runaway time. The use of the DSU to minimize runaway time effectively minimizes data loss that can occur due to software-related runaway states.

18.2 Register Set

The DSU registers are listed below.

Table 18-1. DSU Registers

Address	R/W	Symbol	Function
0x0B00 00E0	R/W	DSUCNTREG	DSU Control Register
0x0B00 00E2	R/W	DSUSETREG	DSU Dead Time Set Register
0x0B00 00E4	W	DSUCLRREG	DSU Clear Register
0x0B00 00E6	R/W	DSUTIMREG	DSU Elapsed Time Register

Each register is described in detail below.

18.2.1 DSUCNTREG (0x0B00 00E0)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	DSWEN
R/W	R	R	R	R	R	R	R	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:1	RFU	Write 0 to these bits. 0 is returned after a read.
0	DSWEN	Deadman's Switch function enable 1: Enable 0: Prohibit

This register is used to enable use of the Deadman's Switch functions.

18.2.2 DSUSETREG (0x0B00 00E2)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	DEDTIME3	DEDTIME2	DEDTIME1	DEDTIME0
R/W	R	R	R	R	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:4	RFU	Write 0 to these bits. 0 is returned after a read.
3:0	DEDTIME(3:0)	Deadman's Switch cycle setting 1111: 15 sec 1110: 14 sec : 0010: 2 sec 0001: 1 sec 0000: RFU

This register sets the cycle for Deadman's Switch functions.

The Deadman's Switch cycle can be set in 1-second increments in a range from 1 to 15 seconds. However, the VR4121's operation is undefined when 0x0 has been set to DEDTIME(3:0). The DSUCLRREG's DSWCLR bit must be set by means of software within the specified cycle time.

18.2.3 DSUCLRREG (0x0B00 00E4)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	DSWCLR
R/W	R	R	R	R	R	R	R	W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:1	RFU	Write 0 to these bits. 0 is returned after a read.
0	DSWCLR	Deadman's Switch counter clear. Cleared to 0 when 1 is written. 1: Clear 0: Don't clear

This register clears the Deadman's Switch counter by setting the DSWCLR bit in this register to 1.

The VR4121 automatically shuts down if 1 is not written to this register within the period specified in DSUSETREG.

18.2.4 DSUTIMREG (0x0B00 00E6)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	CRTTIME3	CRTTIME2	CRTTIME1	CRTTIME0
R/W	R	R	R	R	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:4	RFU	Write 0 to these bits. 0 is returned after a read.
3:0	CRTTIME(3:0)	Current Deadman's Switch timer value (elapsed time) 1111: 15 sec 1110: 14 sec : 0010: 2 sec 0001: 1 sec 0000: RFU

This register indicates the elapsed time for the current Deadman's Switch timer.

18.3 Register Setting Flow

The DSU register setting flow is described below.

1. Set the DSU's count-up value (From 1 to 15 seconds).
   The CPU will be reset if it does not clear (1 is not written to DSUCLRREG) the timer within this time period.
   DSUDTMREG address: 0x0B00 00E2 data: 0x000x

2. Enable the DSU
   DSUCNTREG address: 0x0B00 00E0 data: 0x0001

3. Clear the timer within the time period specified in step 1 above.
   DSUCLRREG address: 0x0B00 00E4 data: 0x0001

   For normal use, repeat step 3. To obtain the current elapsed time:
   DSUTIMREG address: 0x0B00 00E6 read (4 bits)

4. Disable the DSU for DOZE mode or a shutdown.
   DSUCNTREG address: 0x0B00 00E0 data: 0x0000

CHAPTER 19 GIU (GENERAL-PURPOSE I/O UNIT)

19.1 General

The GIU controls the GPIO and DCD# pins. GPIO is a general-purpose port for which input and output are available. An interrupt request signal input function can be assigned to GPIO with input signal change (rising edge or falling edge of signal), low level, or high level used as the trigger.

When not used for an interrupt, the registers corresponding to these pins can be written to output a low-level or high-level signal. Each register can be read to check the state of the signal currently being input to the corresponding pin.

The GPIO pins can be used as transition factors from the Standby, Suspend, or Hibernate mode to the Fullspeed mode. When the same setting as for generating an interrupt request is made the operation returns from the standby mode to the Fullspeed mode when the GPIO(31:0) or GPIO(14:9) pins becomes active, and from the suspend mode to the Fullspeed mode when the GPIO(3:0) pin becomes active. When these pins are specified as activation factors by the PMU registers and the GPIO(12:9) and GPIO(3:0) pins become active, the operation shifts from Hibernate mode to Fullspeed mode. Moreover, GPIO3 pins can be used as an activation factor from the RTC reset by setting each register of PMU.

An interrupt request is acknowledged from SIU as well as GIU to GPIO15 (DCD#).

Mask requests as necessary.

Table 19-1 shows settings when GPIO pins are used for interrupt requests or activation.

Table 19-1. Activation and Interrupt Request by GPIO Pins

Pin	RTC Reset Activation		Hibernate Mode Return		Suspend Mode Return (Interrupt Request)		Standby Mode Return (Interrupt Request) Interrupt Request in Fullspeed Mode	
	Synchronous Clock	Trigger	Synchronous Clock	Trigger	Synchronous Clock	Trigger	Synchronous Clock	Trigger
GPIO(49:32)	–	–	–	–	–	–	–	–
GPIO(31:16)	–	–	–	–	–	–	TClock	**Note 2**
GPIO15/DCD#	RTC	Falling edge	RTC	Falling edge	MasterOut	**Note 2**	MasterOut	**Note 2**
GPIO(14:13)	–	–	–	–	MasterOut	**Note 2**	MasterOut	**Note 2**
GPIO(12:9)	–	–	RTC	**Note 1**	MasterOut	**Note 2**	MasterOut	**Note 2**
GPIO(8:4)	–	–	–	–	–	–	TClock	**Note 2**
GPIO3	RTC	Falling edge	RTC	**Note 1**	RTC	**Note 2**	RTC	**Note 2**
GPIO(2:0)	–	–	RTC	**Note 1**	RTC	**Note 2**	RTC	**Note 2**

Notes 1. Selected from the falling edge or rising edge (set by PMU).

2. Selected from the signal change (falling or rising edge), low level, or high level (set by GIU).

19.2 Register Set

The GIU registers are listed below.

Table 19-2. GIU Registers

Address	R/W	Register Symbols	Function
0x0B00 0100	R/W	GIUIOSELL	GPIO Input/Output Select Register L
0x0B00 0102	R/W	GIUIOSELH	GPIO Input/Output Select Register H
0x0B00 0104	R/W	GIUPIODL	GPIO Port Input/Output Data Register L
0x0B00 0106	R/W	GIUPIODH	GPIO Port Input/Output Data Register H
0x0B00 0108	R/W	GIUINTSTATL	GPIO Interrupt Status Register L
0x0B00 010A	R/W	GIUINTSTATH	GPIO Interrupt Status Register H
0x0B00 010C	R/W	GIUINTENL	GPIO Interrupt Enable Register L
0x0B00 010E	R/W	GIUINTENH	GPIO Interrupt Enable Register H
0x0B00 0110	R/W	GIUINTTYPL	GPIO Interrupt Type (Edge or Level) Select Register L
0x0B00 0112	R/W	GIUINTTYPH	GPIO Interrupt Type (Edge or Level) Select Register H
0x0B00 0114	R/W	GIUINTALSELL	GPIO Interrupt Active Level Select Register L
0x0B00 0116	R/W	GIUINTALSELH	GPIO Interrupt Active Level Select Register H
0x0B00 0118	R/W	GIUINTHTSELL	GPIO Interrupt Hold/Through Select Register L
0x0B00 011A	R/W	GIUINTHTSELH	GPIO Interrupt Hold/Through Select Register H
0x0B00 011C	R/W	GIUPODATL	GPIO Port Output Data Register L
0x0B00 011E	R/W	GIUPODATH	GPIO Port Output Data Register H
0x0B00 02E0	R/W	GIUUSEUPDN	GPIO Pullup/Down User Register
0x0B00 02E2	R/W	GIUTERMUPDN	GPIO Terminal Pullup/Down Register

19.2.1 GIUIOSELL (0x0B00 0100)

Bit	15	14	13	12	11	10	9	8
Name	IOS15	IOS14	IOS13	IOS12	IOS11	IOS10	IOS9	IOS8
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	IOS7	IOS6	IOS5	IOS4	IOS3	IOS2	IOS1	IOS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15	IOS15	GPIO15 (DCD#) pin input/output select 1: RFU 0: Input
14:0	IOS(14:0)	GPIO(14:0) pin input/output select 1: Output 0: Input

This register is used to set input/output modes for GPIO(15:0) pins. The IOS(15:0) bits correspond to the GPIO(15:0) pins.

When the IOS bit is set to "1", the corresponding GPIO pin is set for output and the value that has been written to the corresponding PIOD bit in the GIUPIODL (GPIO Port Input/Output Data Register) is output.

When this bit is set to "0", the corresponding GPIO pin is set for input.

Caution Since GPIO15 (DCD#) is fixed as input, IOS15 cannot be set for output.

19.2.2 GIUIOSELH (0x0B00 0102)

Bit	15	14	13	12	11	10	9	8
Name	IOS31	IOS30	IOS29	IOS28	IOS27	IOS26	IOS25	IOS24
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	IOS23	IOS22	IOS21	IOS20	IOS19	IOS18	IOS17	IOS16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:0	IOS(31:16)	GPIO(31:16) pin input/output select 1: Output 0: Input

This register is used to set input/output modes for GPIO(31:16) pins. The IOS(31:16) pins correspond to the GPIO(31:16) pins.

When the IOS bit is set to "1", the corresponding GPIO pin is set for output and the value that has been written to the corresponding PIOD bit in the GIUPIODH (GPIO Port Input/Output Data Register) is output.

When this bit is set to "0", the corresponding GPIO pin is set for input.

19.2.3 GIUPIODL (0x0B00 0104)

Bit	15	14	13	12	11	10	9	8
Name	PIOD15	PIOD14	PIOD13	PIOD12	PIOD11	PIOD10	PIOD9	PIOD8
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	PIOD7	PIOD6	PIOD5	PIOD4	PIOD3	PIOD2	PIOD1	PIOD0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15	PIOD15	GPIO15 (DCD#) pin output data specification 1: RFU 0: Low
14:0	PIOD(14:0)	GPIO(14:0) pin output data specification 1: High 0: Low

This register is used to read GPIO pins and write data. The PIOD(15:0) bits correspond to the GPIO(15:0) pins.

When "1" is set to the corresponding IOS bit in the GIUIOSELL register (GPIO Input/Output Select Register), the data written to the PIOD bit is output via the corresponding GPIO pin.

When the value of the corresponding IOS bit in the GIUIOSELL register (GPIO Input/Output Select Register) is "0", writing a value to the PIOD bit does not affect the GPIO pin (the write data is ignored).

When the value of the IOS bit in the GIUIOSELL register (GPIO Input/Output Select Register) is "0", reading the PIOD bit enables the corresponding GPIO pin's state to be read.

Caution Since GPIO15 (DCD#) pin is fixed as input, data cannot be written to PIOD15.

19.2.4 GIUPIODH (0x0B00 0106)

Bit	15	14	13	12	11	10	9	8
Name	PIOD31	PIOD30	PIOD29	PIOD28	PIOD27	PIOD26	PIOD25	PIOD24
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	PIOD23	PIOD22	PIOD21	PIOD20	PIOD19	PIOD18	PIOD17	PIOD16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:0	PIOD(31:16)	GPIO(13:16) pin output data specification 1: High 0: Low

This register is used to read GPIO pins and write data. The PIOD(31:16) bits correspond to the GPIO(31:16) pins.

When "1" is set to the corresponding IOS bit in the GIUIOSELH register (GPIO Input/Output Select Register), the data written to the PIOD bit is output via the corresponding GPIO pin.

When the value of the corresponding IOS bit in the GIUIOSELH register (GPIO Input/Output Select Register) is "0", writing a value to the PIOD bit does not affect the GPIO pin (the write data is ignored).

When the value of the IOS bit in the GIUIOSELH register (GPIO Input/Output Select Register) is "0", reading the PIOD bit enables the corresponding GPIO pin's state to be read.

19.2.5 GIUINTSTATL (0x0B00 0108)

Bit	15	14	13	12	11	10	9	8
Name	INTS15	INTS14	INTS13	INTS12	INTS11	INTS10	INTS9	INTS8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	INTS7	INTS6	INTS5	INTS4	INTS3	INTS2	INTS1	INTS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:0	INTS(15:0)	Interrupt to GPIO(15:0) pin. Cleared to 0 when 1 is written. 1: Interrupt occurred 0: No interrupt

This register indicates the interrupt status of GPIO pins.

The INTS(15:0) bits correspond to the GPIO(15:0) pins.

"1" is set to the corresponding INTS bit when the signal input to the GPIO pin meets the condition set via the GIUINTTYPL register (0x0B00 0110: GPIO Interrupt Type (Edge or Level) Select Register) or the GIUINTALSELL register (0x0B00 0114: GPIO Interrupt Active Level Select Register).

Even if the corresponding bit is set to "1", however, no interrupt occurs when the GIUINTENL register (0x0B00 010C: GPIO Interrupt Enable Register) is set to prohibit interrupt.

If a GIPO pin is low at default status, it is judged that the condition is met, and the corresponding bit is set to "1".

When using this register, it should be cleared to 0 once after the GIUINTTYPL and GIUINTALSELL registers are set to enable interrupt.

Caution The function of GPIO15 is fixed as the DCD# signal input.

19.2.6 GIUINTSTATH (0x0B00 010A)

Bit	15	14	13	12	11	10	9	8
Name	INTS31	INTS30	INTS29	INTS28	INTS27	INTS26	INTS25	INTS24
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	INTS23	INTS22	INTS21	INTS20	INTS19	INTS18	INTS17	INTS16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:0	INTS(31:16)	Interrupt to GPIO(31:16) pin. Cleared to 0 when 1 is written. 1: Interrupt occurred 0: No interrupt

This register indicates the interrupt status of GPIO pins.

The INTS(31:16) bits correspond to the GPIO(31:16) pins.

"1" is set to the corresponding INTS bit when the signal input to the GPIO pin meets the condition set via the GIUINTTYPH register (0x0B00 0112: GPIO Interrupt Type (Edge or Level) Select Register) or GIUINTALSELH register (0x0B00 0116: GPIO Interrupt Active Level Select Register).

Even if the corresponding bit is set to "1", however, no interrupt occurs when the GIUINTENH register (0x0B00 010E: GPIO Interrupt Enable Register) is set to prohibit interrupt.

If a GIPO pin is low at default status, it is judged that the condition is met, and the corresponding bit is set to "1".

When using this register, it should be cleared to 0 once after the GIUINTTYPH and GIUINTALSELH registers are set to enable interrupt.

19.2.7 GIUINTENL (0x0B00 010C)

Bit	15	14	13	12	11	10	9	8
Name	INTE15	INTE14	INTE13	INTE12	INTE11	INTE10	INTE9	INTE8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	INTE7	INTE6	INTE5	INTE4	INTE3	INTE2	INTE1	INTE0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:0	INTE(15:0)	Interrupt enable to GPIO(15:0) pin 1: Interrupt enable 0: Interrupt prohibit

This register is used to set interrupt enable status for GPIO pins. The INTE(15:0) bits correspond to the GPIO(15:0) pins.

When "1" is set to the corresponding INTE bit, interrupts are enabled for the corresponding GPIO pins.

Caution The function of GPIO15 is fixed as the DCD# signal input.

19.2.8 GIUINTENH (0x0B00 010E)

Bit	15	14	13	12	11	10	9	8
Name	INTE31	INTE30	INTE29	INTE28	INTE27	INTE26	INTE25	INTE24
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	INTE23	INTE22	INTE21	INTE20	INTE19	INTE18	INTE17	INTE16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:0	INTE(31:16)	Interrupt enable to GPIO(31:16) pin 1: Interrupt enable 0: Interrupt prohibit

This register is used to set interrupt enable status for GPIO pins. The INTE(31:16) bits correspond to the GPIO(31:16) pins.

When "1" is set to the corresponding INTE bit, interrupts are enabled for the corresponding GPIO pins.

19.2.9 GIUINTTYPL (0x0B00 0110)

Bit	15	14	13	12	11	10	9	8
Name	INTT15	INTT14	INTT13	INTT12	INTT11	INTT10	INTT9	INTT8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	INTT7	INTT6	INTT5	INTT4	INTT3	INTT2	INTT1	INTT0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:0	INTT(15:0)	Interrupt detection trigger 1: Edge 0: Level

This register is used to set the trigger to detect an interrupt request for GPIO pins. The INTT(15:0) bits correspond to the GPIO(15:0) pins.

When "1" is set to the corresponding INTT bit, the edge detection method is used for the interrupt signal at the corresponding GPIO pin (an interrupt is triggered when the signal state changes from low to high or from high to low).

The level detection method is used when "0" is set, in which case the level set to the corresponding bit in the GIUINTALSELL register (GPIO Interrupt Active Level Select Register) is detected.

Caution The function of GPIO15 is fixed as DCD# signal input.

19.2.10 GIUINTTYPH (0x0B00 0112)

Bit	15	14	13	12	11	10	9	8
Name	INTT31	INTT30	INTT29	INTT28	INTT27	INTT26	INTT25	INTT24
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	INTT23	INTT22	INTT21	INTT20	INTT19	INTT18	INTT17	INTT16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:0	INTT(31:16)	Interrupt detection trigger 1: Edge 0: Level

This register is used to set the detection method for interrupts to GPIO pins. The INTT(31:16) bits correspond to the GPIO(31:16) pins.

When "1" is set to the corresponding INTT bit, the edge detection method is used for the interrupt request signal at the corresponding GPIO pin (an interrupt request is triggered when the signal state changes from low to high or from high to low).

The level detection method is used when "0" is set, in which case the level set to the corresponding bit in the GIUINTALSELH register (GPIO Interrupt Active Level Select Register) is detected.

19.2.11 GIUINTALSELL (0x0B00 0114)

Bit	15	14	13	12	11	10	9	8
Name	INTL15	INTL14	INTL13	INTL12	INTL11	INTL10	INTL9	INTL8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	INTL7	INTL6	INTL5	INTL4	INTL3	INTL2	INTL1	INTL0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:0	INTL(15:0)	Interrupt request detection level 1: High active 0: Low active

This register is used to set the active level when using the level detection method for interrupts to GPIO pins. The INTL(15:0) bits correspond to the GPIO(15:0) pins.

The contents of this register are not reflected when the edge detection method is selected via the GIUINTTYPL register (GPIO Interrupt Type (Edge or Level) Select Register). When using this register, be sure to set the level detection method via the GIUINTTYPL register (GPIO Interrupt Type (Edge or Level) Select Register).

Caution The function of GPIO15 is fixed as the DCD# signal input.

19.2.12 GIUINTALSELH (0x0B00 0116)

Bit	15	14	13	12	11	10	9	8
Name	INTL31	INTL30	INTL29	INTL28	INTL27	INTL26	INTL25	INTL24
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	INTL23	INTL22	INTL21	INTL20	INTL19	INTL18	INTL17	INTL16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:0	INTL(31:16)	Interrupt request detection level 1: High active 0: Low active

This register is used to set the active level when using the level detection method for interrupts to GPIO pins. The INTL(31:16) bits correspond to the GPIO(31:16) pins.

The contents of this register are not reflected when the edge detection method is selected via the GIUINTTYPH register (GPIO Interrupt Type (Edge or Level) Select Register). When using this register, be sure to set the level detection method via the GIUINTTYPH register (GPIO Interrupt Type (Edge or Level) Select Register).

19.2.13 GIUINTHTSELL (0x0B00 0118)

Bit	15	14	13	12	11	10	9	8
Name	INTH15	INTH14	INTH13	INTH12	INTH11	INTH10	INTH9	INTH8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	INTH7	INTH6	INTH5	INTH4	INTH3	INTH2	INTH1	INTH0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:0	INTH(15:0)	GPIO(15:0) pin interrupt signal hold/through 1: Hold 0: Through

This register is used to set whether or not interrupt signals to the GPIO pins should be held. The INTH(15:0) bits correspond to the GPIO(15:0) pins.

When "1" is set to the corresponding INTH bit, any interrupt signal input to the corresponding GPIO pin is held.

When "0" is set to this bit, any interrupt signal input to the corresponding GPIO pin is not held and is instead allowed to pass through.

Any held interrupt signal is cleared when "1" is set to the corresponding bit in the GIUINTSTATL register (GPIO Interrupt Status Register).

INTH(15:0) are not affected by GIUINTENL (interrupt enable register).

If "1" (hold) is set to the INTH bit while the interrupt enable bit is set to 0 (prohibit interrupts), any change in the pin state is retained as change data. Therefore, an interrupt still occurs when the interrupt enable bit is again set to enable interrupts.

Caution The function of GPIO15 is fixed as the DCD# signal input.

19.2.14 GIUINTHTSELH (0x0B00 011A)

Bit	15	14	13	12	11	10	9	8
Name	INTH31	INTH30	INTH29	INTH28	INTH27	INTH26	INTH25	INTH24
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	INTH23	INTH22	INTH21	INTH20	INTH19	INTH18	INTH17	INTH16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:0	INTH(31:16)	GPIO(31:16) pin interrupt signal hold/through 1: Hold 0: Through

This register is used to set whether or not interrupt signals to the GPIO pins should be held. The INTH(31:16) bits correspond to the GPIO(31:16) pins.

When "1" is set to the corresponding INTH bit, any interrupt signal input to the corresponding GPIO pin is held.

When "0" is set to this bit, any interrupt signal input to the corresponding GPIO pin is not held and is instead allowed to pass through.

Any held interrupt signal is cleared when "1" is set to the corresponding bit in the GIUINTSTATH register (GPIO Interrupt Status Register).

INTH(31:16) are not affected by GIUINTENH (interrupt enable register).

If "1" (hold) is set to the INTH bit while the interrupt enable bit is set to 0 (prohibit interrupts), any change in the pin state is retained as change data. Therefore, an interrupt still occurs when the interrupt enable bit is again set to enable interrupts.

The relationship between settings of GPIO interrupts enable/prohibit and hold/through is as below.

Table 19-3. Correspondences Between Interrupt Mask and Interrupt Hold

Interrupt Trigger	Setting of GIUINTHSEL	Setting of GIUINTEN	Hold in GIU	Notation to ICU
Level	Hold	Masked	Held	Not noticed
		Not masked	Held	Noticed
		Masked → canceled	Held	Noticed
	Through	Masked	Through	Not noticed
		Not masked	Through	Noticed
		Masked → canceled	Through	Not noticed
Edge	Hold	Masked	Held	Not noticed
		Not masked	Held	Noticed
		Masked → canceled	Held	Noticed
	Through	Masked	Through	Not noticed
		Not masked	Prohibited	Prohibited
		Masked → canceled	Through	Not noticed

19.2.15 GIUPODATL (0x0B00 011C)

Bit	15	14	13	12	11	10	9	8
Name	PIOD47	PIOD46	PIOD45	PIOD44	PIOD43	PIOD42	PIOD41	PIOD40
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	1	1	1	1
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	7	6	5	4	3	2	1	0
Name	PIOD39	PIOD38	PIOD37	PIOD36	PIOD35	PIOD34	PIOD33	PIOD32
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	1	1	1	1	1	1	1	1
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15:0	PIOD(47:32)	GPIO(47:32) pin output data specification 1: High 0: Low

Note Previous value is retained.

This register is used to set the output level for GPIO(47:32) pins. The PIOD(47:32) bits correspond to the GPIO(47:32) pins.

The data written to the PIOD bit is output via the corresponding GPIO pin. The set value can be read by reading the PIOD bit.

Pins set by this register are output-only. Pins set by this register are used exclusively from other function pins. Therefore, when using this register, set the enable bit to prohibit in the corresponding unit.

The correspondences between PIOD bits and function pins are listed in the table on the next page.

Table 19-4. Correspondences Between GPIO(47:32) and Alternate Function

GPIO Pin	PIOD Bit	Alternate Function
GPIO47	PIOD47	DCTS#
GPIO46	PIOD46	DRTS#
GPIO45	PIOD45	DDIN
GPIO44	PIOD44	DDOUT
GPIO43	PIOD43	KSCAN11
GPIO42	PIOD42	KSCAN10
GPIO41	PIOD41	KSCAN9
GPIO40	PIOD40	KSCAN8
GPIO39	PIOD39	KSCAN7
GPIO38	PIOD38	KSCAN6
GPIO37	PIOD37	KSCAN5
GPIO36	PIOD36	KSCAN4
GPIO35	PIOD35	KSCAN3
GPIO34	PIOD34	KSCAN2
GPIO33	PIOD33	KSCAN1
GPIO32	PIOD32	KSCAN0

19.2.16 GIUPODATH (0x0B00 011E)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	PIOEN1	PIOEN0
R/W	R	R	R	R	R	R	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	PIOD49	PIOD48
R/W	R	R	R	R	R	R	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15:0	RFU	Write 0 to these bits. 0 is returned after a read.
9	PIOEN1	GPIO49/SMODE1 pin output control 1: Enable 0: Disable
8	PIOEN0	GPIO48/DBUS32 pin output control 1: Enable 0: Disable
7:2	RFU	Write 0 to these bits. 0 is returned after a read.
1:0	PIOD(49:48)	GPIO(49:48) pin output data specification 1: High 0: Low

Note Previous value is retained.

This register is used to enable/disable the output to the GPIO(49:48) pins and to set the output level for GPIO(49:48) pins. The PIOEN(1:0) bits or the PIOD(49:48) bits correspond to the GPIO(49:48).

The data written to the PIOD bit is output via the corresponding GPIO pin. The set value can be read by reading the PIOD bit. The GPIO(49:48) pins support output-only. However, these pins also serve as initial setting pins and change to the input state during RTC reset.

The correspondence between GPIO pins and alternate functions is shown below.

Table 19-5. Correspondence Between GPIO(49:48) and Alternate Function

GPIO Pin	PIOD Bit	Alternate Function
GPIO49	PIOD49	SMODE1
GPIO48	PIOD48	DBUS32

19.2.17 GIUUSEUPDN (0x0B00 02E0)

Bit	15	14	13	12	11	10	9	8
Name	RFU	USEUPDN 14	USEUPDN 13	USEUPDN 12	USEUPDN 11	USEUPDN 10	USEUPDN 9	USEUPDN 8
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	7	6	5	4	3	2	1	0
Name	USEUPDN 7	USEUPDN 6	USEUPDN 5	USEUPDN 4	USEUPDN 3	USEUPDN 2	USEUPDN 1	USEUPDN 0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15	RFU	Write 0 to this bit. 0 is returned after a read.
14:0	USEUPDN(14:0)	GPIO(14:0) pin pull-up/pull-down usage specification 1: Use pull-up/pull-down function 0: Does not use pull-up/pull-down function

Note Previous value is retained.

This register is used to specify whether the pull-up/pull-down function is used or not for the GPIO(14:0) pins. The USEUPDN(14:0) bits correspond to the GPIO(14:0) pins.

Setting "1" to the USEUPDN bit enables to use the pull-up/pull-down function for the corresponding GPIO pin.

The setting of the corresponding TERMUPDN bit of the GIUTERMUPDN register (0x0B00 02E2: GPIO Terminal Pullup/Down Register) determines whether the corresponding pin is pulled up or pulled down.

This function is valid only when all the bits of the GIUIOSELL register (0x0B00 0100: GPIO Input/Output Select Register) are set to 0 (setting for input).

Caution GPIO15 is not provided with this function because its function is fixed as the DCD# signal input.

19.2.18 GIUTERMUPDN (0x0B00 02E02)

Bit	15	14	13	12	11	10	9	8
Name	RFU	TERM UPDN14	TERM UPDN13	TERM UPDN12	TERM UPDN11	TERM UPDN0	TERM UPDN9	TERM UPDN8
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	7	6	5	4	3	2	1	0
Name	TERM UPDN7	TERM UPDN6	TERM UPDN5	TERM UPDN4	TERM UPDN3	TERM UPDN2	TERM UPDN1	TERM UPDN0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15	RFU	Write 0 to this bit. 0 is returned after a read.
14:0	TERMUPDN(14:0)	GPIO(14:0) pin pull-up/pull-down selection 1: Pull-up 0: Pull-down

Note Previous value is retained.

This register is used to specify whether the pull-up or pull-down function is used for the GPIO(14:0) pins. The TERMUPDN(14:0) bits correspond to the GPIO(14:0) pins.

When the corresponding bit of the GIUUSEUPNL register (0x0B00 02E0: GPIO Pullup/Down Register) is 1, setting "1" to the TERMUPDN bit specifies the pull-up function for the corresponding pin, and setting "0" does the pull-down function.

Caution GPIO15 is not provided with this function because its function is fixed as the DCD# signal input.

CHAPTER 20 PIU (TOUCH PANEL INTERFACE UNIT)

20.1 General

The PIU uses an on-chip A/D converter and detects the X and Y coordinates of pen contact locations on the touch panel and scans the general-purpose A/D input port. Since the touch panel control circuit and the A/D converter (conversion precision: 10 bits) are both on-chip, the touch panel is connected directly to the VR4121.

The PIU's function, namely the detection of X and Y coordinates, is performed partly by hardware and partly by software.

Hardware tasks:
- Touch panel applied voltage control
- Reception of coordinate data

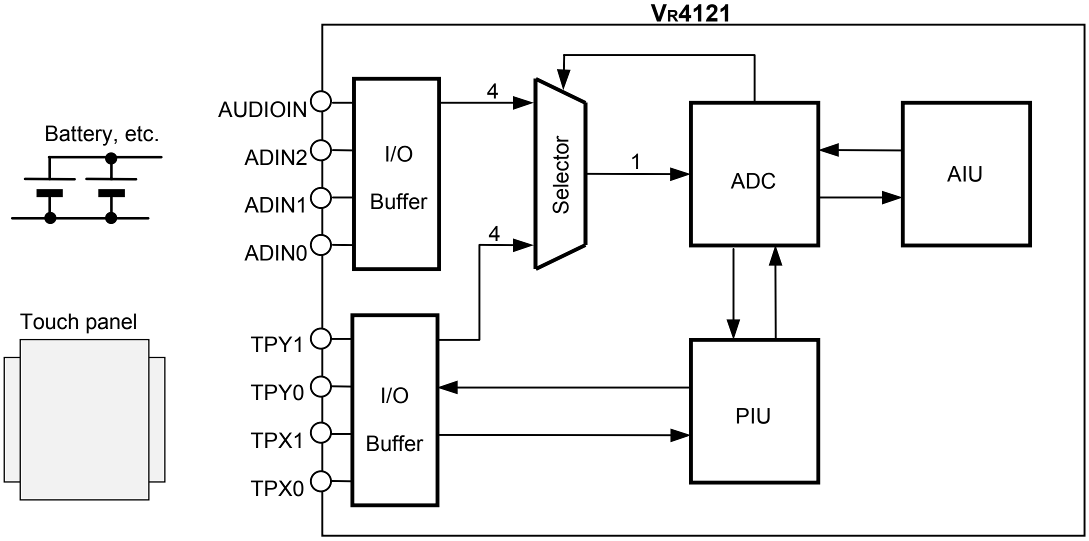
Software task:
- Processing of coordinate data based on data sampled by hardware

Features of the PIU's hardware tasks are described below.

- Can be directly connected to touch panel with four-pin resistance layers (on-chip touch panel driver)
- Interface for on-chip A/D converter
- Voltage detection at three general-purpose AD ports and one audio input port
- Operation of A/D converter based on various settings and control of voltage applied to touch panel
- Sampling of X-coordinate and Y-coordinate data
- Variable coordinate data sampling interval
- Interrupt is triggered if pen touch occurs regardless of CPU operation mode (interrupts do not occur when in CPU hibernate mode)
- Four dedicated buffers for up to two pages each of coordinate data
- Four buffers for A/D port scan
- Auto/manual options for coordinate data sampling start/stop control

20.1.1 Block diagrams

Figure 20-1. PIU Peripheral Block Diagram



- **Touch panel**

  A set of four pins are located at the edges of the X-axis and Y-axis resistance layers, and the two layers have high resistance when there is no pen contact and low resistance when there is pen contact. The resistance between the two edges of the resistance layers is about 1 kΩ. When a voltage is applied to both edges of the Y-axis resistance layer, the voltage (V_{Y1} and V_{Y2} in the figure below) is measures at the X-axis resistance layer's pins to determine the Y coordinate. Similarly, when a voltage is applied to both edges of the X-axis resistance layer, the voltage (V_{X1} and V_{X2} in the figure below) is measures at the Y-axis resistance layer's pins to determine the X coordinate. For greater precision, voltage applied to individual resistance-layer pins can be measured to obtain X and Y coordinate data based on four voltage measurements. The obtained data is stored into the PIUBPnmREG register (n = 0 or 1, m = 0 to 3).

Figure 20-2. Coordinate Detection Equivalent Circuits

(a) Y-coordinate detection

(b) X-coordinate detection

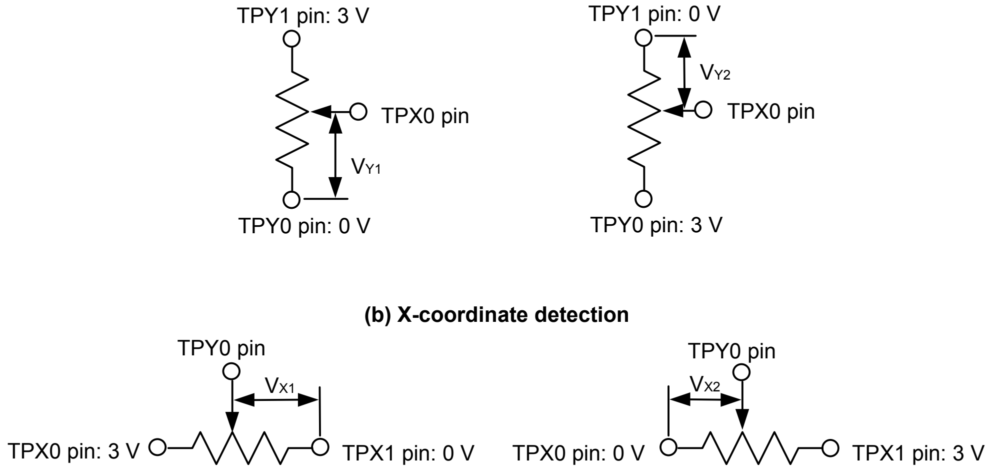
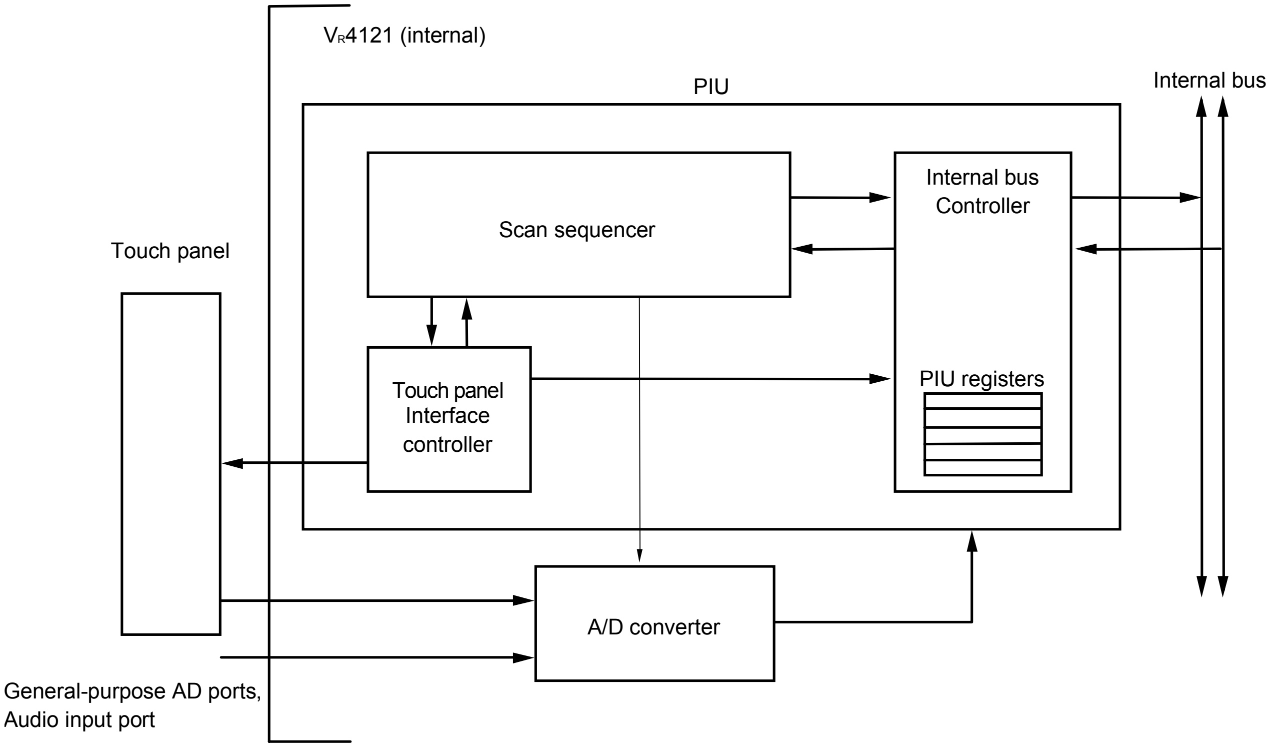


Figure 20-3. Internal Block Diagram of PIU



The PIU includes three blocks: an internal bus controller, a scan sequencer, and a touch panel interface controller.

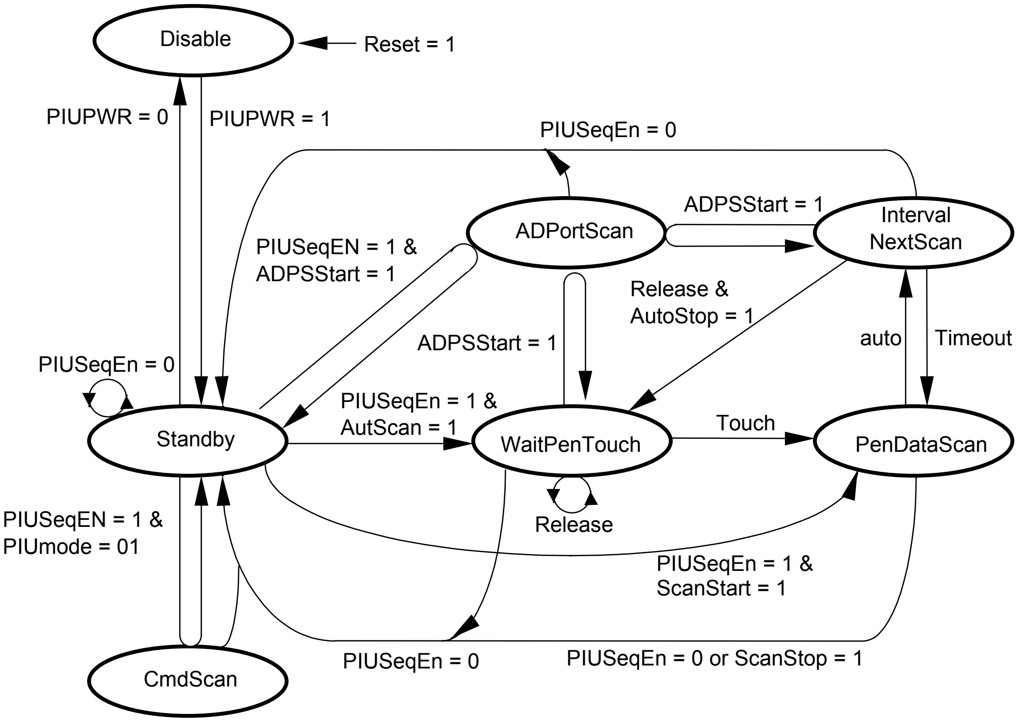
(1) Internal bus controller
The internal bus controller controls the internal bus, the PIU registers, and interrupts and performs serial/parallel conversion of data from the A/D converter.

(2) Scan sequencer
The scan sequencer is used for PIU state management.

(3) Touch panel interface controller
The touch panel interface controller is used to control the touch panel.

20.2 Scan Sequencer State Transition

Figure 20-4. Scan Sequencer State Transition Diagram



★ **(1) Disable state**

In this state, the A/D converter is in standby mode, the output pins are in touch detection mode and the input pins are in mask mode (to prevent misoperation when an undefined input is applied).

State transition to suspend mode is possible, however, it is necessary to wait for the time set by STABLE(5:0) in the PIUSTBLREG to ensure stabilization.

(2) Standby state

In this state, the unit is in scan idle mode. The touch panel is in low-power mode (0-V voltage is applied to the touch panel and the A/D converter is in disable mode). Normally, this is the state from which various mode settings are made.

Caution State transitions occur when the PIUSEQEN bit is active, so the PIUSEQEN bit must be set as active after each mode setting has been completed.

(3) ADPortScan state

This is the state in which voltage is measured at the A/D converter's three general-purpose ports and one audio input port. After the A/D converter is activated and voltage data is obtained, the data is stored in the PIU's internal data buffer (PIUABxREG). After the four ports are scanned, a PadCMDIntr interrupt occurs. After this interrupt occurs, the ADPSSTART bit is automatically set as inactive and the state changes to the state in which the ADPSSTART bit was active.

(4) CmdScan state

When in this state, the A/D converter operates using various settings. Voltage data from one port only is fetched based on a combination of the touch panel pin setting (TPX(1:0), TPY(1:0)) and the selection of an input port (TPX(1:0), TPY(1:0), AUDIOIN, ADIN(2:0)) to the A/D converter. Use PIUCMDREG to make the touch panel pin setting and to select the input port.

(5) WaitPenTouch state

This is the standby state that waits for a touch panel "touch" state. When the PIU detects a touch panel "touch" state, PenChgIntr (an internal interrupt in the PIU) occurs. At this point, if the PADATSCAN bit is active, the state changes to the PenDataScan state.

★ In the WaitPenTouch state, it is possible to change to Suspend mode, however, the TClock stops and panel status detection is not performed.

(6) PenDataScan state

This is the state in which touch panel coordinates are detected. The A/D converter is activated and the four sets of data for each coordinate are sampled.

Caution If one complete pair of coordinates is not obtained during the interval between one pair of coordinates and the next coordinate data, a PadDataLostIntr interrupt occurs.

(7) IntervalNextScan state

This is the standby state that waits for the next coordinate sampling period and the touch panel's "Release" state. After the touch panel state is detected, the time period specified via PIUSIVLREG elapses before the transition to the PenDataScan state. If the PIU detects the "Release" state within the specified time period, PenChgIntr (an internal interrupt in the PIU) occurs. At this point, the state changes to the WaitPenTouch state if the PADAUTOSTOP bit is active. If the PADATSTOP bit is inactive, it changes to the PenDataScan state after the specified time period has elapsed.

20.3 Register Set

The PIU registers are listed below.

Table 20-1. PIU Registers

Address	R/W	Register Symbols	Function
0x0B00 0122	R/W	PIUCNTREG	PIU Control register
0x0B00 0124	R/W	PIUINTREG	PIU Interrupt cause register
0x0B00 0126	R/W	PIUSIVLREG	PIU Data sampling interval register
0x0B00 0128	R/W	PIUSTBLREG	PIU A/D converter start delay register
0x0B00 012A	R/W	PIUCMDREG	PIU A/D command register
0x0B00 0130	R/W	PIUASCNREG	PIU A/D port scan register
0x0B00 0132	R/W	PIUAMSKREG	PIU A/D scan mask register
0x0B00 013E	R	PIUCIVLREG	PIU Check interval register
0x0B00 02A0	R/W	PIUPB00REG	PIU Page 0 Buffer 0 register
0x0B00 02A2	R/W	PIUPB01REG	PIU Page 0 Buffer 1 register
0x0B00 02A4	R/W	PIUPB02REG	PIU Page 0 Buffer 2 register
0x0B00 02A6	R/W	PIUPB03REG	PIU Page 0 Buffer 3 register
0x0B00 02A8	R/W	PIUPB10REG	PIU Page 1 Buffer 0 register
0x0B00 02AA	R/W	PIUPB11REG	PIU Page 1 Buffer 1 register
0x0B00 02AC	R/W	PIUPB12REG	PIU Page 1 Buffer 2 register
0x0B00 02AE	R/W	PIUPB13REG	PIU Page 1 Buffer 3 register
0x0B00 02B0	R/W	PIUAB0REG	PIU A/D scan Buffer 0 register
0x0B00 02B2	R/W	PIUAB1REG	PIU A/D scan Buffer 1 register
0x0B00 02B4	R/W	PIUAB2REG	PIU A/D scan Buffer 2 register
0x0B00 02B6	R/W	PIUAB3REG	PIU A/D scan Buffer 3 register
0x0B00 02BC	R/W	PIUPB04REG	PIU Page 0 Buffer 4 register
0x0B00 02BE	R/W	PIUPB14REG	PIU Page 1 Buffer 4 register

These registers are described in detail below.

20.3.1 PIUCNTREG (0x0B00 0122)

(1/2)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	PENSTC	PADSTATE 2	PADSTATE 1	PADSTATE 0	PADAT STOP	PADAT START
R/W	R	R	R	R	R	R	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	PADSCAN STOP	PADSCAN START	PADSCAN TYPE	PIUMODE1	PIUMODE0	PIUSEQEN	PIUPWR	PADRST
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:14	RFU	Write 0 to these bits. 0 is returned after a read.
13	PENSTC	Touch/release when touch panel contact state changes 1: Touch 0: Release
12:10	PADSTATE	Scan sequencer status 111: CmdScan 110: IntervalNextScan 101: PenDataScan 100: WaitPenTouch 011: RFU 010: ADPortScan 001: Standby 000: Disable
9	PADATSTOP	Sequencer auto stop setting during touch panel release state 1: Auto stop after sampling data for one set of coordinates during release state 0: No auto stop (even during release state)
8	PADATSTART	Sequencer auto start setting during touch panel touch state 1: Auto start during touch state 0: No auto start during touch state
7	PADSCANSTOP	Forced stop setting for touch panel sequencer 1: Forced stop after sampling data for one set of coordinates 0: Do not stop

(2/2)

Bit	Name	Function
6	PADSCANSTART	Start setting for touch panel sequencer 1: Forced start 0: Do not start
5	PADSCANTYPE	Touch pressure sampling enable 1: Enable 0: Prohibit
4:3	PIUMODE(1:0)	PIU mode setting 11: RFU 10: RFU 01: Operate A/D converter using any command 00: Sample coordinate data
2	PIUSEQEN	Scan sequencer operation enable 1: Enable 0: Prohibit
1	PIUPWR	PIU power mode setting 1: Set PIU output as active and change to standby mode 0: Set panel to touch detection state and shift to PIU operation stop enabled mode
0	PADRST	PIU reset. Once the PADRST bit is set to "1", it is automatically cleared to 0 after four TClock cycles. 1: Reset 0: Normal

This register is used to make various settings for the PIU.

The PENSTC bit indicates the touch panel contact state at the time when the PENCHGINTR bit of PIUINTREG is set to 1. This bit's state remains as it is until PENCHGINTR is cleared to 0. Also, when PENCHGINTR is cleared to 0, PENSTC indicates the touch panel contact state. However, PENSTC does not change while PENCHGINTR is set to 1, even if the touch panel contact state changes between release and touch.

Some bits in this register cannot be set in a specific state of scan sequencer. The combination of the setting of this register and the sequencer state is as follows.

Table 20-2. PIUCNTREG Bit Manipulation and States

PIUCNTREG Bit Manipulation		Scan Sequencer's State			
		Disable	Standby	WaitPenTouch	PenDataScan
PADRST[Note 1]	0 → 1	–	Disable	Disable	Disable
PIUPWR	0 → 1	Standby	?	×	×
	1 → 0	?	Disable	×	×
PIUSEQEN	0 → 1	×	WaitPenTouch	?	?
	1 → 0	?	?	Standby	Standby
PADATSTART	0 → 1	×	–	PenDataScan[Note 2]	×
	1 → 0	×	–	–	×
PADATSTOP	0 → 1	×	–	×	×
	1 → 0	×	–	×	×
PADSCANSTART	0 → 1	×	PenDataScan[Note 3]	×	×
	1 → 0	×	–	×	×
PADSCANSTOP	0 → 1	×	–	×	Standby[Note 4]
	1 → 0	×	–	×	–

PIUCNTREG Bit Manipulation		Scan Sequencer's State		
		IntervalNextScan	ADPortScan	CmdScan
PADRST[Note 1]	0 → 1	Disable	Disable	Disable
PIUPWR	0 → 1	?	?	?
	1 → 0	×	×	×
PIUSEQEN	0 → 1	?	?	?
	1 → 0	Standby	Standby	Standby
PADATSTART	0 → 1	×	×	×
	1 → 0	×	×	×
PADATSTOP	0 → 1	×	×	×
	1 → 0	×	×	×
PADSCANSTART	0 → 1	×	×	×
	1 → 0	×	×	×
PADSCANSTOP	0 → 1	Standby[Note 4]	Standby[Note 4]	Standby[Note 4]
	1 → 0	?	–	–

Notes 1. After "1" is written, the bit is automatically cleared to 0 after four TClock cycles.

2. State transition occurs during touch state

3. State transition occurs when PIUSEQEN = 1

4. State transition occurs after one set of data is sampled. This bit is cleared to 0 after the state transition occurs.

Remark –: The bit change is retained but there is no state transition.
×: Setting prohibited (operation not guaranteed)
?: Combination of state and bit status before setting does not exist

20.3.2 PIUINTREG (0x0B00 0124)

Bit	15	14	13	12	11	10	9	8
Name	OVP	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R/W	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	PADCMD INTR	PADADP INTR	PADPAGE1 INTER	PADPAGE0 INTER	PADDLOST INTR	RFU	PENCHG INTR
R/W	R	R/W	R/W	R/W	R/W	R/W	R	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15	OVP	Valid page ID bit (older valid page) 1: Valid data older than page 1 buffer data is retained 0: Valid data older than page 0 buffer data is retained
14:7	RFU	Write 0 to these bits. 0 is returned after a read.
6	PADCMDINTR	PIU command scan interrupt. Cleared to 0 when 1 is written. 1: Indicates that command scan found valid data 0: Indicates that command scan did not find valid data in buffer
5	PADADPINTR	PIU A/D port scan interrupt . Cleared to 0 when 1 is written. 1: Indicates that A/D port scan found valid data with "1" value in buffer 0: Indicates that A/D port scan did not find valid data with "1" value in buffer
4	PADPAGE1INTER	PIU data buffer page 1 interrupt. Cleared to 0 when 1 is written. 1: Valid data with "1" value is stored in page 1 of data buffer 0: No valid data with "1" value in page 1 of data buffer
3	PADPAGE0INTER	PIU data buffer page 0 interrupt. Cleared to 0 when 1 is written. 1: Valid data with "1" value is stored in page 0 of data buffer 0: No valid data with "1" value in page 0 of data buffer
2	PADDLOSTINTR	A/D data timeout. Cleared to 0 when 1 is written. 1: Not data with "1" value found within specified time 0: No timeout
1	RFU	Write 0 to this bit. 0 is returned after a read.
0	PENCHGINTR	Change in touch panel contact state. Cleared to 0 when 1 is written. 1: Change has occurred 0: No change

This register sets and indicates the interrupt request generation of PIU.

When the PENCHGINTR bit is set to1, the PENSTC bit indicates the touch panel contact state (touch or release) when a contact state changes. The PENSTC bit's state remains until PENCHGINTR bit is cleared to 0. Also, when PENCHGINTR is cleared to 0, PENSTC indicates the touch panel contact state. However, PENSTC does not change while PENCHGINTR is set to 1, even if the touch panel contact state changes between release and touch.

Caution In the Hibernate mode, the VR4121 retains the touch panel state. Therefore, if the Hibernate mode has been entered while the touch panel is touched, the contact state may be mistakenly recognized as having changed, when the Fullspeed mode returns.
This may result in PENCHGINTR being set to 1, when a touch panel state change interrupt occurs immediately after the Fullspeed mode returns from the Hibernate mode. Similarly, other bits of PINUINTREG may be set to 1 on returning from the Hibernate mode. Therefore, set each bit of PIUINTREG to 1 to clear an interrupt request, immediately after the Fullspeed mode returns from the Hibernate mode.

20.3.3 PIUSIVLREG (0x0B00 0126)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	SCAN INTVAL10	SCAN INTVAL9	SCAN INTVAL8
R/W	R	R	R	R	R	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

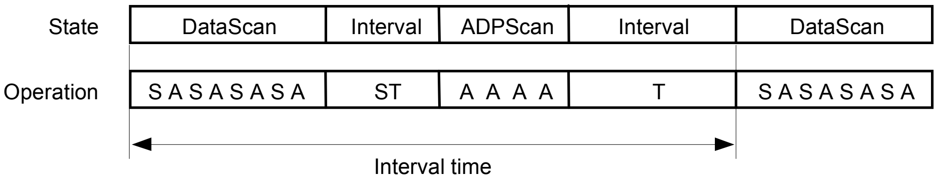
Bit	7	6	5	4	3	2	1	0
Name	SCAN INTVAL7	SCAN INTVAL6	SCAN INTVAL5	SCAN INTVAL4	SCAN INTVAL3	SCAN INTVAL2	SCAN INTVAL1	SCAN INTVAL0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	1	0	1	0	0	1	1	1
After reset	1	0	1	0	0	1	1	1

Bit	Name	Function
15:11	RFU	Write 0 to these bits. 0 is returned after a read.
10:0	SCANINTVAL(10:0)	Coordinate data scan sampling interval setting Interval = SCANINTVAL(10:0) × 30 μs

This register sets the sampling interval for coordinate data sampling.

The sampling interval for one pair of coordinate data is the value set via SCANINTVAL(10:0) multiplied by 30 μs. Accordingly, the logical range of sampling intervals that can be set in 30-μs units is from 0 μs to 60,810 μs (about 60 ms). Actually, if the sampling interval setting is shorter than the time required for obtaining a pair of coordinate data or ADPortScan data, a PIULostIntr interrupt will occur. If PIULostIntr interrupts occur frequently, set a longer interval time.

Figure 20-5. Interval Times and States



Remark S: Voltage stabilization standby time (STABLE(5:0) in PIUSTBLREG)
A: A/D converter (about 10 μs)
T: Touch/release detection

20.3.4 PIUSTBLREG (0x0B00 0128)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	STABLE5	STABLE4	STABLE3	STABLE2	STABLE1	STABLE0
R/W	R	R	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	1	1	1
After reset	0	0	0	0	0	1	1	1

Bit	Name	Function
15:6	RFU	Write 0 to these bits. 0 is returned after a read.
5:0	STABLE(5:0)	Panel applied voltage stabilization standby time (DataScan, CmdScan state) A/D scan timeout time (ADPScan state) Standby time = STABLE(5:0) × 30 μs (Disable, WaitPenTouch, Interval state) During A/D scan, this can be used as a timeout counter.

The voltage stabilization standby time for the voltage applied to the touch panel can be set via STABLE(5:0) in 30-μs units between 0 μs and 1,890 μs.

20.3.5 PIUCMDREG (0x0B00 012A)

(1/2)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	STABLEON	TPYEN1	TPYEN0	TPXEN1	TPXEN0
R/W	R	R	R	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	TPYD1	TPYD0	TPXD1	TPXD0	ADCMD3	ADCMD2	ADCMD1	ADCMD0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	1	1	1	1
After reset	0	0	0	0	1	1	1	1

Bit	Name	Function
15:13	RFU	Write 0 to these bits. 0 is returned after a read.
12	STABLEON	Touch panel applied voltage stabilization time set during command scan (STABLE(5:0) of PIUSTBLREG) enable 1: Retain panel voltage stabilization time 0: Ignore panel voltage stabilization time (voltage stabilization standby time = 0)
11:10	TPYEN(1:0)	TPY port input/output switching during command scan 00: TPY1 input, TPY0 input 01: TPY1 input, TPY0 output 10: TPY1 output, TPY0 input 11: TPY1 output, TPY0 output
9:8	TPXEN(1:0)	TPX port input/output switching during command scan 00: TPX1 input, TPX0 input 01: TPX1 input, TPX0 output 10: TPX1 output, TPX0 input 11: TPX1 output, TPX0 output
7:6	TPYD(1:0)	TPY output level during command scan 00: TPY1 = "L", TPY0 = Low 01: TPY1 = "L", TPY0 = High 10: TPY1 = "H", TPY0 = Low 11: TPY1 = "H", TPY0 = High TPYD value is ignored when TPYEN is set for input.
5:4	TPXD(1:0)	TPX output level during command scan 00: TPX1 = "L", TPX0 = Low 01: TPX1 = "L", TPX0 = High 10: TPX1 = "H", TPX0 = Low 11: TPX1 = "H", TPX0 = High TPXD value is ignored when TPXEN is set for input.

(2/2)

Bit	Name	Function
3:0	ADCMD(3:0)	A/D converter input port selection for command scan 1111: A/D converter standby mode request 1110: RFU : 1000: RFU 0111: AUDIOIN port 0110: ADIN2 port 0101: ADIN1 port 0100: ADIN0 port 0011: TPY1 port 0010: TPY0 port 0001: TPX1 port 0000: TPX0 port

This register switches input/output and sets output level for each port during a command scanning operation.

Setting of the TPYD bit is invalid when the port is set in input mode by the TPYEN bit.

Setting of the TPXD bit is invalid when the port is set in input mode by the TPXEN bit.

20.3.6 PIUASCNREG (0x0B00 0130)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	TPPSCAN	ADPS START
R/W	R	R	R	R	R	R	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:2	RFU	Write 0 to these bits. 0 is returned after a read.
1	TPPSCAN	Port selection for ADPortScan 1: Select TPX(1:0), TPY(1:0) (for touch panel) as A/D port 0: Select ADIN(2:0) (general-purpose) as A/D port and AUDIOIN as audio input port
0	ADPSSTART	ADPortScan start 1: Start ADPortScan 0: Do not perform ADPortScan

This register is used for ADPortScan setting

The ADPortScan begins when the ADPSSTART bit is set. After the ADPortScan is completed, the state returns to the state when ADPortScan was started.

If the ADPortScan is not completed within the time period set via PIUSTBLREG's STABLE bits, a PIULostIntr interrupt occurs as a timeout interrupt.

Caution TPPSCAN bit operation is valid during Standby state. The operation is not guaranteed during other states.

Some bits in this register cannot be set in a specific state of scan sequencer. The combination of the setting of this register and the sequencer state is as follows.

Table 20-3. PIUASCNREG Bit Manipulation and States

PIUASCNREG Bit Manipulation		Scan Sequencer's State			
		Disable	Standby	WaitPenTouch	PenData Scan
ADPSSTART	0 → 1	×	ADPortScan[Note]	×	×
	1 → 0	×	Disable	×	×
TPPSCAN	0 → 1	–	–	–	–
	1 → 0	–	–	–	–

PIUCNTREG Bit Manipulation		Scan Sequencer's State		
		IntervalNextScan	ADPortScan	CmdScan
ADPSSTART	0 → 1	×	ADPortScan[Note]	×
	1 → 0	×	Disable	×
TPPSCAN	0 → 1	×	WaitPenTouch	?
	1 → 0	?	?	Standby

Note After ADPortScan is completed, the bit is automatically cleared to 0.

Remarks –: The bit change is retained but there is no state transition.

×: Setting prohibited (operation not guaranteed)

?: Combination of state and bit status before setting does not exist

20.3.7 PIUAMSKREG (0x0B00 0132)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	ADINM3	ADINM2	ADINM1	ADINM0	TPYM1	TPYM0	TPXM1	TPXM0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:8	RFU	Write 0 to these bits. 0 is returned after a read.
7	ADINM3	Audio input port mask Valid only during ADPortScan state. If masked, A/D conversions are not performed for the corresponding port. 1: Mask 0: Normal
6:4	ADINM(2:0)	General-purpose A/D port mask Valid only during ADPortScan state. If masked, A/D conversions are not performed for the corresponding port. 1: Mask 0: Normal
3:2	TPYM(1:0)	Touch panel A/D port TPY mask Valid only during ADPortScan state. If masked, A/D conversions are not performed for the corresponding port. 1: Mask 0: Normal
1:0	TPXM(1:0)	Touch panel A/D port TPX mask Valid only during ADPortScan state. If masked, A/D conversions are not performed for the corresponding port. 1: Mask 0: Normal

This register is used to set masking each A/D port. One bit corresponds to one port. When a port is masked (1), the analog data of that port is not converted into digital data.

The setting of this register is valid only in the ADPortScan state.

20.3.8 PIUCIVLREG (0x0B00 013E)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	CHECK INTVAL10	CHECK INTVAL9	CHECK INTVAL8
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	CHECK INTVAL7	CHECK INTVAL6	CHECK INTVAL5	CHECK INTVAL4	CHECK INTVAL3	CHECK INTVAL2	CHECK INTVAL1	CHECK INTVAL0
R/W	R	R	R	R	R	R	R	R
RTCRST	1	0	1	0	0	1	1	1
After reset	1	0	1	0	0	1	1	1

Bit	Name	Function
15:11	RFU	Write 0 to these bits. 0 is returned after a read.
10:0	CHECKINTVAL(10:0)	Interval count value.

This register is used for real-time reading of internal register values being counted down based on the PIUSIVLREG setting.

20.3.9 PIUPBnmREG (0x0B00 02A0 to 0x0B00 02AE, 0x0B00 02BC to 0x0B00 02BE)

Remark n = 0, 1, m = 0 to 4

PIUPB00REG	(0x0B00 02A0)	PIUPB10REG	(0x0B00 02A8)
PIUPB01REG	(0x0B00 02A2)	PIUPB11REG	(0x0B00 02AA)
PIUPB02REG	(0x0B00 02A4)	PIUPB12REG	(0x0B00 02AC)
PIUPB03REG	(0x0B00 02A6)	PIUPB13REG	(0x0B00 02AE)
PIUPB04REG	(0x0B00 02BC)	PIUPB14REG	(0x0B00 02BE)

Bit	15	14	13	12	11	10	9	8
Name	VALID	RFU	RFU	RFU	RFU	RFU	PADDATA9	PADDATA8
R/W	R/W	R	R	R	R	R	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	PADDATA7	PADDATA6	PADDATA5	PADDATA4	PADDATA3	PADDATA2	PADDATA1	PADDATA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15	VALID	Indicates validity of data in PADDATA 1: Valid 0: Invalid
14:10	RFU	Write 0 to these bits. 0 is returned after a read.
9:0	PADDATA(9:0)	A/D converter's sampling data

These registers are used to store coordinate data or touch pressure data. There are four coordinate data buffers and one touch pressure data buffer, each of which holds two pages of coordinate data or pressure data, and the addresses (register addresses) where the coordinate data or the pressure data is stored are fixed. Read coordinate data from the corresponding register in a valid page.

The VALID bit, which indicates when the data is valid, is automatically rendered invalid when the page buffer interrupt source (PIUPAGE0INTR or PIUPAGE1INTR in PIUINTREG) is cleared.

Table 20-4 shows correspondences between the sampled data and the register in which the sampled data is stored.

Table 20-4. Detected Coordinates and Page Buffers

Detected Data	Page0 Buffer	Page1 Buffer
X–	PIUPB00REG	PIUPB10REG
X+	PIUPB01REG	PIUPB11REG
Y–	PIUPB02REG	PIUPB12REG
Y+	PIUPB03REG	PIUPB13REG
Z (Touch pressure)	PIUPB04REG	PIUPB14REG

20.3.10 PIUABnREG (0x0B00 02B0 to 0x0B00 02B6)

Remark n = 0 to 3

PIUAB0REG	(0x0B00 02B0)
PIUAB1REG	(0x0B00 02B2)
PIUAB2REG	(0x0B00 02B4)
PIUAB3REG	(0x0B00 02B6)

Bit	15	14	13	12	11	10	9	8
Name	VALID	RFU	RFU	RFU	RFU	RFU	PADDATA9	PADDATA8
R/W	R/W	R	R	R	R	R	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	PADDATA7	PADDATA6	PADDATA5	PADDATA4	PADDATA3	PADDATA2	PADDATA1	PADDATA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15	VALID	Indicates validity of data in PADDATA 1: Valid 0: Invalid
14:10	RFU	Write 0 to these bits. 0 is returned after a read.
9:0	PADDATA(9:0)	A/D converter's sampling data

These registers are used to store general-purpose A/D port/audio input port sampling data or command scan data. There are four data buffers and the addresses (register address) where the data is stored are fixed.

The VALID bit, which indicates when the data is valid, is automatically rendered invalid when the page buffer interrupt cause (PIUADPINTR in PIUINTREG) is cleared.

Table 20-5 shows correspondences between the sampled data and the register in which the sampled data is stored.

Table 20-5. A/D Ports and Data Buffers

Register	During ADPortScan		During CmdScan
	TPPSCAN = 0	TPPSCAN = 1	
PIUAB0REG	ADIN0	TPX0	CMDScanDATA
PIUAB1REG	ADIN1	TPX1	–
PIUAB2REG	ADIN2	TPY0	–
PIUAB3REG	AUDIOIN	TPY1	–

20.4 Status Transfer Flow

Be sure to reset the PIU before operating the scan sequencer. Setting initial values via a reset sets particular values for the sequence interval, etc., that are required.

The following registers require initial settings.

PIUSITVLREG	SCANINTVAL(10:0)
PIUSTBLREG	STABLE(3:0)

Interrupt mask cancellation settings are required for registers other than the PIU registers.

Table 20-6. Mask Clear During Scan Sequence Operation

Setting	Unit	Register	Bit	Value
Interrupt mask clear	ICU	MSYSINT1REG	PIUINTR	1
	ICU	MPIUINTREG	bits 6:0	0x7F
Clock mask clear	CMU	CMUCLKMSK	MSKPIU	1

(1) Transfer Flow for Voltage Detection at A/D General-purpose Ports and Audio Input Port

Standby, WaitPenTouch, or Interval state

<1> PIUAMSKREG Mask setting for A/D port and audio input port

<2> PIUASCNREG ADPSSTART = 1

↓

ADPortScan state

<3> PIUASCNREG ADPSSTART = 0

↓

Standby, WaitPenTouch, or IntervalNextScan state

(2) Transfer Flow for Auto Scan Coordinate Detection

Standby state

<1> PIUCNTREG PIUMODE(1:0) = 00
PADATSCAN = 1
PADATSTOP = 1

<2> PIUCNTREG PIUSEQEN = 1

↓

WaitPenTouch state

(3) Transfer Flow for Manual Scan Coordinate Detection

Disable state
<1> PIUCNTREG PIUPWR = 1
↓
Standby state
<2> PIUCNTREG PIUMODE(1:0) = 00
PADSCANSTART = 1
<3> PIUCNTREG PIUSEQEN = 1
↓
PenDataScan state

★ (4) Transfer Flow during Suspend Mode Transition (WaitPenTouch state)

WaitPenTouch state
<1> Only waiting for the time set in PIUSTBLREG's STABL(5:0)
<2> Execution of the SUSPEND instruction (PenChgIntr not occur)

★ (5) Transfer Flow when Returning from Suspend Mode Transition (WaitPenTouch state)

WaitPenTouch state (Register setting and stabilization wait are not needed.)
Touch detected
↓
PenDataScan state

★ (6) Transfer Flow during Suspend Mode Transition (Disable state)

Standby, WaitPenTouch, or IntervalNextScan state
<1> PIUCNTREG PIUSEQEN = 0
↓
Standby state
<2> PIUCNTREG PIUPWR = 1
↓
Disable state
<3> Only waiting for the time set in PIUSTBLREG's STABL(5:0)
<4> Execution of the SUSPEND instruction

★ (7) Transfer Flow when Returning from Suspend Mode (Disable state)

Disable state
<1> PIUCNTREG   PIUPWR = 1
↓
Standby state
<2> PIUCNTREG   PIUMODE(1:0) = 00
PADATSCAN = 1

PADATSTOP = 1
<3> PIUCNTREG   PIUSEQEN = 1
↓
WaitPenTouch state
Touch detected
↓
PenDataScan state

(8) Transfer Flow of Command Scan

Disable state
<1> PIUCNTREG   PIUPWR = 1
↓
Standby state
<2> PIUCNTREG   PIUMODE(1:0) = 01
<3> PIUCNTREG   Touch panel pin setup and input port selection
<4> PIUCNTREG   PIUSEQEN = 1
↓
CmdScan state

20.5 Relationships among TPX, TPY, ADIN, and AUDIOIN Pins and States

State	PadState(2:0)	TPX(1:0)	TPY(1:0)	ADIN(3:0)
PIU disable (pen status detection)	Disable[Note]	HH	D–	––––
Low-power standby	Standby	LL	LL	––––
Pen status detection	WaitPenTouch/Interval	HH	D–	––––
Voltage detection at general-purpose AD0 port	ADPortScan	00	00	–––I
Voltage detection at general-purpose AD1 port	ADPortScan	00	00	––I–
Voltage detection at general-purpose AD2 port	ADPortScan	00	00	–I––
Voltage detection at audio input port	ADPortScan	00	00	I–––
TPY1=H, TPY0=L, TPX0=samp (X+)	PadDataScan	–I	HL	––––
TPY1=L, TPY0=H, TPX0=samp (X–)	PadDataScan	–I	LH	––––
TPX1=H, TPX0=L, TPY0=samp (Y+)	PadDataScan	HL	–I	––––
TPX1=L, TPX0=H, TPY0=samp (Y–)	PadDataScan	LH	–I	––––
Touch pressure detection (Z)	PadDataScan	HH	d–	––––

Note The states of pins are not guaranteed when the PadState(2:0) that precedes the CPU's Suspend or Hibernate instruction execution is in a state other then the Disable state.

Remark

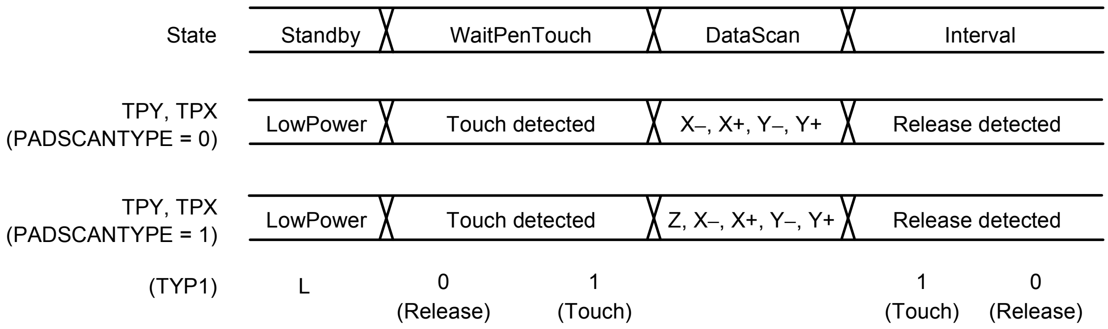
0:	Low-level input
1:	High-level input
L:	Low-level output
H:	High-level output
I:	A/D converter input
D:	Touch interrupt input (with a pull-down resistor)
d:	No touch interrupt input (with a pull-down resistor)
–:	Don't care

20.6 Timing

20.6.1 Touch/release detection timing

Touch/release detection does not use the A/D converter but instead uses the voltage level of the TPY1 pin to determine the panel's touch/release state. The following figure shows a touch/release detection timing diagram.

Figure 20-6. Touch/Release Detection Timing

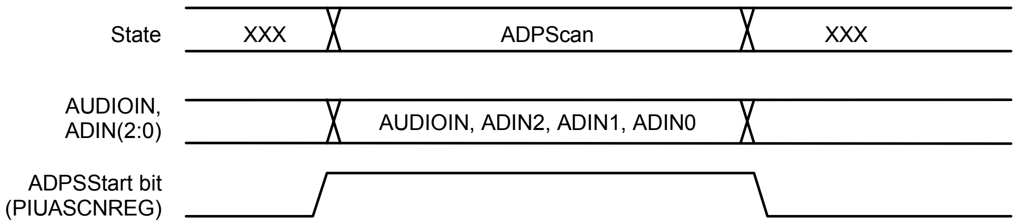


20.6.2 A/D port scan timing

The A/D port scan function sequentially scans the A/D converter's four input channel port pins and stores the data in the data buffer used for A/D port scanning.

The following figure shows an A/D port scan timing diagram.

Figure 20-7. A/D Port Scan Timing



XXX state: Standby or WaitPenTouch or Interval

20.7 Data Lost Generation Conditions

The PIU issues a PIUDataLostIntr interrupt when any of the following four conditions exist.
Once a PIUDataLostIntr interrupt occurs, the sequencer is forcibly changed to the Standby state.

1. Data for one coordinate has not been obtained within the interval period
2. The A/D port scan has not been completed within the time set via PIUSTBLREG
3. Transfer of the next coordinate data has begun while valid data for both pages remains in the buffer
4. The next data transfer starts while there is valid data in the ADPortScan buffer

(1) When data for one coordinate has not been obtained within the interval period

Cause

This condition occurs when the AIU has exclusive use of the A/D converter and the PIU is therefore unable to use the A/D converter.
If this data loss condition occurs frequently, implement a countermeasure that temporarily prohibits the AIU's use of the A/D converter.

Response

After clearing the cause of the PIUDataLostIntr interrupt, set PIUCIUCNTREG's PADATSTART bit or PADSCANSTART bit to restart the coordinate detection operation. Once the PIUDataLostIntr interrupt is cleared, the page in which the loss occurred becomes invalid. If the valid data prior to the data loss is needed, be sure to save the data that is being stored in the page buffer before clearing the PIUDataLostIntr interrupt.

(2) When the A/D port scan has not been completed within the time set via PIUSTBLREG

Cause

Same as cause of condition 1

Response

After clearing the cause of the PIUDataLostIntr interrupt, set PIUASCNREG's ADPSSTART bit to restart the A/D port scan operation. Once the PIUDataLostIntr interrupt is cleared, the page in which the loss occurred becomes invalid. If the valid data prior to the data loss is needed, be sure to save the data that is being stored in the page buffer before clearing the PIUDataLostIntr interrupt.

(3) When transfer of the next coordinate data has begun while valid data for both pages remains in the buffer

Cause

This condition is caused when the data buffer contains two pages of valid data (both the PIUPAGE1INTR and PIUPAGE0INTR interrupts have occurred) but the valid data has not been processed. If the A/D converter is used frequently, this may shorten the time that would normally be required from when both pages become full until when the data loss occurs.

Response

In condition 3, valid data contained in the pages when the PIUDataLostIntr interrupt occurs is never overwritten.

After two pages of valid data are processed, clear the causes of the three interrupts (PIUDataLostIntr, PIUPAGE1INTR, and PIUPAGE0INTR).

After clearing these interrupt causes, set the PADATSTART bit or PADSCANSTART bit of PIUCIUCNTREG to restart the coordinate detection operation.

(4) When the next data transfer starts while there is valid data in the ADPortScan buffer

Cause

This condition is caused when valid data is not processed even while the ADPortScan buffer holds valid data (PADADPINTR interrupt occurrence).

Response

In condition 4, valid data contained in the buffer when the PIUDataLostIntr interrupt occurs is never overwritten.

After valid data in the buffer is processed, clear the causes of the two interrupts (PIUDataLostIntr, PADADPINTR).

After clearing these interrupt causes, set the ADPSSTART bit of PIUASCNREG to restart the general-purpose A/D port scan.

[MEMO]

CHAPTER 21 AIU (AUDIO INTERFACE UNIT)

21.1 General

The AIU supports speaker output and MIC input. The resolution of the D/A converter used for a speaker and the A/D converter used for a microphone is usually 10 bits.

21.2 Register Set

The AIU registers are listed below.

Table 21-1. AIU Registers

Address	R/W	Register Symbols	Function
0x0B00 0160	R/W	MDMADATREG	MIC DMA Data Register
0x0B00 0162	R/W	SDMADATREG	Speaker DMA Data Register
0x0B00 0166	R/W	SODATREG	Speaker Output Data Register
0x0B00 0168	R/W	SCNTREG	Speaker Output Control Register
0x0B00 016A	R/W	SCNVRREG	Speaker Conversion Rate Register
0x0B00 0170	R/W	MIDATREG	MIC Input Data Register
0x0B00 0172	R/W	MCNTREG	MIC Input Control Register
0x0B00 0174	R/W	MCNVRREG	MIC Conversion Rate Register
0x0B00 0178	R/W	DVALIDREG	Data Valid Register
0x0B00 017A	R/W	SEQREG	Sequential Register
0x0B00 017C	R/W	INTREG	Interrupt Register

These registers are described in detail below.

21.2.1 MDMADATREG (0x0B00 0160)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	MDMA9	MDMA8
R/W	R	R	R	R	R	R	R/W	R/W
RTCRST	0	0	0	0	0	0	1	0
After reset	0	0	0	0	0	0	1	0

Bit	7	6	5	4	3	2	1	0
Name	MDMA7	MDMA6	MDMA5	MDMA4	MDMA3	MDMA2	MDMA1	MDMA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:10	RFU	Write 0 to these bits. 0 is returned after a read.
9:0	MDMA(9:0)	MIC input DMA data

This register is used prior to DMA transfer to store 10-bit data that has been converted by the A/D converter and stored in MIDATREG. Write is used for debugging and is enabled when AIUMEN bit of SEQREG is set to 1. This register is cleared (0x0200) by resetting AIUMEN bit of SEQREG to 0. Therefore, if the AIUMEN bit is set to 0 during DMA transfer, invalid data may be transferred.

21.2.2 SDMADATREG (0x0B00 0162)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	SDMA9	SDMA8
R/W	R	R	R	R	R	R	R/W	R/W
RTCRST	0	0	0	0	0	0	1	0
After reset	0	0	0	0	0	0	1	0

Bit	7	6	5	4	3	2	1	0
Name	SDMA7	SDMA6	SDMA5	SDMA4	SDMA3	SDMA2	SDMA1	SDMA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:10	RFU	Write 0 to these bits. 0 is returned after a read.
9:0	SDMA(9:0)	Speaker output DMA data

This register is used to store 10-bit DMA data for speaker output. When SODATREG is empty, the data is transferred to SODATREG. Write is used for debugging and is enabled when AIUSEN bit of SEQREG is set to 1. This register is cleared (0x0200) by resetting AIUSEN bit of SEQREG to 0.

21.2.3 SODATREG (0x0B00 0166)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	SODAT9	SODAT8
R/W	R	R	R	R	R	R	R/W	R/W
RTCRST	0	0	0	0	0	0	1	0
After reset	0	0	0	0	0	0	1	0

Bit	7	6	5	4	3	2	1	0
Name	SODAT7	SODAT6	SODAT5	SODAT4	SODAT3	SODAT2	SODAT1	SODAT0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:10	RFU	Write 0 to these bits. 0 is returned after a read.
9:0	SODAT(9:0)	Speaker output data

This register is used to store 10-bit DMA data for speaker output. Data is received from the D/A converter and is sent to SDMADATREG. Write is used for debugging and is enabled when AIUSEN bit of SEQREG is set to 1. This register is cleared (0x0200) by resetting AIUSEN bit of SEQREG to 0.

21.2.4 SCNTREG (0x0B00 0168)

Bit	15	14	13	12	11	10	9	8
Name	DAENAIU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R/W	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	SSTATE	RFU	SSTOPEN	RFU
R/W	R	R	R	R	R	R	R/W	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15	DAENAIU	This is the speaker D/A enable bit. 1: Vref ON 0: Vref OFF
14:4	RFU	Write 0 to these bits. 0 is returned after a read.
3	SSTATE	Indicates speaker operation state 1: In operation 0: Stopped
2	RFU	Write 0 to this bit. 0 is returned after a read.
1	SSTOPEN	Speaker output DMA transfer page boundary interrupt stop 1: Stop DMA request at 1-page boundary 0: Stop DMA request at 2-page boundary
0	RFU	Write 0 to this bit. 0 is returned after a read.

This register is used to control the AIU's speaker block.

The DAENAIU bit controls the connection of DV_{DD} and Vref input to ladder type resistors in the D/A converter. Setting this bit to 0 (OFF) allows low power consumption when not using the D/A converter. When using the D/A converter, this bit must be set following the sequence described in **21.3 Operation Sequence**.

The content of the SSTATE bit is valid only when the AIUSEN bit of SEQREG is set to 1.

21.2.5 SCNVRREG (0x0B00 016A)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	SCNVR2	SCNVR1	SCNVR0
R/W	R	R	R	R	R	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:3	RFU	Write 0 to these bits. 0 is returned after a read.
2:0	SCNVR(2:0)	D/A Conversion Rate 111: RFU : 101: RFU 100: 8 ksps 011: RFU 010: 44.1 ksps 001: 22.05 ksps 000: 11.025 ksps

This register is used to select a conversion rate for the D/A converter.

21.2.6 MIDATREG (0x0B00 0170)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	MIDAT9	MIDAT8
R/W	R	R	R	R	R	R	R/W	R/W
RTCRST	0	0	0	0	0	0	1	0
After reset	0	0	0	0	0	0	1	0

Bit	7	6	5	4	3	2	1	0
Name	MIDAT7	MIDAT6	MIDAT5	MIDAT4	MIDAT3	MIDAT2	MIDAT1	MIDAT0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:10	RFU	Write 0 to these bits. 0 is returned after a read.
9:0	MIDAT(9:0)	MIC input data

This register is used to store 10-bit speaker input data that has been converted by the A/D converter. Data is sent to MDMADATREG and is received from the A/D converter. Write is used for debugging and is enabled when AIUMEN bit of SEQREG is set to 1. This register is cleared (0x0200) by resetting AIUMEN bit of SEQREG to 0.

21.2.7 MCNTREG (0x0B00 0172)

Bit	15	14	13	12	11	10	9	8
Name	ADENAIU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R/W	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	MSTATE	RFU	MSTOPEN	ADREQAIU
R/W	R	R	R	R	R	R	R/W	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15	ADENAIU	This is the MIC A/D enable bit. 1: Vref ON 0: Vref OFF
14:4	RFU	Write 0 to these bits. 0 is returned after a read.
3	MSTATE	Indicates MIC operation state 1: In operation 0: Stopped
2	RFU	Write 0 to this bit. 0 is returned after a read.
1	MSTOPEN	MIC input DMA transfer page boundary interrupt stop 1: Stop DMA request at 1-page boundary 0: Stop DMA request at 2-page boundary
0	ADREQAIU	A/D use request bit 1: Request 0: Normal

This register is used to control the AIU's MIC block.

The ADENAIU bit controls the connection of AV_{DD} and Vref input to ladder type resistors in the A/D converter. Setting this bit to 0 (OFF) allows low power consumption when not using the A/D converter. When using the A/D converter, this bit must be set following the sequence described in **21.3 Operation Sequence**.

The content of the MSTATE bit is valid only when the AIUMEN bit of SEQREG is set to 1.

This unit has priority when a conflict occurs with the PIU in relation to A/D conversion requests.

21.2.8 MCNVRREG (0x0B00 0174)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	MCNVR2	MCNVR1	MCNVR0
R/W	R	R	R	R	R	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:3	RFU	Write 0 to these bits. 0 is returned after a read.
2:0	MCNVR(2:0)	A/D Conversion Rate 111: RFU : 101: RFU 100: 8 ksps 011: RFU 010: 44.1 ksps 001: 22.05 ksps 000: 11.025 ksps

This register is used to select a conversion rate for the A/D converter.

21.2.9 DVALIDREG (0x0B00 0178)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	SODATV	SDMAV	MIDATV	MDMAV
R/W	R	R	R	R	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:4	RFU	Write 0 to these bits. 0 is returned after a read
3	SODATV	This indicates when valid data has been stored in SODATREG. 1: Valid data exists 0: No valid data
2	SDMAV	This indicates when valid data has been stored in SDMADATREG. 1: Valid data exists 0: No valid data
1	MIDATV	This indicates when valid data has been stored in MIDATREG. 1: Valid data exists 0: No valid data
0	MDMAV	This indicates when valid data has been stored in MDMADATREG. 1: Valid data exists 0: No valid data

This register indicates when valid data has been stored in SODATREG, SDMADATREG, MIDATREG, or MDMADATREG.

If data has been written directly to SODATREG, SDMADATREG, MIDATREG, or MDMADATREG via software, the bits in this register are not active, so write "1" via software.

Write is used for debugging and is enabled when AIUSEN or AIUMEN bit of SEQREG is set to 1.

If AIUSEN = 0 or AIUMEN = 0 in SEQREG, then SODATV = SDMAV = 0 or MIDATV = MDMAV = 0.

21.2.10 SEQREG (0x0B00 017A)

Bit	15	14	13	12	11	10	9	8
Name	AIURST	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R/W	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	AIUMEN	RFU	RFU	RFU	AIUSEN
R/W	R	R	R	R/W	R	R	R	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15	AIURST	AIU reset via software 1: Reset 0: Normal
14:5	RFU	Write 0 to these bits. 0 is returned after a read.
4	AIUMEN	MIC block operation enable, DMA enable 1: Enable operation 0: Disable operation
3:1	RFU	Write 0 to these bits. 0 is returned after a read.
0	AIUSEN	Speaker block operation enable, DMA enable 1: Enable operation 0: Disable operation

This register is used to enable/disable the AIU's operation.

21.2.11 INTREG (0x0B00 017C)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	MENDINTR	MINTR	MIDLEINTR	MSTINTR
R/W	R	R	R	R	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	SENDINTR	SINTR	SIDLEINTR	RFU
R/W	R	R	R	R	R/W	R/W	R/W	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:12	RFU	Write 0 to these bits. 0 is returned after a read.
11	MENDINTR	MIC DMA 2 page interrupt. Cleared to 0 when 1 is written. 1: Occurred 0: Normal
10	MINTR	MIC DMA 1 page interrupt. Cleared to 0 when 1 is written. 1: Occurred 0: Normal
9	MIDLEINTR	MIC idle interrupt (receive data loss). Cleared to 0 when 1 is written. 1: Occurred 0: Normal
8	MSTINTR	MIC receive complete interrupt. Cleared to 0 when 1 is written. 1: Occurred 0: Normal
7:4	RFU	Write 0 to these bits. 0 is returned after a read.
3	SENDINTR	SPEAKER DMA 2 page interrupt. Cleared to 0 when 1 is written. 1: Occurred 0: Normal
2	SINTR	SPEAKER DMA 1 page interrupt. Cleared to 0 when 1 is written. 1: Occurred 0: Normal
1	SIDLEINTR	SPEAKER idle interrupt (mute). Cleared to 0 when 1 is written. 1: Occurred 0: Normal
0	RFU	Write 0 to this bit. 0 is returned after a read.

This register is used to set/indicate whether AIU interrupts have occurred or not.

When data is received from the A/D converter, MIDLEINTR is set if valid data still exists in MIDATREG (MIDATV = 1). In this case, MIDATREG is overwritten.

MSTINTR is set when data is received in MDMADATREG.

When data is passed to the D/A converter, SIDLEINTR is set if there is no valid data in SODATREG (SODATV = 0). However, this interrupt is valid only after AIUSEN = 1, after which SODATV = 1 in DVALIDREG.

21.3 Operation Sequence

21.3.1 Output (Speaker)

(1) When using DMA transfer

1. Set conversion rate (0x0B00 016A: SCNVR = any value)
2. Set output data area to DMAAU
3. DMA enable in DCU
4. Set D/A converter's Vref to ON (0x0B00 0168: DAENAIU = 1)
5. Wait for Vref resistor stabilization time (about 5 μs) (use the RTC counter)

   Even if speaker power is set to ON and speaker operation is enabled (AIUSEN = 1) without waiting for Vref resistor stabilization time, speaker output starts after the period calculated with the formula below.

   5 + 1/conversion rate (44.1, 22.05, 11.025, or 8 ksps) (μs)

   In this case, however, a noise may occur when speaker power is set to ON.
6. Set speaker power ON via GPIO.
7. Speaker operation enable (0x0B00 017A: AIUSEN = 1)

   DMA request

   Receive acknowledge and DMA data from DMA

   0x0B00 0178: SDMAV = SODATV = 1

   Output 10-bit data (0x0B00 0166: SODAT) to D/A converter

   SODATV = 0, SDMAV = 1

   Send SDMADATREG data to SODATREG.

   SODATV = 1, SDMAV = 0

   Output DMA request and store the data after the next into SDMADATREG.

   SODATV = 1, SDMAV = 1

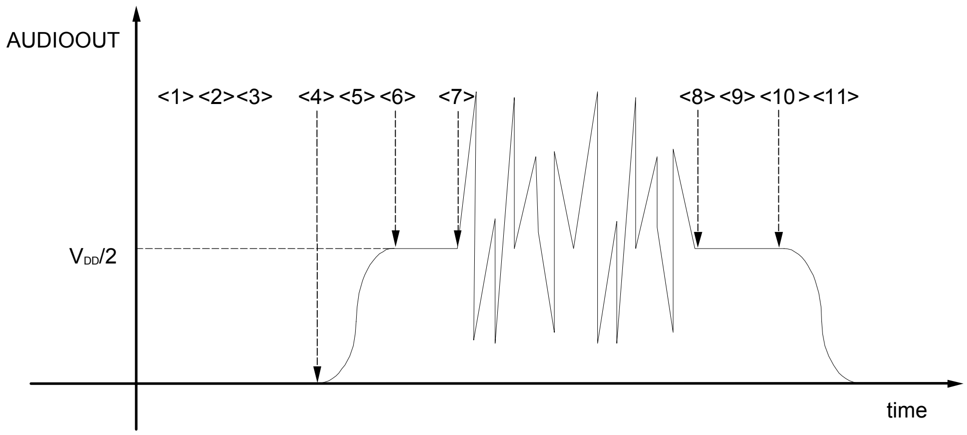
   Refresh data at each conversion timing interval

   Becomes SIDLEINTR = 1 when DMA is slow and SODATV = 0 during conversion timing interval, and (mute) interrupt occurs

   DMA page boundary interrupt occurs at page boundary

   Clear the page interrupt request to continue output.
8. Speaker operation to disable (0x0B00 017A: AIUSEN = 0)
9. Set speaker power OFF via GPIO.
10. Set D/A converter's Vref to OFF (0x0B00 0168: DAENAIU = 0)
11. DMA disable in DCU

Figure 21-1. Speaker Output and AUDIOOUT Pin



(2) When not using DMA transfer

1. Enable clock supply to AIU in CMU
2. Set D/A conversion rate (0x0B00 016A: SCNVR = any value)
3. DMA disable in DCU (disable set as initial value)
4. Set D/A converter's Vref to ON (0x0B00 0168: DAENAIU = 1)
5. Wait for Vref resistor stabilization time (about 5 μs)
6. Set speaker power ON via GPIO
7. Speaker operation enable (0x0B00 017A: AIUSEN = 1)
   Sampling counter begins to count up
8. Speaker operation disable (0x0B00 017A: AIUSEN = 0)
9. Set speaker power OFF via GPIO
10. Set D/A converter's Vref to OFF (0x0B00 0168: DAENAIU = 0)

Remark The interrupt request caused by mute is valid after setting AIUSEN = 1 and then SODATV = 1. However, SODATV does not become 1 until DVALIDREG is written by the DMA or software. Therefore mute interrupt requests will not occur as long as DMA is disabled and SODATV is not converted by software.

21.3.2 Input (MIC)

1. Set conversion rate (0x0B00 0174: MCNVR = any value)
2. Set input data area in DMAAU
3. DMA enable in DCU
4. Set A/D converter's Vref to ON (0x0B00 0172: ADENAIU = 1)

   MIC power can be set ON and MIC operation can be enabled without waiting for Vref resistor stabilization time (about 5 μs). However, in such a case, sampling starts after the period calculated with the formula below.

   5 + 1/conversion rate (44.1, 22.05, 11.025, or 8 ksps) (μs)
5. Set MIC power ON via GPIO.
6. MIC operation enable (0x0B00 017A: AIUMEN = 1)

   Output A/D conversion request (ADREQAIU) to A/D converter

   Return acknowledge (aiuadack) and 10-bit conversion data from A/D converter.

   Store data in MIDATREG.

   0x0B00 0178: MDMAV = 0, MIDATV = 1

   Transfer data from MIDATREG to MDMADATREG.

   MDMAV = 1, MIDATV = 0

   The INTMST value becomes "1" and an interrupt (receive complete) occurs.

   Issue DMA request and store MIDMADATREG data to memory.

   MDMAV = 0, MIDATV = 0

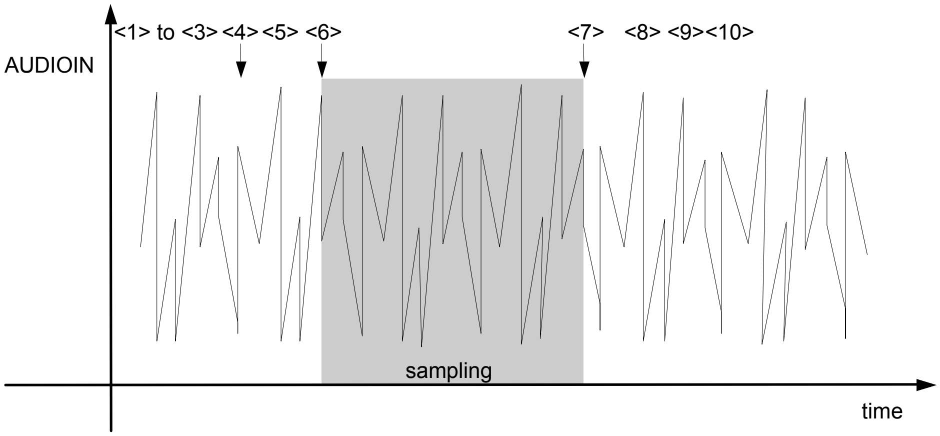
   An A/D request is issued once per conversion timing interval and 10-bit data is received

   Becomes MIDLEINTR = 1 when DMA is slow and MIDATV = 1 during conversion timing interval, and (data loss) interrupt occurs

   DMA page boundary interrupt occurs at page boundary

   Clear the page interrupt request to continue output.
7. MIC operation to disable (0x0B00 017A: AIUMEN = 0)
8. Set MIC power OFF via GPIO.
9. Set A/D converter's Vref to OFF (0x0B00 0172: ADENAIU = 0)
10. DMA disable in DCU

Figure 21-2. AUDIOIN Pin and MIC Operation



[MEMO]

CHAPTER 22 KIU (KEYBOARD INTERFACE UNIT)

22.1 General

The KIU includes 12 scan lines and 8 detection lines. The number of key inputs to be detected can be selected from 96/80/64, by switching the number of scan lines from 12/10/8.

The register can be set to enable the 12 scan lines to be used as a general-purpose output port.

22.2 Register Set

The KIU registers are listed below.

Table 22-1. KIU Registers

Address	R/W	Register Symbols	Function
0x0B00 0180	R/W	KIUDAT0	KIU Data0 Register
0x0B00 0182	R/W	KIUDAT1	KIU Data1 Register
0x0B00 0184	R/W	KIUDAT2	KIU Data2 Register
0x0B00 0186	R/W	KIUDAT3	KIU Data3 Register
0x0B00 0188	R/W	KIUDAT4	KIU Data4 Register
0x0B00 018A	R/W	KIUDAT5	KIU Data5 Register
0x0B00 0190	R/W	KIUSCANREP	KIU Scan/Repeat Register
0x0B00 0192	R	KIUSCANS	KIU Scan Status Register
0x0B00 0194	R/W	KIUWKS	KIU Wait Keyscan Stable Register
0x0B00 0196	R/W	KIUWKI	KIU Wait Keyscan Interval Register
0x0B00 0198	R/W	KIUINT	KIU Interrupt Register
0x0B00 019A	W	KIURST	KIU Reset Register
0x0B00 019C	R/W	KIUGPEN	KIU General Purpose Output Enable
0x0B00 019E	R/W	SCANLINE	KIU Scan Line Register

22.2.1 KIUDATn (0x0B00 0180 to 0x0B00 018A)

Remark n = 0 to 5
KIUDAT0 (0x0B00 0180)
KIUDAT1 (0x0B00 0182)
KIUDAT2 (0x0B00 0184)
KIUDAT3 (0x0B00 0186)
KIUDAT4 (0x0B00 0188)
KIUDAT5 (0x0B00 018A)

Bit	15	14	13	12	11	10	9	8
Name	KEYDAT15	KEYDAT14	KEYDAT13	KEYDAT12	KEYDAT11	KEYDAT10	KEYDAT9	KEYDAT8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	KEYDAT7	KEYDAT6	KEYDAT5	KEYDAT4	KEYDAT3	KEYDAT2	KEYDAT1	KEYDAT0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:8	KEYDAT(15:8)	Scan data from odd-numbered scans
7:0	KEYDAT(7:0)	Scan data from even-numbered scans

These registers are used to hold key scan data.

Each KIU data register is able to hold the data from one scan operation.

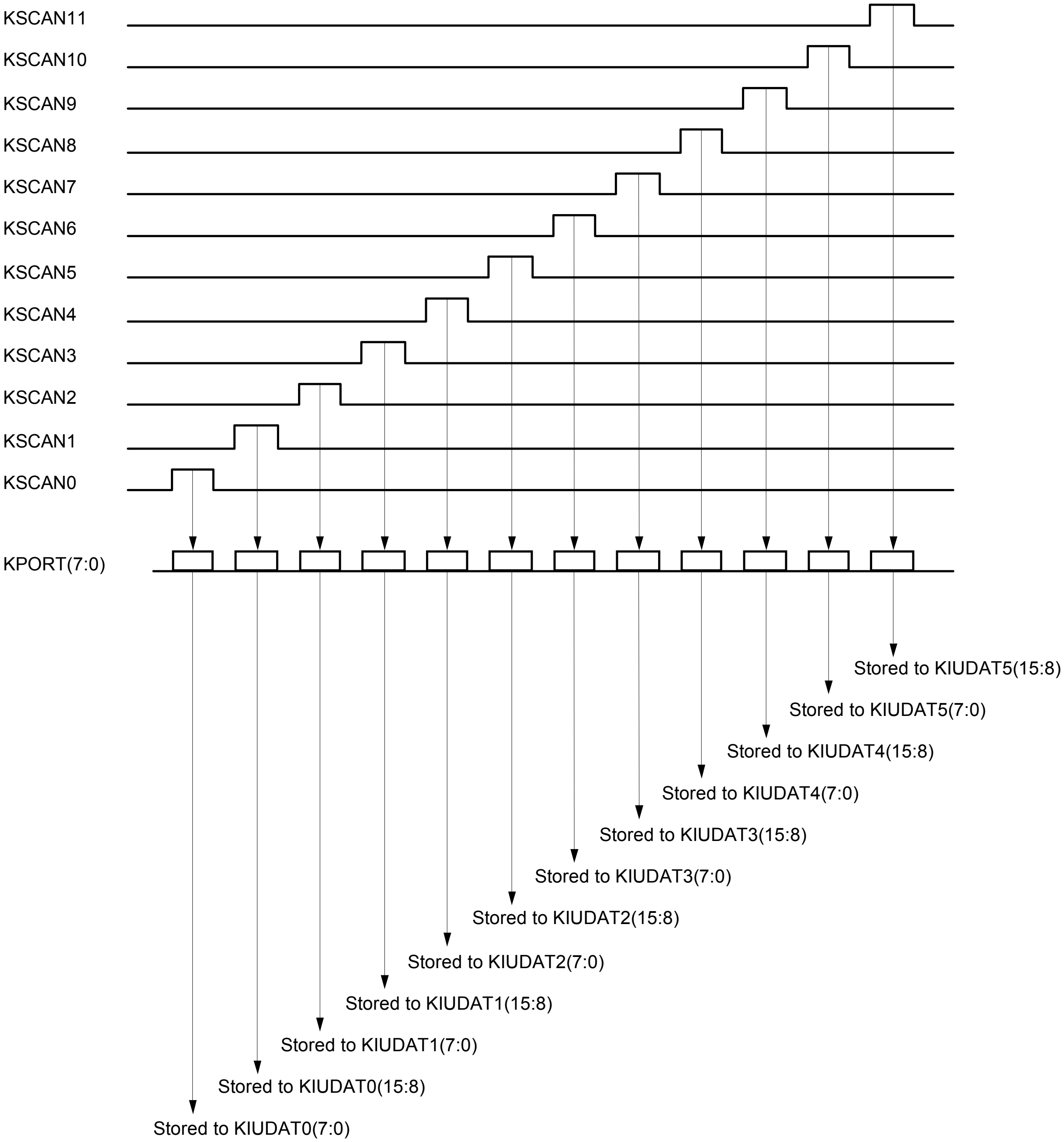
How scan data is input to the registers is as below. Figure 22-1 shows a scan operation and storing timing.

Register	Bits	Data
KIUDAT0	KEYDAT(7:0)	Stores the data scanned by the KSCAN0 pin.
	KEYDAT(15:8)	Stores the data scanned by the KSCAN1 pin.
KIUDAT1	KEYDAT(7:0)	Stores the data scanned by the KSCAN2 pin.
	KEYDAT(15:8)	Stores the data scanned by the KSCAN3 pin.
KIUDAT2	KEYDAT(7:0)	Stores the data scanned by the KSCAN4 pin.
	KEYDAT(15:8)	Stores the data scanned by the KSCAN5 pin.
KIUDAT3	KEYDAT(7:0)	Stores the data scanned by the KSCAN6 pin.
	KEYDAT(15:8)	Stores the data scanned by the KSCAN7 pin.
KIUDAT4	KEYDAT(7:0)	Stores the data scanned by the KSCAN8 pin.
	KEYDAT(15:8)	Stores the data scanned by the KSCAN9 pin.
KIUDAT5	KEYDAT(7:0)	Stores the data scanned by the KSCAN10 pin.
	KEYDAT(15:8)	Stores the data scanned by the KSCAN11 pin.

The data in the KIUDAT00 to KIUDAT05 registers should be read out in the interval time between two key scan operations. Scan interval is set by the KIUWKI register.

When data is not read before the next key scan operation starts, the key scan data lost interrupt occurs (see **22.2.6 KIUINT**). The data registers KIUDAT00 through KIUDAT05 overwrite the following scan data.

Figure 22-1. Scan Operation and Key Data Store Register



22.2.2 KIUSCANREP (0x0B00 0190)

Bit	15	14	13	12	11	10	9	8
Name	KEYEN	RFU	RFU	RFU	RFU	RFU	STPREP5	STPREP4
R/W	R/W	R	R	R	R	R	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	STPREP3	STPREP2	STPREP1	STPREP0	SCANSTP	SCANSTART	ATSTP	ATSCAN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	1
After reset	0	0	0	0	0	0	0	1

Bit	Name	Function
15	KEYEN	Key scan enable 1: Enable 0: Prohibit
14:10	RFU	Write 0 to these bits. 0 is returned after a read.
9:4	STPREP(5:0)	Key scan sequencer stop count setting 111111: 63 times : 000001: 1 time 000000: RFU
3	SCANSTP	Key scan stop 1: Stop 0: Operate
2	SCANSTART	Key scan start When this bit is set to 1, the key scan operation starts immediately. 1: Start 0: Stop
1	ATSTP	Key auto stop setting 1: Auto stop 0: Not auto stop
0	ATSCAN	Key auto scan setting 1: Auto scan 0: Not auto scan

This register is used to enable operation of the key scan unit and to make settings for key scan and the key scan sequencer.

When the number of scan lines is set to 0 in the SCANLINE register, the KEYEN bit cannot be set to "1". Each mode is described in detail below.

- Key scan stop
  The SCANSTP bit should be set to 1 when the KIUSCANS register stops the key scan operation in Scanning or Interval Next Scan mode.
  When this bit is set to "1", the key scan sequencer stops. However, if this bit is set to "1" during a key scan operation, the key scan sequencer stops after the current set of key data is received.
  This bit becomes 0 when the key scan sequencer stops.
  When the key scan operation is started by setting this bit to 1 during Stopped or WaitKeyIn state, the key scan operation stops immediately after a set of key scan operation is completed.

- Key scan start
  When the SCANSTART bit is set to "1", the key scan sequencer starts regardless of key contact detection.
  This bit becomes 0 when the key scan operation starts.
  This bit cannot be set while the KEYEN bit is 0.

- Key scan auto stop setting
  In the key scan auto stop mode, the key scan operation stops automatically when the data of all zeros is input to the KPORT(7:0) pins (no key contact is detected).
  The number of zeros is set by the STPREP(5:0) bits.

- Key auto scan setting
  When the ATSCAN bit is set to 1, the key touch wait state is entered, and key scan operation starts automatically upon a key touch ("1" is input to any of the KPORT(7:0) pins).
  When the KEYEN bit is 0, the key touch wait state is not entered even if this bit is set to 1. The key wait state is entered and the key auto scan mode is set from the point when the KEYEN bit is set to 1.

For details, see **Figure 22-5 Transition of Sequencer Status** and **Figure 22-6 Basic Operation Timing Chart**.

22.2.3 KIUSCANS (0x0B00 0192)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	SSTAT1	SSTAT0
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:2	RFU	Write 0 to these bits. 0 is returned after a read.
1:0	SSTAT(1:0)	KIU sequencer status 11: Scanning 10: Interval Next Scan 01: WaitKeyIn 00: Stopped

This register indicates the current KIU sequencer status.

Details of the status of the KIU sequencer are described below.

- Scanning: This is the state where the scan sequencer performs key scan to load key data.

- Interval Next Scan: This is the state where the scan of a set of key data has completed and the start of the next key scan is being waited for.

Note The number of bits differs according to the number of KSCAN pins used. The number of KSCAN pins is set in the SCANLINE register.

KSCAN Pins	Data Bits
8	64 bits
10	80 bits
12	96 bits

- Wait Key In: This is the state of waiting for key input in the key auto scan mode. When the ATSCAN bit of the KIUSCANREP register is set to 1 and the scan sequencer is enabled, key input is waited for. In this state, all the KSCAN pin[Note] outputs are high level. Prior to shifting the CPU into Suspend mode (or Standby mode with the TClock masked), KIU must always be set in auto scan mode and whether the state of the sequencer is Wait key in must be confirmed.

Note The setting of the SCANLINE register's LINE(1:0) bit determines the number of KSCAN pins used, as follows.

LINE(1:0)	KSCAN Pins
10	8
01	10
00	12

- Stopped: This is the state where the sequencer is disabled.

22.2.4 KIUWKS (0x0B00 0194)

Bit	15	14	13	12	11	10	9	8
Name	RFU	T3CNT4	T3CNT3	T3CNT2	T3CNT1	T3CNT0	T2CNT4	T2CNT3
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	1	1	1	1	1	1	1
After reset	0	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
Name	T2CNT2	T2CNT1	T2CNT0	T1CNT4	T1CNT3	T1CNT2	T1CNT1	T1CNT0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	1	1	1	1	1	1	1	1
After reset	1	1	1	1	1	1	1	1

Bit	Name	Function
15	RFU	Write 0 to this bit. 0 is returned after a read.
14:10	T3CNT(4:0)	Wait time setting ((T3CNT(4:0) + 1) × 30 μs) 11111: 960 μs : 00001: 60 μs 00000: RFU
9:5	T2CNT(4:0)	Off time setting ((T2CNT(4:0) + 1) × 30 μs) 11111: 960 μs : 00001: 60 μs 00000: RFU
4:0	T1CNT(4:0)	Stabilization time setting ((T1CNT(4:0) + 1) × 30 μs) 11111: 960 μs : 00001: 60 μs 00000: RFU

This register is used to set the wait time between when the key scan sequencer sets the KSCAN pin active during a key matrix scan and when the status is read from the KPORT pin.

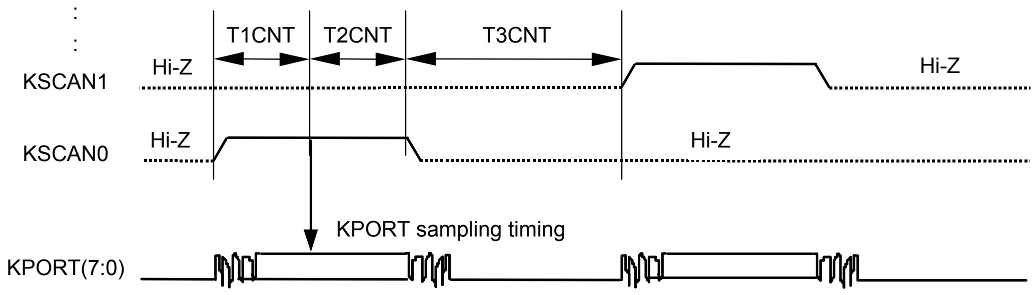
The T1CNT bit is used to set the stabilization time between when the KSCAN pin becomes high and when the key scan data is read.

The T2CNT bit is used to set the time between when the key data is read and when the KSCAN pin becomes low.

The T3CNT bit is used to set the time between when the KSCAN pin becomes low and when it becomes high again.

The status of output from the KSCAN pins and the timing of KPORT sampling are shown below.

Figure 22-2. KSCAN Pin Status and KPORT Sampling Timing



22.2.5 KIUWKI (0x0B00 0196)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	WINTVL9	WINTVL8
R/W	R	R	R	R	R	R	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

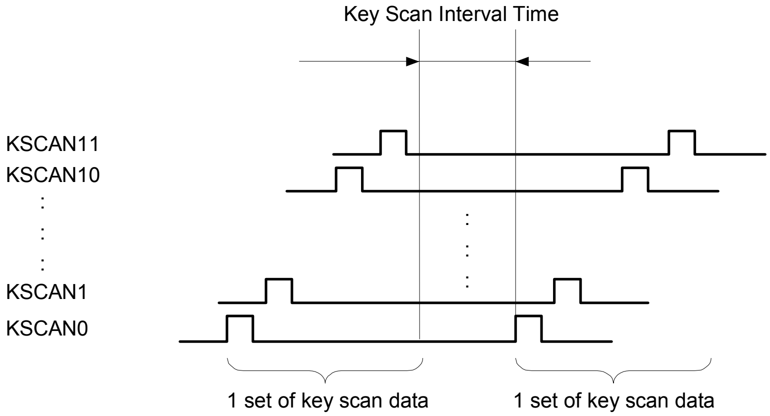
Bit	7	6	5	4	3	2	1	0
Name	WINTVL7	WINTVL6	WINTVL5	WINTVL4	WINTVL3	WINTVL2	WINTVL1	WINTVL0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:10	RFU	Write 0 to these bits. 0 is returned after a read.
9:0	WINTVL(9:0)	Key scan interval time setting (WINTVL(9:0) × 30 μs) 1111111111: 30,690 μs : 0000000001: 30 μs 0000000000: No Wait

This register is used to set the interval time between when one set of key data is obtained by the key scan sequencer and when the next set of key data is obtained.

The following figure shows the key scan interval time.

Figure 22-3. Key Scan Interval



22.2.6 KIUINT (0x0B00 0198)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	KDATLOST	KDATRDY	SCANINT
R/W	R	R	R	R	R	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:3	RFU	Write 0 to these bits. 0 is returned after a read.
2	KDATLOST	Key scan data lost interrupt. Cleared to 0 when 1 is written. 1: Yes 0: No
1	KDATRDY	Key data scan complete interrupt. Cleared to 0 when 1 is written. 1: Yes 0: No
0	SCANINT	Key input detection interrupt. Cleared to 0 when 1 is written. 1: Yes 0: No

This register indicates the type of interrupt that has occurred in the KIU.

The key scan data lost interrupt occurs when data is not read out from the KIU data register (KIUDAT0 through KIUDAT5) between when data is input to the data register after a key scan and when the next scan operation starts. The contents of the data registers are overwritten to the new key scan data.

Key data scan complete interrupt occurs when all the key data is input after one scan operation is completed.

Key input detection interrupt occurs in the auto key scan mode when a key touch is detected (1 is detected from any of the KPORT(7:0) pins) in the key touch wait state, when a key scan operation starts after setting the start of key scan, or when a key scan operation starts after returning from the Suspend mode upon key touch detection.

22.2.7 KIURST (0x0B00 019A)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	KIURST
R/W	R	R	R	R	R	R	R	W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:1	RFU	Write 0 to these bits. 0 is returned after a read.
0	KIURST	KIU reset. Cleared to 0 when 1 is written. 1: Reset 0: Normal operation

This register is used to forcibly reset the KIU registers, except for the KIUGPEN register.

22.2.8 KIUGPEN (0x0B00 019C)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	KGPEN11	KGPEN10	KGPEN9	KGPEN8
R/W	R	R	R	R	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	**Note**	**Note**	**Note**	**Note**

Bit	7	6	5	4	3	2	1	0
Name	KGPEN7	KGPEN6	KGPEN5	KGPEN4	KGPEN3	KGPEN2	KGPEN1	KGPEN0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15:12	RFU	Write 0 to these bits. 0 is returned after a read.
11:0	KGPEN(11:0)	KSCAN pin function 1: Use as output port 0: Use as KSCAN pin

Note The value before reset is retained.

This register is used to set whether or not the KSCAN(11:0) pins will function as a general-purpose output port. The KGPEN(11:0) bits correspond to the KSCAN(11:0) pins bitwise.

The output port setting are made via the GIU's GIUPODATL register (0x0B00 011C).

The correspondences between KSCAN and KGPEN pins are as below.

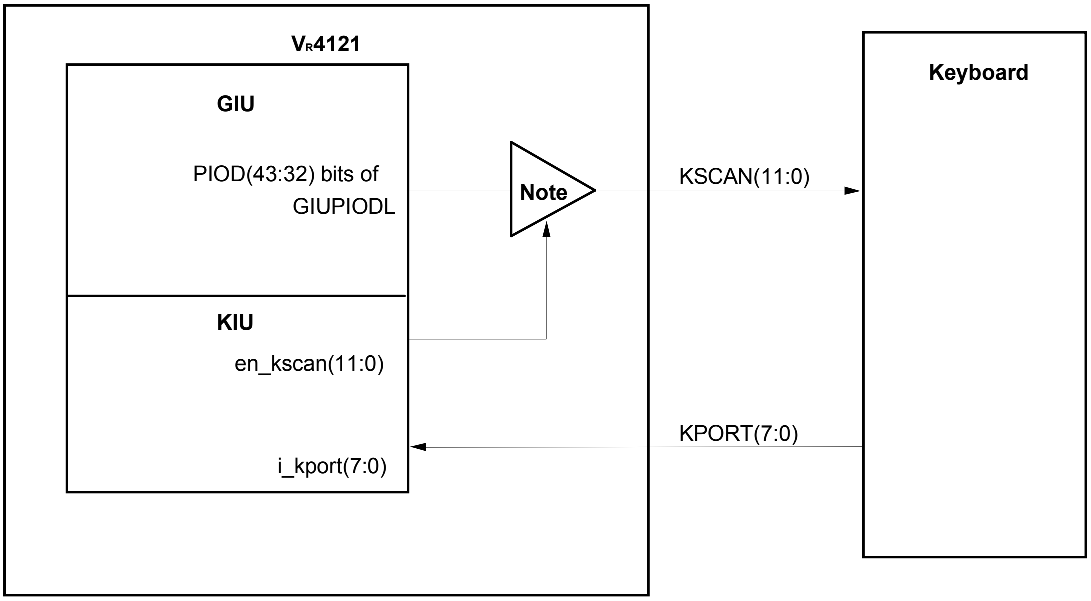
Bit	Pin
KGPEN11	KSCAN11/GPIO43
KGPEN10	KSCAN10/GPIO42
KGPEN9	KSCAN9/GPIO41
KGPEN8	KSCAN8/GPIO40
KGPEN7	KSCAN7/GPIO39
KGPEN6	KSCAN6/GPIO38
KGPEN5	KSCAN5/GPIO37
KGPEN4	KSCAN4/GPIO36
KGPEN3	KSCAN3/GPIO35
KGPEN2	KSCAN2/GPIO34
KGPEN1	KSCAN01/GPIO33
KGPEN0	KSCAN00/GPIO32

However, the number of pins used as KSCAN pin or output port is limited by the SCANLINE register's setting.
For details, refer to **22.2.9 SCANLINE (0x0B00 019E)**.

KIU controls enable/disable of the output buffer of the KSCAN(11:0)/GPIO(43:32) pins with the internal signal (en_kscan(11:0) of the figure below) according to the setting of the KGPEN(11:0) bits.

One en_kscan signal corresponds to each KGPEN bit. The en_kscan signal that is set for a KSCAN pin by the KGPEN bit outputs the high level during a scan, and outputs the low level during a wait time. On the other hand, the en_kscan signal that is set for a general output port by the KGPEN bit always outputs the high level.

Figure 22-4. Connection of Keyboard Interface Pin



Note The output of this buffer is enabled with a high-level signal (active-high).
When en_kscan(11:0) is high, this buffer outputs data to the KSCAN pin.
When en_kscan(11:0) is low, the KSCAN pin goes into the high-impedance state.

22.2.9 SCANLINE (0x0B00 019E)

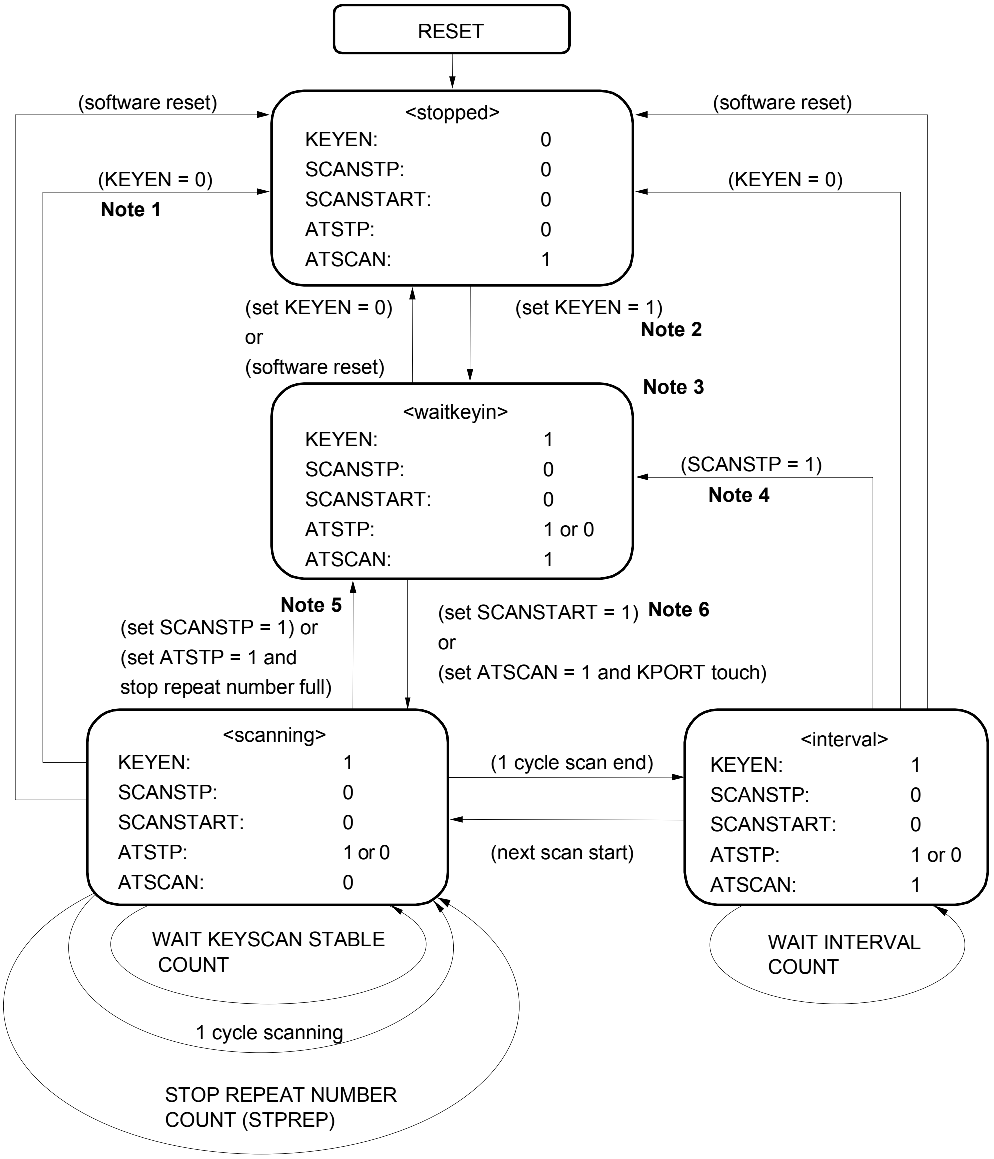
Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	LINE1	LINE0
R/W	R	R	R	R	R	R	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:2	RFU	Write 0 to these bits. 0 is returned after a read.
1:0	LINE(1:0)	KSCAN pin use/do not use setting 11: Do not use KSCAN pins for key scan All the KIU's KSCAN pins are used as output ports. 10: Use eight key scan pins (KSCAN(7:0)) Key scan uses eight key scan pins (supports 64 keys) The remaining four pins can be used as an output port. 01: Use ten key scan pins (KSCAN(9:0)) Key scan uses ten key scan pins (supports 80 keys) The remaining two pins can be used as an output port. 00: Use twelve key scan pins (KSCAN(11:0)) Key scan uses twelve key scan pins (supports 96 keys) No pins can be used as an output port.

This register is used to switch the number of scan lines.
When using KSCAN pins for key scan, set the bits in the KIUGPEN register to 0.

Figure 22-5. Transition of Sequencer Status



<Description>

<stopped>:	KIUSCANS register bit 1 = 0, bit 0 = 0
<waitkeyin>:	KIUSCANS register bit 1 = 0, bit 0 = 1
<interval>:	KIUSCANS register bit 1 = 1, bit 0 = 0
<scanning>:	KIUSCANS register bit 1 = 1, bit 0 = 1
KEYEN:	KIUSCANREP register bit 15
STPREP:	KIUSCANREP register bits 9, 8, 7, 6, 5, 4
SCANSTP:	KIUSCANREP register bit 3
SCANSTART:	KIUSCANREP register bit 2
ATSTP:	KIUSCANREP register bit 1
ATSCAN:	KIUSCANREP register bit 0
software reset:	KIURST bit 0 = 1 write
KPORT touch:	When any of KPORT(7:0) pins is "1"
stop repeat: number full	When the scan data is "0" for the number of times specified by the STPREP register

Notes 1. When the KETEN is set to 0 during a scanning operation, the status changes to the stopped status after that scanning operation has completed.

2. The KETEN bit cannot be set to 1 while both bits 1 and 0 of the SCANLINE register are 1.

3. When the status changes from the waitkeyin mode to the scanning mode after the SCANSTP bit is set to 1, the status returns to the waitkeyin mode again after scanning a set of data.

4. When the SCANSTP bit is set to 1 in the interval mode, the status changes to the waitkeyin mode and the SCANSTP bit becomes 0 automatically.

5. If the SCANSTP bit is set to 1 during a scanning operation, that one set of data scanning is continued. After this scanning is completed, the status changes to the waitkeyin mode and the SCANSTP bit becomes 0 automatically.

6. The SCANSTART bit becomes automatically 0 when the status changes to the scanning mode, except if the SCANSTART bit was set to 1 during the interval or scanning mode.

Figure 22-6. Basic Operation Timing Chart (1/2)
(a) Auto Start/Auto Stop

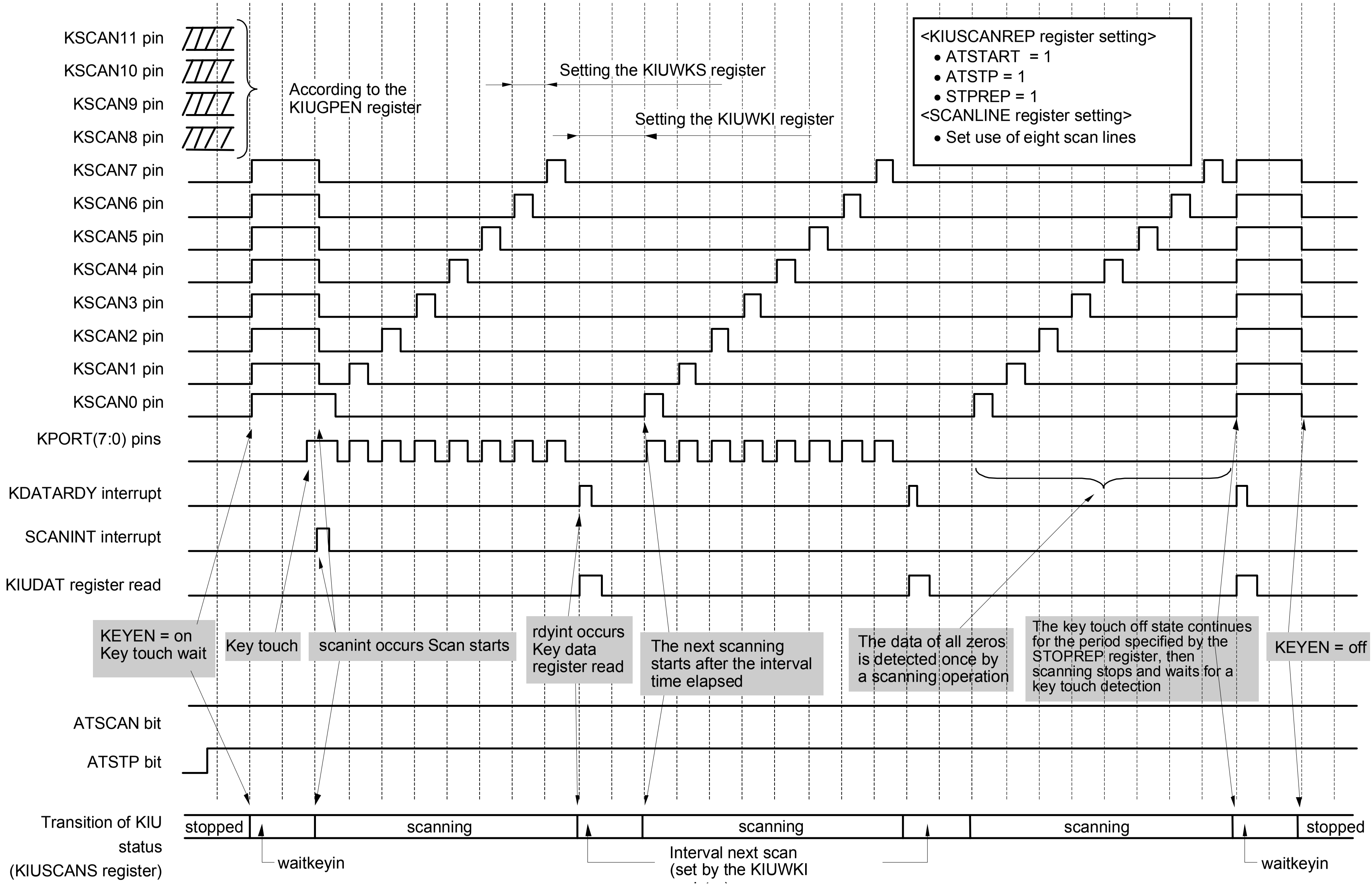
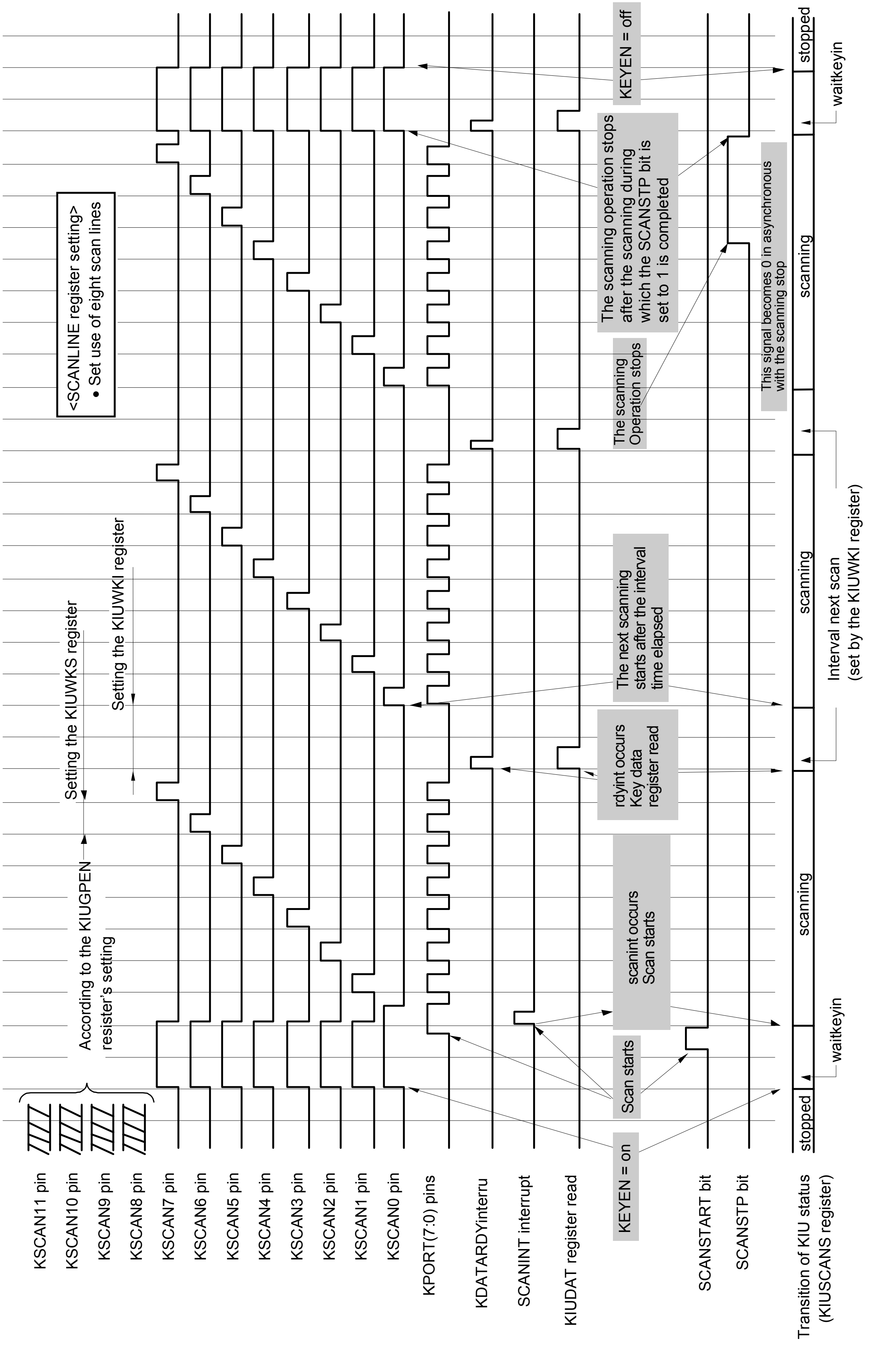

KSCAN11 pin
KSCAN10 pin
KSCAN9 pin
KSCAN8 pin
KSCAN7 pin
KSCAN6 pin
KSCAN5 pin
KSCAN4 pin
KSCAN3 pin
KSCAN2 pin
KSCAN1 pin
KSCAN0 pin
KPORT(7:0) pins
KDATARDY interrupt
SCANINT interrupt
KIUDAT register read
ATSCAN bit
ATSTP bit
Transition of KIU status (KIUSCANS register)
According to the KIUGPEN register
Setting the KIUWKS register
Setting the KIUWKI register
<KIUSCANREP register setting>
• ATSTART = 1
• ATSTP = 1
• STPREP = 1
<SCANLINE register setting>
• Set use of eight scan lines
KEYEN = on Key touch wait
Key touch
scanint occurs Scan starts
rdyint occurs Key data register read
The next scanning starts after the interval time elapsed
The data of all zeros is detected once by a scanning operation
The key touch off state continues for the period specified by the STOPREP register, then scanning stops and waits for a key touch detection
KEYEN = off
stopped
scanning
scanning
scanning
stopped
waitkeyin
Interval next scan (set by the KIUWKI
waitkeyin

Figure 22-6. Basic Operation Timing Chart (2/2)
(b) Auto Start/Auto Stop



[MEMO]

CHAPTER 23 DSIU (DEBUG SERIAL INTERFACE UNIT)

23.1 General

The DSIU (debug serial interface unit) supports transfer rates of up to 115 kbps. In addition to the DDIN and DDOUT input/output pins, the DSIU supports the DCTS# and DRTS# pins that are used for hardware flow control.

23.2 Register Set

The DSIU registers are listed below.

Table 23-1. DSIU Registers

Address	R/W	Register Symbols	Function
0x0B00 01A0	R/W	PORTREG	Port Change Register
0x0B00 01A2	R	MODEMREG	Modem Control Register
0x0B00 01A4	R/W	ASIM00REG	Asynchronous Mode 0 Register
0x0B00 01A6	R/W	ASIM01REG	Asynchronous Mode 1 Register
0x0B00 01A8	R	RXB0RREG	Receive Buffer Register (Extended)
0x0B00 01AA	R	RXB0LREG	Receive Buffer Register
0x0B00 01AC	R/W	TXS0RREG	Transmit Data Register (Extended)
0x0B00 01AE	R/W	TXS0LREG	Transmit Data Register
0x0B00 01B0	R	ASIS0REG	Status Register
0x0B00 01B2	R/W	INTR0REG	Debug SIU Interrupt Register
0x0B00 01B6	R/W	BPRM0REG	Baud rate Generator Prescaler Mode Register
0x0B00 01B8	R/W	DSIURESETREG	Debug SIU Reset Register

These registers are described in detail below.

23.2.1 PORTREG (0x0B00 01A0)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	CDDIN	CDDOUT	CDRTS	CDCTS
R/W	R	R	R	R	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15:4	RFU	Write 0 to these bits. 0 is returned after a read.
3	CDDIN	This pin is used to switch the DDIN pin for use as a general-purpose output pin. 1: General-purpose output 0: DDIN
2	CDDOUT	This pin is used to switch the DDOUT pin for use as a general-purpose output pin. 1: General-purpose output 0: DDOUT
1	CDRTS	This pin is used to switch the DRTS# pin for use as a general-purpose output pin. 1: General-purpose output 0: DRTS#
0	CDCTS	This pin is used to switch the DCTS# pin for use as a general-purpose output pin. 1: General-purpose output 0: DCTS#

Note Previous value is retained.

This register is used to switch the DSIU pin for use as a general-purpose output pin.
Note that the output value should be set in the GIU when the DSIU pins are set to general-purpose outputs.

23.2.2 MODEMREG (0x0B00 01A2)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	DRTS	DCTS
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	1	1
After reset	0	0	0	0	0	0	1	1

Bit	Name	Function
15:2	RFU	Write 0 to these bits. 0 is returned after a read.
1	DRTS	DRTS# pin output 1: High level 0: Low level
0	DCTS	DCTS# pin input 1: High level 0: Low level

This register is used for flow control and can be used to pass signals between the VR4121 and external agents. The setting of the RXE0 bit in ASIM00REG reflects on the DRTS#'s output.

23.2.3 ASIM00REG (0x0B00 01A4)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RXE0	PS01	PS00	CL0	SL0	RFU	RFU
R/W	R	R/W	R/W	R/W	R/W	R/W	R	R
RTCRST	1	0	0	0	0	0	0	0
After reset	1	0	0	0	0	0	0	0

Bit	Name	Function
15:8	RFU	Write 0 to these bits. 0 is returned after a read.
7	RFU	Write 1 to this bit. 1 is returned after a read.
6	RXE0	Debug serial reception enable 1: Enable 0: Prohibit
5:4	PS0(1:0)	Debug serial parity select 11: Even parity 10: Odd parity 01: Zero parity bits during transmit No parity during receive 00: No parity. Set to 00 for extended-bit operations
3	CL0	Debug serial character length setting 1: 8 bits 0: 7 bits
2	SL0	Debug serial stop bit setting 1: 2 bits 0: 1 bit
1:0	RFU	Write 0 to these bits. 0 is returned after a read.

This register is used to make various serial communication settings for debugging.

The setting of RXE0 reflects on the DRTS#'s output. Setting this bit to 1 (reception enable) makes the DRTS# pin output 0, and clearing to 0 (reception prohibit) makes DRTS# output 1.

If this register is changed during transmission or reception of serial data for debugging, the DSIU's operations cannot be guaranteed.

23.2.4 ASIM01REG (0x0B00 01A6)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	EBS0
R/W	R	R	R	R	R	R	R	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:1	RFU	Write 0 to these bits. 0 is returned after a read.
0	EBS0	Extended bit operation enable 1: Enable 0: Prohibit

This register is used to set extended bit operations for the DSIU.

When "1" is set to the EBS0 bit, one bit is added to the 8-bit data length for transmission and reception to enable operations using 9-bit data. Extended-bit operations are valid only when "00" has been set to ASIM00REG's PS(1:0) bits. If a value other than "00" has been set to ASIM00REG's PS(1:0) bits, the EBS0 bit specification is ignored and extended-bit operations cannot be performed.

23.2.5 RXB0RREG (0x0B00 01A8)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RXB08
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RXB07	RXB06	RXB05	RXB04	RXB03	RXB02	RXB01	RXB00
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:9	RFU	Write 0 to these bits. 0 is returned after a read.
8:0	RXB0(8:0)	Receive data

This register is used to store debug serial receive data.

The RXB08 bit stores the extended bit during extended-bit operations and stores a zero during 7- or 8-bit character reception. The RXB07 bit stores a zero during 7-bit character reception.

23.2.6 RXB0LREG (0x0B00 01AA)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RXB0L7	RXB0L6	RXB0L5	RXB0L4	RXB0L3	RXB0L2	RXB0L1	RXB0L0
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:8	RFU	Write 0 to these bits. 0 is returned after a read.
7:0	RXB0L(7:0)	Receive data

This register is used to store debug serial receive data.

The RXB0L7 bit stores a zero during 7-bit character reception.

The only difference between this register and RXB0RREG is that this register does not support extended-bit operations.

23.2.7 TXS0RREG (0x0B00 01AC)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	TXS08
R/W	R	R	R	R	R	R	R	R/W
RTCRST	0	0	0	0	0	0	0	1
After reset	0	0	0	0	0	0	0	1

Bit	7	6	5	4	3	2	1	0
Name	TXS07	TXS06	TXS05	TXS04	TXS03	TXS02	TXS01	TXS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	1	1	1	1	1	1	1	1
After reset	1	1	1	1	1	1	1	1

Bit	Name	Function
15:9	RFU	Write 0 to these bits. 0 is returned after a read.
8:0	TXS0(8:0)	Transmit data

This register is used to store debug serial transmit data.

The TXS08 bit is used to transmit the extended bit during extended-bit operations.

23.2.8 TXS0LREG (0x0B00 01AE)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	TXS0L7	TXS0L6	TXS0L5	TXS0L4	TXS0L3	TXS0L2	TXS0L1	TXS0L0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	1	1	1	1	1	1	1	1
After reset	1	1	1	1	1	1	1	1

Bit	Name	Function
15:8	RFU	Write 0 to these bits. 0 is returned after a read.
7:0	TXS0L(7:0)	Transmit data

This register is used to store debug serial transmit data.

The only difference between this register and TXS0RREG is that this register does not support extended-bit operations.

23.2.9 ASIS0REG (0x0B00 01B0)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	SOT0	RFU	RFU	RFU	RFU	PE0	FE0	OVE0
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:8	RFU	Write 0 to these bits. 0 is returned after a read.
7	SOT0	Transmit mode status 1: Transmission start 0: Transmission complete
6:3	RFU	Write 0 to these bits. 0 is returned after a read.
2	PE0	Parity error status 1: Parity error 0: Normal
1	FE0	Framing error status 1: Framing error 0: Normal
0	OVE0	Overrun error status 1: Overrun error status 0: Normal

This register indicates the debug serial transmit/receive status.

A write to the TXS0RREG or TXS0LREG register sets "1" to the SOT0 bit. When the transmission is completed, "1" is set to the INTR0REG register's INTST0 bit and the SOT0 bit is cleared to zero. This bit can be used as a means of determining whether or not it is possible to write to the transmission shift register when transmitting data in debug serial mode.

If the received data contains a parity error, "1" is set to the PE0 bit. If the stop bit is not detected, "1" is set to the FE0 bit.

An overrun error occurs and "1" is set to the OVE0 bit if the sequencer completes the next receive processing before receive data is read from the receive buffer. When an overrun error occurs, the old data in the receive buffer is overwritten by the newly received data.

23.2.10 INTR0REG (0x0B00 01B2)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	INTDCD	INTSER0	INTSR0	INTST0
R/W	R	R	R	R	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:4	RFU	Write 0 to these bits. 0 is returned after a read.
3	INTDCD	CTS# change interrupt. Cleared to 0 when 1 is written. 1: CTS change interrupt 0: Normal
2	INTSER0	Debug serial receive error interrupt. Cleared to 0 when 1 is written. 1: Error interrupt 0: Normal
1	INTSR0	Debug serial receive complete interrupt. Cleared to 0 when 1 is written. 1: Receive complete 0: Other
0	INTST0	Debug serial transmit complete interrupt. Cleared to 0 when 1 is written. 1: Transmit complete 0: Other

This register indicates interrupt events that occur during debug serial transfer.

When debug serial operations are in the reception-enable mode, and either the PE0 bit, FE0 bit, or OVE0 bit in the ASIS0REG has been set, "1" is set to the INTSER0 bit.

When debug serial operations are in the reception-enable mode, and receive data is transferred to the receive buffer, "1" is set to the INTSR0 bit. When one frame of transmit data is sent from the transmit register, "1" is set to the INTST0 bit.

When CTS# (flow control signal from an external agent) is changed, "1" is set to the INTDCD bit.

23.2.11 BPRM0REG (0x0B00 01B6)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	BRCE0	RFU	RFU	RFU	RFU	BPR02	BPR01	BPR00
R/W	R/W	R	R	R	R	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:8	RFU	Write 0 to these bits. 0 is returned after a read.
7	BRCE0	Baud rate generator count enable 1: Enable 0: Prohibit
6:3	RFU	Write 0 to these bits. 0 is returned after a read.
2:0	BPR0(2:0)	Debug serial baud rate setting 111: 115,200 bps 110: 57,600 bps 101: 38,400 bps 100: 19,200 bps 011: 9,600 bps 010: 4,800 bps 001: 2,400 bps 000: 1,200 bps

This register is used to set the baud rate for debug serial communications.

DSIU's operations are not guaranteed if the baud rate is changed during transmission or reception.

23.2.12 DSIURESETREG (0x0B00 01B8)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	DSIURST
R/W	R	R	R	R	R	R	R	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:1	RFU	Write 0 to these bits. 0 is returned after a read.
0	DSIURST	DSIU reset. Cleared to 0 when 1 is written. 1: Reset 0: Normal

This register is used to reset DSIU forcibly.

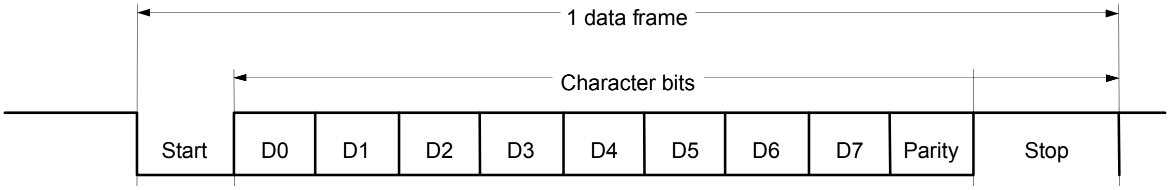
23.3 Description of Operations

23.3.1 Data format

Serial data is transmitted and received in full-duplex mode.

The format of the transmit and receive data is shown in the following figure. Each frame includes a start bit, character bits, parity bit, and stop bit(s). Specification of the character bit length in one data frame, along with the parity setting, and stop bit length specification are all made via the mode registers (ASIM00REG and ASIM01REG).

Figure 23-1. Data Format for Transmission and Reception



- Start bit (Start): 1 bit
- Character bits (Dn): 7, 8, or 9 bits (when using extended bit) (n = 0 to 8)
- Parity bit (Parity): Even parity, odd parity, zero parity, or no parity
- Stop bit(s) (Stop): 1 bit or 2 bits

23.3.2 Transmission

After the SOT0 pin value is confirmed as "0", writing data to a transmission shift register (TXS0REG or TXS0LREG) activates transmission via the DDOUT pin. Use the transmit complete interrupt (Dsiu_Intsr0) service routine to write the next data to TXS0REG or TXS0LREG.

Transmission enable status

The DSIU unit is always set to transmission enable status. The DCTS# pin is used when it is necessary to confirm that the remote side is ready to receive.

Activation of transmit operation

Writing data to a transmission shift register (TXS0REG or TXS0LREG) activates the transmit operation. The transmit data is sent in LSB-first order, beginning with the start bit. The start bit, parity bit, and stop bit(s) are added automatically.

Transmit complete interrupt request

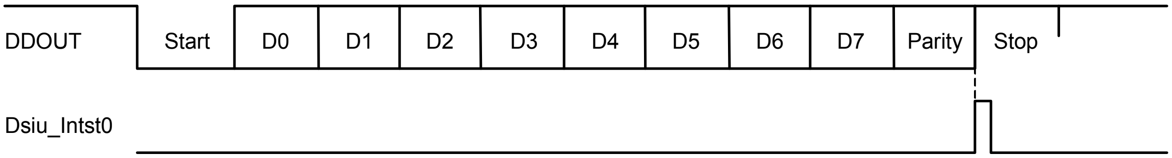
Once one frame of data has been sent, a transmit complete interrupt request (Dsiu_Intst0) occurs. If the next data to be transmitted is then not written to TXS0REG or TXS0LREG, the transmit operation is halted and the transmission rate is lowered.

Cautions 1. Normally, the transmit complete interrupt request (Dsiu_Intst0) occurs when the TXS0REG or TXS0LREG register is empty. However, if a reset is input, the transmit complete interrupt request (Dsiu_Intst0) will not occur even when the transmission shift register (TXS0REG or TXS0LREG) is empty.

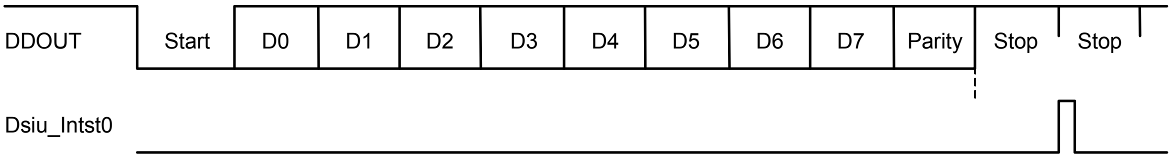
2. Writing to either TXS0REG or TXS0LREG is prohibited during a transmit operation until Dsiu_Intst0 occurs.

Figure 23-2. Transmit Complete Interrupt Timing

(a) Stop bit length: 1



(b) Stop bit length: 2



23.3.3 Reception

Once reception is enabled, sampling of the DDIN pin begins and, when a start bit is detected, data reception begins. A receive complete interrupt request (Dsiu_Intst0) occurs each time reception of one frame of data is completed. Normally, this interrupt service is used to transfer receive data from a receive buffer (RXB0REG or RXB0LREG) to memory.

Reception enable status

Setting the ASIM00REG's bit6 sets enable status for the receive operation, and a zero is output to DRTS#.

RXE0 = 1:	Reception enable status	DRTS# = 0
RXE0 = 0:	Reception prohibit status	DRTS# = 1

The reception hardware is initialized and enters idle mode when reception prohibit status has been set. Once that happens, receive complete interrupts and receive error interrupts are not issued and the contents of the receive buffer are retained.

Activation of receive operation

The receive operation is activated when a start bit is detected.

The DDIN pin is sampled at the interval set by the serial clock specified via ASIM00REG. Once a signal's falling edge is detected at the DDIN pin, the DDIN pin is again sampled after an interval of eight serial clocks. This time, when a low-level state is detected it is recognized as a start bit and control is passed to the receive operation, after which the DDIN pin input continues to be sampled using an interval of 16 serial clocks.

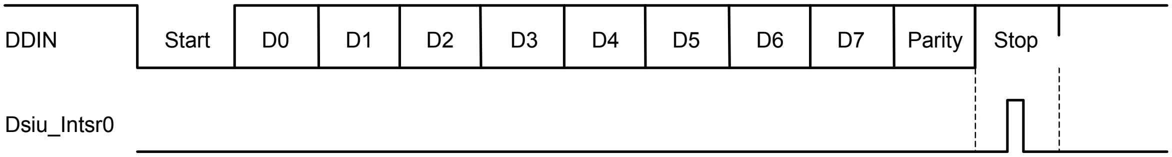
After eight serial clocks have elapsed since a signal's falling edge was detected at the DDIN pin, when sampling recognizes a high-level state it does not recognize the signal's falling edge as a start bit. Instead, the serial clock counter used for the sampling timing is initialized and the receive operation is halted until the next edge input.

Receive complete interrupt request

When RXE0 = 1 and one frame of data has been received, the receive data in the shift register is transferred to a receive buffer (RXB0REG or RXB0LREG) and a receive complete interrupt request (Dsiu_Intsr0) is issued. Even when an error has occurred, the receive data for which the error occurred is still transferred to a receive buffer (RXB0REG or RXB0LREG) and two interrupts; a receive complete interrupt (Dsiu_Intsr0) and a receive error interrupt (Dsiu_Intser0), occur at the same time.

If the RXE0 bit is reset (to "0") during a receive operation, the receive operation is halted immediately. At that point, the contents of the receive buffer (RXB0REG or RXB0LREG) and ASIS0REG are not changed and neither the receive complete interrupt (Dsiu_Intsr0) nor the receive error interrupt (Dsiu_Intser0) occur.

Figure 23-3. Receive Complete Interrupt Timing



Receive error flag

Receive operations can be affected by three types of error flags that are set during the receive operations: a parity error flag, a framing error flag, and an overrun error flag.

A receive error interrupt request is issued after these three types of error flags are ORed.

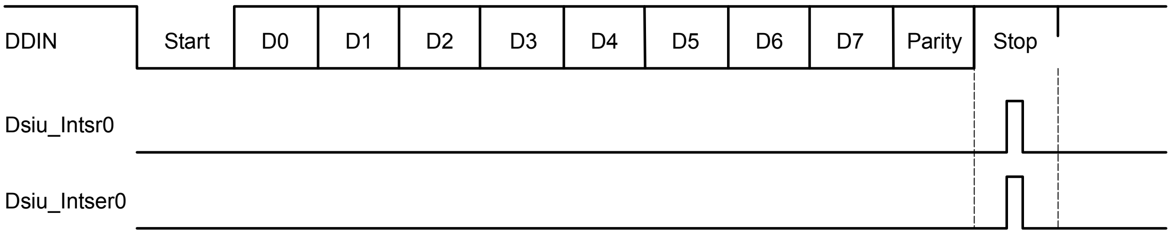
During receive error interrupt service (Dsiu_Intser0), the contents of ASIS0REG can be read to detect which kind of error occurred during reception.

The contents of ASIS0REG are reset (to "0") when the receive buffer (RXB0REG or RXB0LREG) is read or when the next data is received (another error flag is set if the next data also contains an error).

Table 23-2. Receive Error Causes

Receive Error	Cause
Parity error	Parity specified during reception does not match parity of receive data
Framing error	Stop bit is not detected
Overrun error	Reception of the next data is completed before data is read from the receive buffer

Figure 23-4. Receive Error Timing



[MEMO]

CHAPTER 24 LED (LED CONTROL UNIT)

24.1 General

LEDs are switched on and off at a regular interval. The interval can be set as programmable.

24.2 Register Set

The LED registers are listed below.

Table 24-1. LED registers

Address	R/W	Register Symbols	Function
0x0B00 0240	R/W	LEDHTSREG	LED H Time Set register
0x0B00 0242	R/W	LEDLTSREG	LED L Time Set register
0x0B00 0248	R/W	LEDCNTREG	LED Control register
0x0B00 024A	R/W	LEDASTCREG	LED Auto Stop Time Count register
0x0B00 024C	R/W	LEDINTREG	LED Interrupt register

These registers are described in detail below.

24.2.1 LEDHTSREG (0x0B00 0240)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	HTS4	HTS3	HTS2	HTS1	HTS0
R/W	R	R	R	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	1	0	0	0	0
After reset	0	0	0	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15:5	RFU	Write 0 to these bits. 0 is returned after a read.
4:0	HTS(4:0)	Values compared to bits 15 to 11 of LED HL Time Count. 11111: 1.9735 seconds : 10000: 1 second : 01000: 0.5 seconds : 00100: 0.25 seconds : 00010: 0.125 seconds 00001: 0.0625 seconds 00000: Prohibit

Note Previous value is retained.

This register is used to set the LED's ON time (high level width of LEDOUT#).

The ON time ranges from 0.0625 to 1.9735 seconds and can be set in 0.0625-second units. The initial value is 1 second.

This register must not be changed once the LEDENABLE bit of LEDCNTREG has been set to 1 as "enable". The operation is not guaranteed if a change is made after that point.

24.2.2 LEDLTSREG (0x0B00 0242)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	LTS6	LTS5	LTS4	LTS3	LTS2	LTS1	LTS0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	1	0	0	0	0	0
After reset	0	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**	**Note**

Bit	Name	Function
15:7	RFU	Write 0 to these bits. 0 is returned after a read.
6:0	LTS(6:0)	Values compared to bits 17 to 11 of LED HL Time Count. 111111 : 7.9375 seconds : 1000000: 4 seconds : 0100000: 2 seconds : 0010000: 1 second : 0001000: 0.5 seconds : 0000100: 0.25 seconds : 0000010: 0.125 seconds 0000001: 0.0625 seconds 0000000: Prohibit

Note Previous value is retained.

This register is used to set the LED's OFF time (low level width of LEDOUT#).

The OFF time ranges from 0.0625 to 7.9375 seconds and can be set in 0.0625-second units. It should be set by means of software. The initial value is 2 seconds.

This register must not be changed once the LEDENABLE bit of LEDCNTREG has been set as "enable". The operation is not guaranteed if a change is made after that point.

24.2.3 LEDCNTREG (0x0B00 0248)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	LEDSTOP	LEDENABLE
R/W	R	R	R	R	R	R	R/W	R/W
RTCRST	0	0	0	0	0	0	1	0
After reset	0	0	0	0	0	0	**Note**	**Note**

Bit	Name	Function
15:2	RFU	Write 0 to these bits. 0 is returned after a read.
1	LEDSTOP	LED ON/OFF auto stop setting 1: ON 0: OFF
0	LEDENABLE	LED ON/OFF (blink) setting 1: Blink 0: Does not blink

Note Previous value is retained.

This register is used to make various LED settings.

Caution When setting up LED activation, make sure that a value other than zero has already been set to the LEDHTSREG, LEDLTSREG, and LEDASTCREG. The operation is not guaranteed if zero is set to these registers.

24.2.4 LEDASTCREG (0x0B00 024A)

Bit	15	14	13	12	11	10	9	8
Name	ASTC15	ASTC14	ASTC13	ASTC12	ASTC11	ASTC10	ASTC9	ASTC8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	1	0	0
After reset	0	0	0	0	0	1	0	0

Bit	7	6	5	4	3	2	1	0
Name	ASTC7	ASTC6	ASTC5	ASTC4	ASTC3	ASTC2	ASTC1	ASTC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	1	0	1	1	0	0	0	0
After reset	1	0	1	1	0	0	0	0

Bit	Name	Function
15:0	ASTC(15:0)	LED auto stop time count bit

This register is a 16-bit down counter that sets the number of ON/OFF times prior to automatic stopping of LED activation. The set value is read during a read. The initial setting is 1,200 times (ON/OFF pairs) in which each time includes one second of ON time and two seconds of OFF time.

The pair of operations in which the LED is switched ON once and OFF once is counted as "1" by this counter. The counter counts down from the set value and an LEDINT interrupt occurs when it reaches zero.

Caution Setting a zero to this register is prohibited. The operation is not guaranteed if zero is set to this register.

24.2.5 LEDINTREG (0x0B00 024C)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

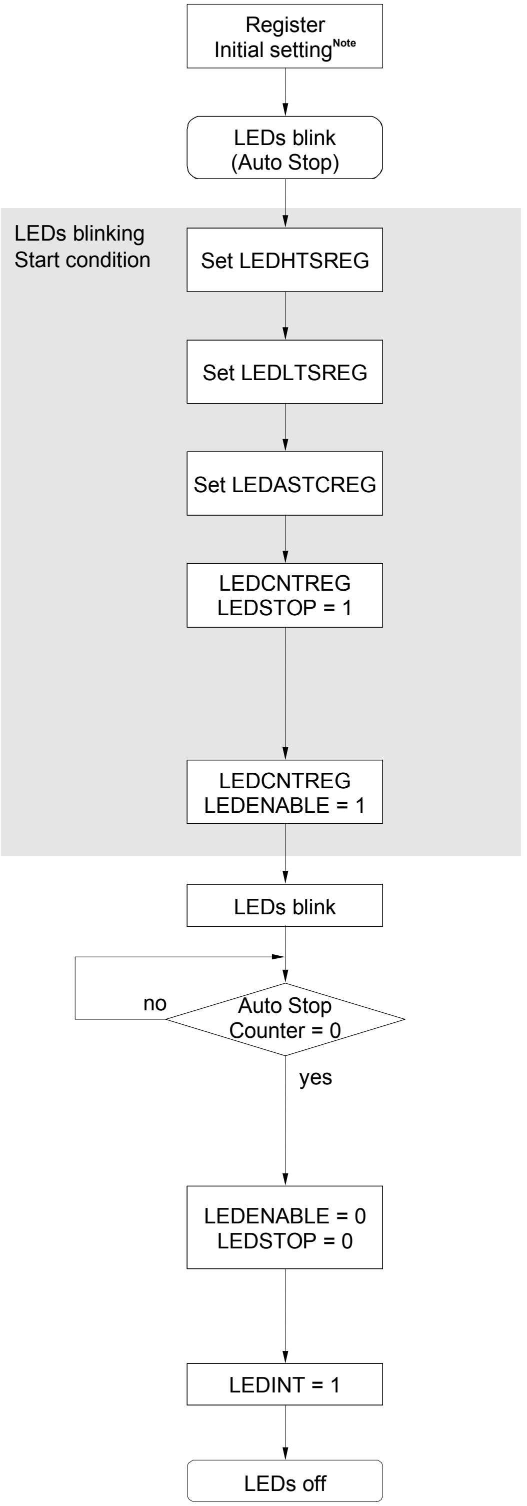
Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	LEDINT
R/W	R	R	R	R	R	R	R	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:1	RFU	Write 0 to these bits. 0 is returned after a read.
0	LEDINT	Auto stop interrupt request. Cleared to 0 when 1 is written. 1: Yes 0: No

This register indicates when an auto stop interrupt request has occurred.

An auto stop interrupt request occurs if "1" has already been set to the LEDSTOP bit and the LEDENABLE bit of LEDCNTREG when LEDASTCREG is cleared to "0". When this interrupt occurs, the LEDSTOP bit and the LEDENABLE bit of LEDCNTREG are both cleared to "0".

24.3 Operation Flow



Register initial setting
Set the initial value to each register

- LEDHTSREG: 0x0010(or LED lighting time)
- LEDLTSREG: 0x0020(or LED off time)
- LEDHLTCLREG: 0x0000
- LEDHLTCHREG: 0x0000
- LEDCNTREG: 0x0002
- LEDASTCREG: 0x04B0

LED blinking time setting

- LEDHTSREG
  Sets LED lighting time.
- LEDLTSREG
  Sets LED off time.
- LEDASTCREG
  Sets number of LEDs blinking.

Caution
Setting these registers to 0 is prohibited because this operation may cause undefined operation.

LED auto-stop setting

- LEDSTOP
  Sets the LED blink auto-stop function to enable.
  This setting terminates LED blinking automatically after blinking time set above has elapsed.

LED blinking start

- LEDENABLE
  Starts LED blinking.

LED blinking

- Supervising the auto-stop counter
  LED blinking terminates when the auto-stop counter reaches "0".

Caution
Setting the LEDENABLE or LEDSTOP bit to 0 is prohibited because this operation may cause undefined operation.

LED blinking termination

- LEDENABLE = "0"
  Terminates LED blinking.

LED blinking terminate interrupt generation
LEDINT = "1"

- Generates an interrupt request for the ICU unit.

Note Initial setting for each register must be performed only when a power is supplied to device for the first time, regardless whether LEDs blinking function is used or not.

[MEMO]

CHAPTER 25 SIU (SERIAL INTERFACE UNIT)

25.1 General

The SIU is a serial interface that conforms to the RS-232-C communication standard and is equipped with two one-channel interfaces, one for transmission and one for reception.

This unit is functionally compatible with the NS16550.

25.2 Register Set

The SIU registers are listed below.

Table 25-1. SIU Registers

Address	LCR7	R/W	Register Symbols	Function
0x0C00 0000	0	R	SIURB	Receiver Buffer Register (Read)
		W	SIUTH	Transmitter Holding Register (Write)
	1	R/W	SIUDLL	Divisor Latch (Least Significant Byte)
0x0C00 0001	0	R/W	SIUIE	Interrupt Enable
	1	R/W	SIUDLM	Divisor Latch (Most Significant Byte)
0x0C00 0002	–	R	SIUIID	Interrupt Identification Register (Read)
	–	W	SIUFC	FIFO Control Register (Write)
0x0C00 0003	–	R/W	SIULC	Line Control Register
0x0C00 0004	–	R/W	SIUMC	MODEM Control Register
0x0C00 0005	–	R/W	SIULS	Line Status Register
0x0C00 0006	–	R/W	SIUMS	MODEM Status Register
0x0C00 0007	–	R/W	SIUSC	Scratch Register
0x0C00 0008	–	R/W	SIUIRSEL	SIU/FIR IrDA Selector
0x0C00 0009	–	R/W	SIURESET	SIU Reset Register
0x0C00 000A	–	R/W	SIUCSEL	SIU Echo-Back Control Register

Remark LCR7 is the bit 7 of SIULC register.

These registers are described in detail below.

25.2.1 SIURB (0x0C00 0000: LCR7 = 0, Read)

Bit	7	6	5	4	3	2	1	0
Name	RXD7	RXD6	RXD5	RXD4	RXD3	RXD2	RXD1	RXD0
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
7:0	RXD(7:0)	Serial receive data

This register stores receive data used in serial communications.

To access this register, set LCR7 (bit 7 of SIULC register) to 0.

25.2.2 SIUTH (0x0C00 0000: LCR7 = 0, Write)

Bit	7	6	5	4	3	2	1	0
Name	TXD7	TXD6	TXD5	TXD4	TXD3	TXD2	TXD1	TXD0
R/W	W	W	W	W	W	W	W	W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
7:0	TXD(7:0)	Serial transmit data

This register stores transmit data used in serial communications.

To access this register, set LCR7 (bit 7 of SIULC register) to 0.

25.2.3 SIUDLL (0x0C00 0000: LCR7 = 1)

Bit	7	6	5	4	3	2	1	0
Name	DLL7	DLL6	DLL5	DLL4	DLL3	DLL2	DLL1	DLL0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
7:0	DLL(7:0)	Baud rate generator divisor (low-order byte)

This register is used to set the divisor (division rate) for the baud rate generator.

The data in this register and the upper 8-bit data in SIUDLM register are together handled as 16-bit data.

To access this register, set LCR7 (bit 7 of SIULC register) to 1.

25.2.4 SIUIE (0x0C00 0001: LCR7 = 0)

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	IE3	IE2	IE1	IE0
R/W	R	R	R	R	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
7:4	RFU	Write 0 to these bits. 0 is returned after read.
3	IE3	Modem status interrupt 1: Interrupt enable 0: Interrupt prohibit
2	IE2	Receive status interrupt 1: Interrupt enable 0: Interrupt prohibit
1	IE1	Transmitter holding register empty interrupt 1: Interrupt enable 0: Interrupt prohibit
0	IE0	Receive data interrupt or timeout interrupt in FIFO mode 1: Interrupt enable 0: Interrupt prohibit

This register is used to specify interrupt enable/prohibit settings for the five types of interrupt used by SIU.

Each interrupt can be used by setting the corresponding bit to 1.

Overall use of interrupt functions can be halted by setting bits 0 to 3 of this register (IER) to zero.

When interrupts are prohibited, "pending" is not displayed in the IIR0 bit even when the interrupt condition has been met and INTR output does not become active.

Other functions in the system are not affected even though interrupts are prohibited and the settings in the line status register and modem status register are valid.

To access this register, set LCR7 (bit 7 of SIULC register) to 0.

25.2.5 SIUDLM (0x0C00 0001: LCR7 = 1)

Bit	7	6	5	4	3	2	1	0
Name	DLM7	DLM6	DLM5	DLM4	DLM3	DLM2	DLM1	DLM0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
7:0	DLM(7:0)	Baud rate generator divisor (high-order byte)

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 SIUDLL register are together handled as 16-bit data.

To access this register, set LCR7 (bit 7 of SIULC register) to 1.

The relationship between baud rates and SIUDLL and SIUDLM registers is as follows.

Table 25-2. Correspondence Between Baud Rates and Divisors

Baud Rate	Divisor	1-Clock Width (μs)
50	23,040	20,000
75	15,360	13,333
110	10,473	9,091
134.5	8,565	7,435
150	7,680	6,667
300	3,840	3,333
600	1,920	1,667
1,200	920	833
1,800	640	556
2,000	573	500
2,400	480	417
3,600	320	278
4,800	240	208
7,200	160	139
9,600	120	104
19,200	60	52.1
38,400	30	26.0
56,000	21	17.9
128,000	9	7.81
144,000	8	6.94
192,000	6	5.21
230,400	5	4.34
288,000	4	3.47
384,000	3	2.60
576,000	2	1.74
1,152,000	1	0.868

25.2.6 SIUIID (0x0C00 0002: Read)

Bit	7	6	5	4	3	2	1	0
Name	IIR7	IIR6	RFU	RFU	IIR3	IIR2	IIR1	IIR0
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	1
After reset	0	0	0	0	0	0	0	1

Bit	Name	Function
7:6	IIR(7:6)	Becomes 11 when FCR0 = 1
5:4	RFU	Write 0 to these bits. 0 is returned after read.
3	IIR3	Pending character timeout interrupt (in FIFO mode) 1: No pending interrupt 0: Pending interrupt
2:1	IIR(2:1)	Indicates the priority level of pending interrupt. See **Table 25-3**.
0	IIR0	Pending interrupts 1: No pending interrupt 0: Pending interrupt

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 contents of IIR3 bit is valid only in FIFO mode, and it is always 0 in 16,550 mode.

IIR2 bit becomes 1 when IIR3 bit is set to 1.

Table 25-3. Interrupt Function

SIUIID Register			Interrupt Set/Reset Function			
Bit 3 **Note**	Bit 2	Bit 1	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Control
0	1	1	Highest (1st)	Receive line status	Overrun error, parity error, framing error, or break interrupt	Read line status register
0	1	0	2nd	Receive data ready	Receive data exists or has reached the trigger level.	Read the receive buffer register or lower trigger level via FIFO.
1	1	0	2nd	Character timeout	During the time period for the four most recent characters, not one character has been read from the receive FIFO nor has a character been input to the receive FIFO. During this period, at least one character has been held in the receive FIFO.	Read receive buffer register
0	0	1	3rd	Transmit holding register empty	Transmit register is empty	Read IIR (if it is the interrupt source) or write to transmit holding register
0	0	0	4th	Modem status	CTS#, DSR#, or DCD#	Read modem status register

Note FIFO mode only.

25.2.7 SIUFC (0x0C00 0002: Write)

Bit	7	6	5	4	3	2	1	0
Name	FCR7	FCR6	RFU	RFU	FCR3	FCR2	FCR1	FCR0
R/W	W	W	R	R	W	W	W	W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
7:6	FCR(7:6)	Receive FIFO trigger level 11: 14 bytes 10: 8 bytes 01: 4 bytes 00: 0 bytes
5:4	RFU	Write 0 to these bits. 0 is returned after read.
3	FCR3	Switch between 16,550 mode and FIFO mode 1: From 16,550 mode to FIFO mode 0: From FIFO mode to 16,550 mode
2	FCR2	Transmit FIFO clear/counter clear. Cleared to 0 when 1 is written. 1: FIFO clear/counter clear 0: Normal
1	FCR1	Receive FIFO clear/counter clear. Cleared to 0 when 1 is written. 1: FIFO clear/counter clear 0: Normal
0	FCR0	Receive/Transmit FIFO enable 1: Enable 0: Disable

This register is used to control FIFO.

- **FIFO interrupt modes**

When receive FIFO is enabled and receive interrupts are enabled, receive interrupts can occur as described below.

1. When the FIFO is reached to the specified trigger level, a receive data ready interrupt occurs to inform the CPU.
   This interrupt is cleared when the FIFO goes below the trigger level.

2. When the FIFO is reached to the specified trigger level, the SIUIID register indicates a receive data ready interrupt.
   As with the interrupt above, SUIII is cleared when the FIFO goes below the trigger level.

3. Receive line status interrupts are assigned a higher priority level than are receive data ready interrupts.

4. When characters are transferred from the shift register to the receive FIFO, "1" is set to the LSR0 bit.
   The value of this bit returns to "0" when the FIFO becomes empty.

When receive FIFO is use-enabled and receive interrupts are enabled, receive FIFO timeout interrupts can occur as described below.

1. The following are conditions under which FIFO timeout interrupts occur.

   - At least one character is being stored in the FIFO.
   - The time required for sending four characters has elapsed since the serial reception of the last character (includes the time for two stop bits in cases where a stop bit has been specified).
   - The time required for sending four characters has elapsed since the CPU last accessed the FIFO.

   The time between receiving the last character and issuing a timeout interrupt is a maximum of 160 ms when operating at 300 baud and receiving 12-bit data.

2. The transfer time for a character is calculated based on the baud rate clock for reception (internal) input as clock signals (which is why the elapsed time is in proportion to the baud rate).

3. Once a timeout interrupt has occurred, the timeout interrupt is cleared and the timer is reset as soon as the CPU reads one character from the receive FIFO.

4. If no timeout interrupt has occurred, the timer is reset when a new character is received or when the CPU reads the receive FIFO.

When transmit FIFO is enabled and transmit interrupts are enabled, transmit interrupts can occur as described below.

1. When the transmit FIFO becomes empty, a transmit holding register empty interrupt occurs.
   This interrupt is cleared when a character is written to the transmit holding register (from one to 16 characters can be written to the transmit FIFO during servicing of this interrupt), or when SIUIID (interrupt ID register) is read.

2. If there are not at least two bytes of character data in the transmit FIFO between one time when LSR5 = 1 (transmit FIFO is empty) and the next time when LSR5 = 1, empty transmit FIFO status is reported to the IIR after a delay period calculated as "the time for one character – the time for the last stop bit(s)".
   When transmit interrupts are enabled, the first transmit interrupt that occurs after the FCR0 (FIFO enable bit) is overwritten is indicated immediately.

The priority level of the character timeout interrupt and receive FIFO trigger level interrupt is the same as that of the receive data ready interrupt.
The priority level of the transmit FIFO empty interrupt is the same as that of the transmit holding register empty interrupt.

Whether data to be transmitted exists or not in the transmit FIFO and the transmit shift register, check the transmit block empty bit (bit 6) of the SIULS register. It cannot be checked by the transmit holding register empty bit (bit 5) of the SIULS register, because this bit is used to check whether data to be transferred exists or not in the transmit FIFO. Therefore, this bit cannot check if there is data in the transmit shift register.

- **FIFO polling mode**

  When FCR0 = 1 (FIFO is enabled), if the value of any or all of the interrupt enable register (SIUIE) bits 3 to 0 becomes "0", SIU enters FIFO polling mode. Because the transmit block and receive blocks are controlled separately, polling mode can be set for either or both blocks.
  When in this mode, the status of the transmit block and/or receive block can be checked by reading the line status register (SIULS) via a user program.
  When in the FIFO polling mode, there is no notification when the trigger level is reached or when a timeout occurs, but the receive FIFO and transmit FIFO can still store characters as they normally do.

25.2.8 SIULC (0x0C00 0003)

Bit	7	6	5	4	3	2	1	0
Name	LCR7	LCR6	LCR5	LCR4	LCR3	LCR2	LCR1	LCR0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
7	LCR7	Divisor latch access bit specification 1: Divisor latch access 0: Receive buffer, transmit holding register, interrupt enable register
6	LCR6	Break control 1: Set break 0: Clear break
5	LCR5	Parity fixing 1: Fixed parity 0: Parity not fixed
4	LCR4	Parity setting 1: Set one bit as even bit 0: Set one bit as odd bit
3	LCR3	Parity enable 1: Create parity (during transmission) or check parity (during reception) 0: No parity (during transmission) or no checking (during reception)
2	LCR2	Stop bit specification 1: 1.5 bits (character length is 5 bits) 2 bits (character length is 6, 7, or 8 bits) 0: 1 bit
1:0	LCR(1:0)	Specifies the length of one character (number of bits) 11: 8 bits 10: 7 bits 01: 6 bits 00: 5 bits

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 (TxD) is forcibly set to the spacing (0) state.

The setting of bit 5 becomes valid according to settings in bits 4 and 3.

25.2.9 SIUMC (0x0C00 0004)

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	MCR4	MCR3	MCR2	MCR1	MCR0
R/W	R	R	R	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
7:5	RFU	Write 0 to these bits. 0 is returned after read.
4	MCR4	For diagnostic testing (local loopback) 1: Enable use of local loopback 0: Disable use of local loopback
3	MCR3	OUT2 signal (internal) specification 1: Output the low-level signal from the OUT2 pin 0: Output the high-level signal from the OUT2 pin
2	MCR2	OUT1 signal (internal) specification 1: Output the low-level signal from the OUT1 pin 0: Output the high-level signal from the OUT1 pin
1	MCR1	RTS# output control 1: Output the low-level signal from the RTS pin 0: Output the high-level signal from the RTS pin
0	MCR0	DTR# output control 1: Output the low-level signal from the DTR pin 0: Output the high-level signal from the DTR pin

This register is used for interface control with a modem or data set (or a peripheral device that emulates a modem).

The settings of bit 3 and bit 2 become valid only when bit 4 is set to 1 (enable use of local loopback).

- **Local Loopback**

  The local loopback can be used to test the transmit/receive data path in SIU.

  The following operation (local loopback) is executed when bit 4 value = 1.

  The transmit block's serial output (TxD) enters the marking state (logical 1) and the serial input (RxD) to the receive block is cut off. The transmit shift register's output is looped back to the receive shift register's input.

  The four modem control inputs (DSR#, CTS#, RI (internal), and DCD#) are cut off and the four modem control outputs (DTR#, RTS#, OUT1 (internal), and OUT2 (internal)) are internally connected to the corresponding modem control inputs.

  The modem control output pins are forcibly set as inactive (high level). During this kind of loopback mode, transmitted data can be immediately and directly received.

  This function can be used to check on the transmit/receive data bus within SIU.

  When in loopback mode, both transmission and receive interrupts can be used. The interrupt sources are external sources in relation to the transmit and receive blocks.

  Although modem control interrupts can be used, the low-order four bits of the modem control register can be used instead of the four modem control inputs as interrupt sources.

  As usual, each interrupt is controlled by an interrupt enable register.

25.2.10 SIULS (0x0C00 0005)

Bit	7	6	5	4	3	2	1	0
Name	LSR7	LSR6	LSR5	LSR4	LSR3	LSR2	LSR1	LSR0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	1	1	0	0	0	0	0
After reset	0	1	1	0	0	0	0	0

Bit	Name	Function
7	LSR7	Error detection in (FIFO mode) 1: Parity error, framing error, or break is detected in the FIFO. 0: Normal
6	LSR6	Transmit block empty 1: No data in transmit holding register or transmit shift register No data in transmit FIFO (during FIFO mode) 0: Data exists in transmit holding register or transmit shift register Data exists in transmit FIFO (during FIFO mode)
5	LSR5	Transmit holding register empty 1: Character is transferred to transmit shift register (during 16,550 mode) Transmit FIFO is empty (during FIFO mode) 0: Character is stored in transmit holding register (during 16,550 mode) Transmit data exists in transmit FIFO (during FIFO mode)
4	LSR4	Break interrupt 1: Break interrupt detected 0: Normal
3	LSR3	Framing error 1: Framing error detected 0: Normal
2	LSR2	Parity error 1: Parity error detected 0: Normal
1	LSR1	Overrun error 1: Overwrite receive data 0: Normal
0	LSR0	Receive data ready 1: Receive data exists in FIFO 0: No receive data in FIFO

The CPU uses this register to get information related to data transfers.
When LSR7 and LSR(4:1) are 1, reading this register clears these bits to 0.

Caution The receive data ready bit (bit 0) is set before the serial data reception is completed. Therefore, the receive data ready bit may not be cleared if the serial receive data is read from the SIURB register immediately after this bit is set.
When reading data from the SIURB register, wait for the stop bit width time since the receive data ready bit is set.

LSR7 bit is valid only in FIFO mode, and it indicates always 0 in 16,550 mode.

The value of LSR4 becomes 1 when the spacing mode (logical 0) is held longer than the time required for transmission of one word of receive data input (start bit + data bits + parity bit + stop bit).

When in FIFO mode, if a break interrupt is detected for one character in the FIFO, the character is regarded as an error character and the CPU is notified of a break interrupt when that character reaches the highest position in the FIFO.

When a break occurs, one "zero" character is sent to the FIFO. The RxD enters marking mode, and when the next valid start bit is received, the next character can be transmitted.

The value of LSR3 becomes 1 when a zero (spacing level) stop bit is detected (framing error) following the final data bit or parity bit. This bit value returns to 0 when the CPU reads the contents of the line status register.

When in FIFO mode, if a framing error is detected for one character in the FIFO, the character is regarded as an error character and the CPU is notified of a framing error when that character reaches the highest position in the FIFO.

When a framing error occurs, the SIU prepares for further synchronization. The next start bit is assumed to be the cause of the framing error and further data is not accepted until the next start bit has been sampled twice.

The value of LSR2 becomes 1 when a parity error is detected.

When in FIFO mode, if a parity error is detected for one character within the FIFO, the character is regarded as an error character and the CPU is notified of a parity error when that character reaches the highest position in the FIFO.

The value of LSR1 becomes 1 when the next character is transmitted to the receive buffer register before the CPU reads this register and the previous character disappears (overrun).

When in FIFO mode, if the data exceeds the trigger level as it continues to be transferred to the FIFO, even after the FIFO becomes full an overrun error will not occur until all characters are stored in the shift register.

The CPU is notified as soon as an overrun error occurs. The characters in the shift register are overwritten and are not transferred to the FIFO.

25.2.11 SIUMS (0x0C00 0006)

Bit	7	6	5	4	3	2	1	0
Name	MSR7	MSR6	MSR5	MSR4	MSR3	MSR2	MSR1	MSR0
R/W	R	R	R	R	R/W	R/W	R/W	R/W
RTCRST	Undefined	Undefined	Undefined	Undefined	0	0	0	0
After reset	Undefined	Undefined	Undefined	Undefined	0	0	0	0

Bit	Name	Function
7	MSR7	Complement of DCD# signal 1: High level 0: Low level
6	MSR6	Complement of RI signal (internal) 1: High level 0: Low level
5	MSR5	Complement of DSR# input 1: High level 0: Low level
4	MSR4	Complement of CTS# input 1: High level 0: Low level
3	MSR3	DCD# signal change 1: Change in DCD# signal 0: No change
2	MSR2	RI signal (internal) change 1: Change in RI signal (internal) 0: No change
1	MSR1	DSR# signal change 1: Change in DSR# signal 0: No change
0	MSR0	CTS# signal change 1: Change in CTS# signal 0: No change

This register indicates the current status of various control signals that are input to the CPU from a modem or other peripheral device.

MSR(3:0) bits are cleared to 0 when they are read.

25.2.12 SIUSC (0x0C00 0007)

Bit	7	6	5	4	3	2	1	0
Name	SCR7	SCR6	SCR5	SCR4	SCR3	SCR2	SCR1	SCR0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
7:0	SCR(7:0)	General-purpose data

This register is a readable/writable 8-bit register, and can be used freely by users.
It does not affect control of the SIU.

25.2.13 SIUIRSEL (0x0C00 0008)

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	TMICMODE	TMICTX	IRMSEL1	IRMSEL0	IRUSESEL	SIRSEL
R/W	R	R	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

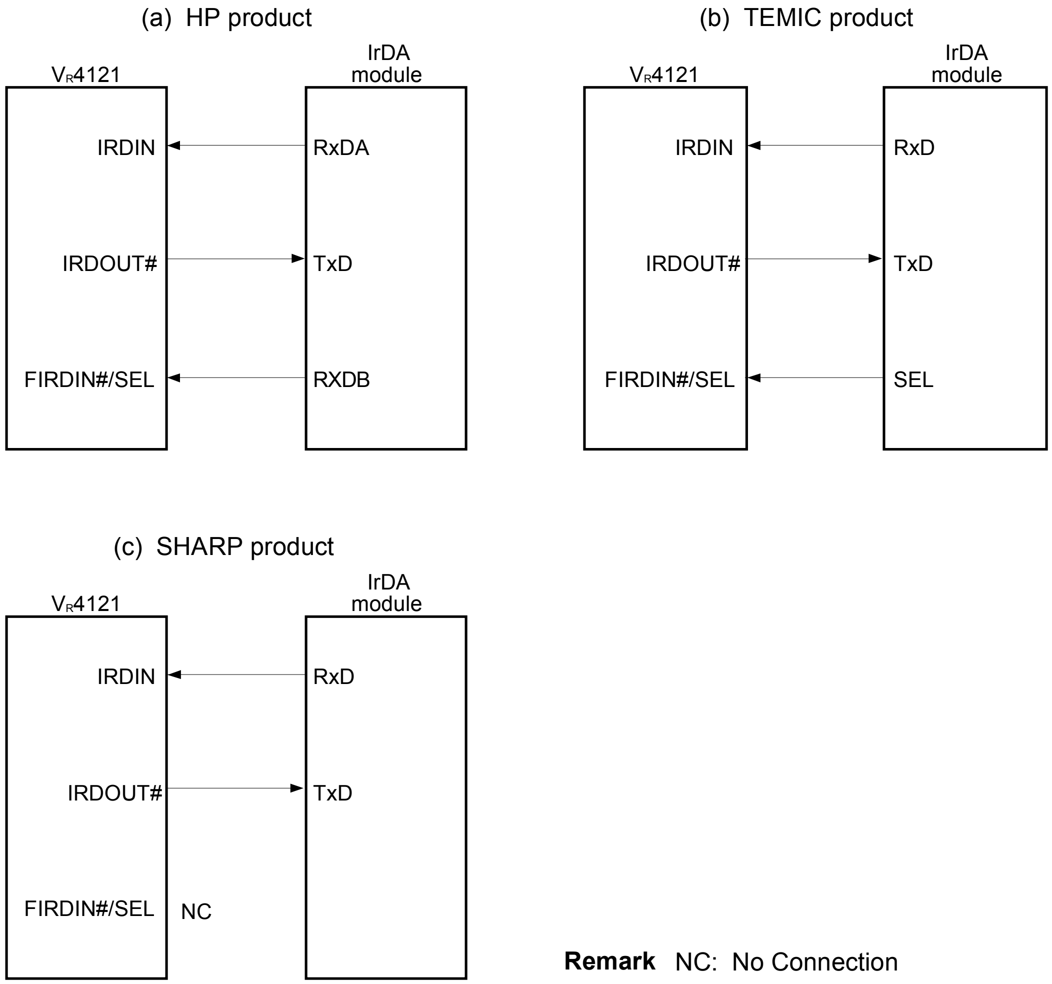
Bit	Name	Function
7:6	RFU	Write 0 to these bits. 0 is returned after read.
5	TMICMODE	Mode setting of the emitter/receptor module.
4	TMICTX	Sets communication rate. 1: Communication at 4 Mbps 0: Communication at 1.15 Mbps or less
3:2	IRMSEL(1:0)	Sets the type of emitter/receptor module to be used 11: RFU 10: HP model (HSDL-1100 is assumed) 01: TEMIC model (TFDS6000 is assumed) 00: SHARP model (RY5FD01D is assumed)
1	IRUSESEL	Selects SIU or FIR for use with IrDA emitter/receptor module 1: FIR uses IrDA module 0: SIU uses IrDA module
0	SIRSEL	Selects whether the SIU uses the IrDA module or the RS-232-C pins during communications 1: Use IrDA module 0: Use RS-232-C interface

This register is used to set the IrDA module settings, IrDA module access privileges, and the SIU's communication format (IrDA or serial).

The settings of the TMICMODE and TMICTX bits are valid only when the IRMSEL(1:0) bit is set to 01 (TEMIC model).

Connection examples of the VR4121 and IrDA modules are shown below.

Figure 25-1. Connection Example Between the V$_R$4121 and IrDA Module



(a) HP product

(b) TEMIC product

(c) SHARP product

Remark NC: No Connection

25.2.14 SIURESET (0x0C00 0009)

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	SIU RESET
R/W	R	R	R	R	R	R	R	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
7:1	RFU	Write 0 to these bits. 0 is returned after a read.
0	SIURESET	This bit is used to reset SIU. 1: Reset SIU 0: Release SIU reset

This register is used to reset SIU forcibly.

25.2.15 SIUCSEL (0x0C00 000A)

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	CSEL
R/W	R	R	R	R	R	R	R	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
7:1	RFU	Write 0 to these bits. 0 is returned after a read.
0	CSEL	This bit is used to specify masking for echo-back prevention. 1: Mask 0: Do not mask

This register is used to specify whether masking is done for echo-back prevention when using SIR.

CHAPTER 26 HSP (MODEM INTERFACE UNIT)

26.1 General

The core of the HSP unit uses an NEC56K chip made by PCTEL. The main functions of this core are as follows:

- CODEC device control and serial ↔ parallel conversion of the CODEC transmit/receive data
- Control of relay lines, hook lines, and other signal lines in the data access arrangement (DAA) block

Block diagrams of HSP unit and an example of connection between the VR4121 and external agents are shown below.

Figure 26-1. HSP Unit Block Diagram

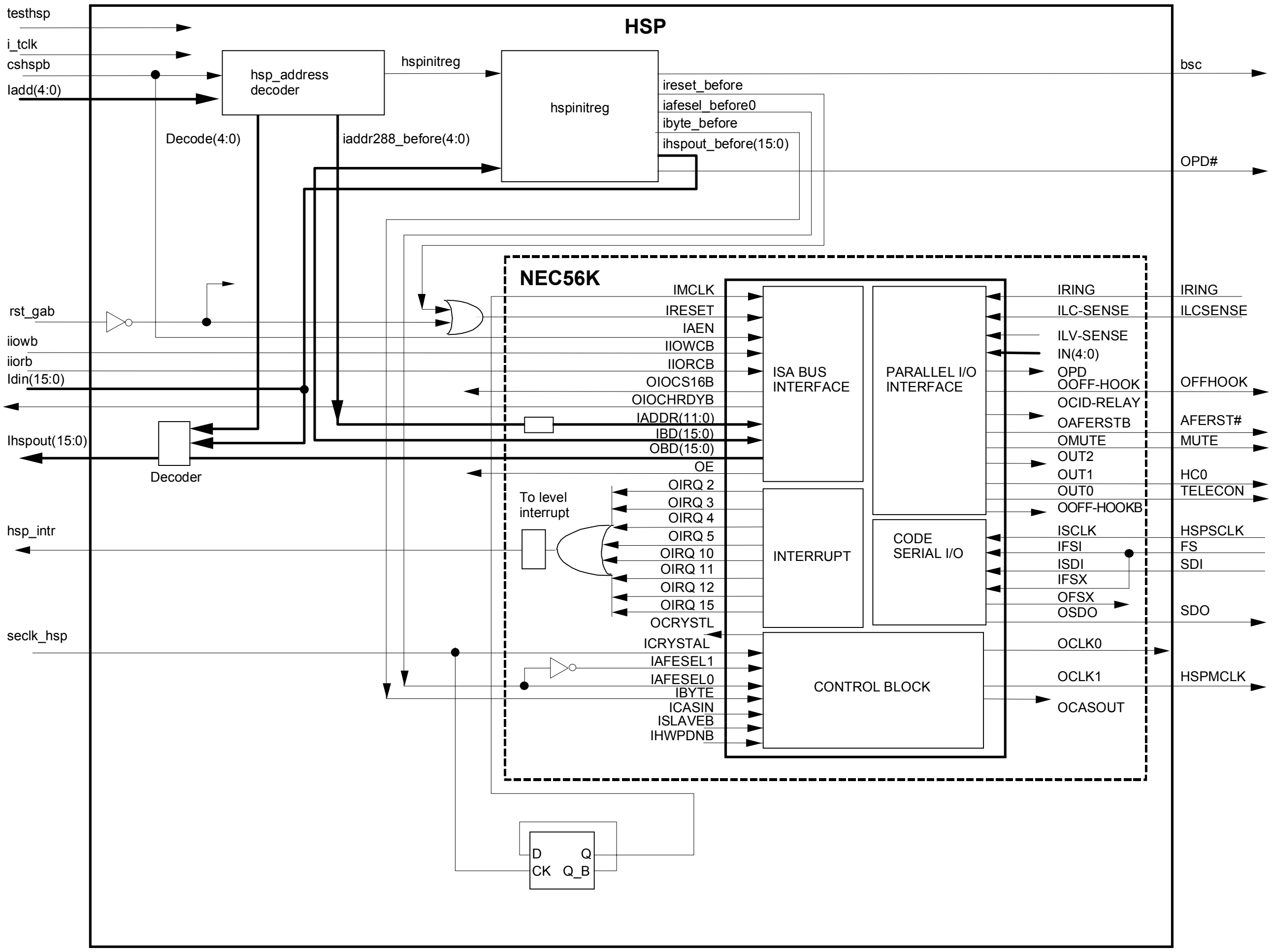
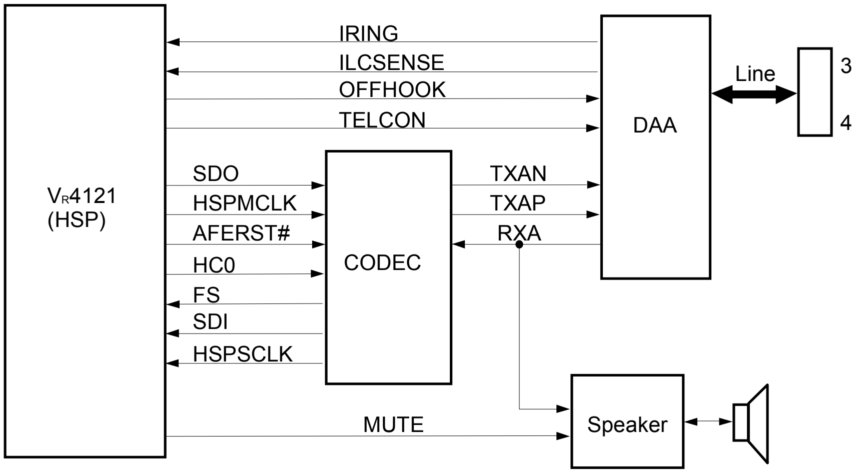


Figure 26-2. Circuit Configuration Block Diagram Examples



26.2 Register Set

The HSP registers are listed below.

The data registers can be accessed as the control registers by specifying the INDEX number and then reading from or writing to.

The HSPINIT register is added for the VR4121, and other registers are original to the NEC56K.

Table 26-1. HSP Registers

Address	R/W	Register Symbols	Name
0x0C00 0020	R/W	HSPINIT	HSP Initialize Register
0x0C00 0022	R/W	HSPDATA(7:0)	HSP Data Register L
0x0C00 0023	R/W	HSPDATA(15:8)	HSP Data Register H
0x0C00 0024	W	HSPINDEX	HSP Index Register
0x0C00 0028	R	HSPID(7:0)	HSP ID Register
0x0C00 0029	R	HSPPCS(7:0)	HSP I/O Address Program Confirmation Register
0x0C00 0029	W	HSPPCTEL(7:0)	HSP Signature Checking Port

These registers are described in detail below.

26.2.1 HSP initialize register

(1) HSPINIT (0x0C00 0020)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	OPD	AFESEL	BYTE	BSC	HSPRST
R/W	R	R	R	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:5	RFU	Write 0 to these bits. 0 is returned after read.
4	OPD	Power-down CODEC (indicates OPD# pin's state) 1: High level 0: Low level
3	AFESEL	CODEC interface mode switch 1: ST7546, STLC7546(SGS), T7525(AT) 0: TLC320C44, TLC320AC01/02(TI)
2	BYTE	HSP data bus width setting 1: 8 bits 0: 16 bits
1	BSC	CODEC interface control 1: Normal 0: Bus is high-impedance, or reset status
0	HSPRST	HSP reset 1: Reset 0: Do not reset

This register is used to control the HSP.

BSC is used to control the CODEC interface. This bit must be set to 1 when using the HSP.

The reset by HSPRST is not effective on the HSPINIT register settings. The HSPRST bit does not automatically become 0 after it is set to 1. Therefore, be sure to set 0 to this bit to release the reset.

26.2.2 HSP data register, HSP index register

HSPDATA(15:0) is a 16-bit data port. This register can be accessed as control registers according to the HSPINDEX(15:0) setting.

HSPINDEX(15:0) is a write-only index register. The function of the data register changes according to the values set to this register.

The correspondence between INDEX numbers and registers is shown below.

Table 26-2. Control Register Definitions

INDEX	WRITE		READ	
	Higher Byte	Lower Byte	Higher Byte	Lower Byte
0	HSPTxData(15:8)	HSPTxData(7:0)	HSPRxData(15:8)	HSPRxData(7:0)
1	HSPCNTL(9:8)	HSPCNTL(7:0)	HSPSTS(15:8)	HSPSTS(7:0)
2	RFU	HSPEXTOUT(7:0)	HSPID(7:0)	HSPEXTIN(7:0)
3	HSPTOC(3:0)	HSPMCLK(4:0)	HSPERRCNT(11:8)	HSPERRCNT(7:0)
4	RFU	HSPFFSZ(7:0)	RFU	
5 to 15	RFU		RFU	
16 to 255	Not Accessible		Not Accessible	

The control registers are described below.

(1) HSPTxData (0x0C00 0022: Index 0, Write)

Bit	15	14	13	12	11	10	9	8
Name	TxData15	TxData14	TxData13	TxData12	TxData11	TxData10	TxData9	TxData8
R/W	W	W	W	W	W	W	W	W
RTCRST	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined
After reset	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined

Bit	7	6	5	4	3	2	1	0
Name	TxData7	TxData6	TxData5	TxData4	TxData3	TxData2	TxData1	TxData0
R/W	W	W	W	W	W	W	W	W
RTCRST	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined
After reset	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined

Bit	Name	Function
15:0	TxData(15:0)	Transmit data

This register is used to store transmission data when the index number is 0.

(2) HSPCNTL (0x0C00 0022: Index 1, Write)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	W	W	W	W	W	W	W	W
RTCRST	0	0	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined
After reset	0	0	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined

Bit	7	6	5	4	3	2	1	0
Name	NTORST	ENIRQ	START	RFU	ENTX	IRQS2	IRQS1	IRQS0
R/W	W	W	W	W	W	W	W	W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:8	RFU	Write 0 to these bits.
7	NTORST	Disable timeout reset 1: Disable 0: Enable
6	ENIRQ	Interrupt enable 1: Enable 0: Disable
5	START	RX/TX FIFO pointer initialization 1: Initialize (at rising edge) 0: Status hold
4	RFU	Write 0 to this bit.
3	ENTX	Transfer enable 1: Enable 0: Disable
2:0	IRQS(2:0)	Interrupt signal select. However, IRQ signal is always selected whatever value is set to these bits (000 through 111).

This register is used to set several settings to control HSP when the index number is 1.

When the NTORST bit is "0", this enables a timeout to occur when a specified number of errors has been counted, at which point HSP resets itself.

When the START bit is set to "1", the RX/TXFIFO pointer is set to the initial value.

Caution If 1 is set to ENTX bit, the setting cannot be changed after that. The only way to stop the operation is by resetting the NEC56K.

(3) HSPEXTOUT (0x0C00 0022: Index 2, Write)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	W	W	W	W	W	W	W	W
RTCRST	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined
After reset	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined

Bit	7	6	5	4	3	2	1	0
Name	RFU	HC0	TELECON	RFU	MUTE	AFERST	RFU	OFFHOOK
R/W	W	W	W	W	W	W	W	W
RTCRST	Undefined	0	0	0	0	1	0	0
After reset	Undefined	0	0	0	0	1	0	0

Bit	Name	Function
15:7	RFU	Write 0 to these bits.
6	HC0	Select CODEC mode This bit is connected to the HC0 pin. 1: High-level signal output 0: Low-level signal output
5	TELECON	Hand set relay control This bit is connected to the TELECON pin. 1: High-level signal output 0: Low-level signal output
4	RFU	Write 0 to this bit.
3	MUTE	Mute speaker This bit is connected to the MUTE pin. 1: High-level signal output 0: Low-level signal output
2	AFERST	CODEC reset This bit is connected to the AFERST# pin. 1: High-level signal output 0: Low-level signal output
1	RFU	Write 0 to this bit.
0	OFFHOOK	OFF HOOK relay control This bit is connected to the OFFHOOK pin. 1: High-level signal output 0: Low-level signal output

This register is used to set output values of the HSP signals when the index number is 2.

(4) HSPTOC and HSPMCLKD (0x0C00 0022: Index 3, Write)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	TOC3	TOC2	TOC1	TOC0
R/W	W	W	W	W	W	W	W	W
RTCRST	Undefined	Undefined	Undefined	Undefined	0	0	0	0
After reset	Undefined	Undefined	Undefined	Undefined	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	MCLKD4	MCLKD3	MCLKD2	MCLKD1	MCLKD0
R/W	W	W	W	W	W	W	W	W
RTCRST	Undefined	Undefined	Undefined	1	1	1	1	0
After reset	Undefined	Undefined	Undefined	1	1	1	1	0

Bit	Name	Function
15:12	RFU	Write 0 to these bits.
11:8	TOC(3:0)	Higher 4 bits of timeout count
7:5	RFU	Write 0 to these bits.
4:0	MCLKD(4:0)	HSPMCLK divisor to clock input HSPMCLK frequency = 18.432 MHz / (MCLKD(4:0) + 1)

The upper byte if this register sets the timeout counter value and lower byte sets the HSPMCLK's division ratio when the INDEX number is 3.

TOC(3:0) is used to set the higher four bits of the final count of the timeout counter. The timeout counter is a 12-bit counter and is incremented once for each interrupt signal that is not serviced. The lower 8 bits are automatically set to 0 when TOC(3:0) is set. When the specified timeout count value is reached, the TO bit of HSPSTS register is set to 1. The user is responsible for resetting the HSP core to prevent a system hang-up.

MCLKD(4:0) is used to set the division ratio when the clock is supplied to the HSPMCLK pin. If MCLKD(4:0) is "0", there is no clock division and the 18.432-MHz clock is output. Note that an even number must be set to MCLKD(4:0).

(5) HSPFFSZ (0x0C00 0022: Index 4, Write)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	W	W	W	W	W	W	W	W
RTCRST	Undefined	Undefined	Undefined	0	0	0	0	0
After reset	Undefined	Undefined	Undefined	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	FFSZ7	FFSZ6	FFSZ5	FFSZ4	FFSZ3	FFSZ2	FFSZ1	FFSZ0
R/W	W	W	W	W	W	W	W	W
RTCRST	0	0	1	0	0	0	0	0
After reset	0	0	1	0	0	0	0	0

Bit	Name	Function
15:8	RFU	Write 0 to these bits.
7:0	FFSZ(7:0)	FIFO size control

When the index number is 4, this register is used to set the transmit/receive buffer size, and can be set up to 96 words (0x60) word-wise (16 bits). If buffer-full interrupt is enabled, an interrupt will occur when the data in the transmit/ receive buffer reaches to the size set in this register.

(6) HSPRxData (0x0C00 0022: Index 0, Read)

Bit	15	14	13	12	11	10	9	8
Name	RxData15	RxData14	RxData13	RxData12	RxData11	RxData10	RxData9	RxData8
R/W	R	R	R	R	R	R	R	R
RTCRST	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined
After reset	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined

Bit	7	6	5	4	3	2	1	0
Name	RxData7	RxData6	RxData5	RxData4	RxData3	RxData2	RxData1	RxData0
R/W	R	R	R	R	R	R	R	R
RTCRST	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined
After reset	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined

Bit	Name	Function
15:0	RxData(15:0)	Receive data from the receive FIFO

This register is used to store the receive data sent from the receive FIFO when the index number is 0.

(7) HSPSTS (0x0C00 0022: Index 1, Read)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	Undefined	Undefined	Undefined
After reset	0	0	0	0	0	Undefined	Undefined	Undefined

Bit	7	6	5	4	3	2	1	0
Name	AFESEL1	AFESEL0	IBYTE	TO	CFGCP	IRQS	RxOVRUN	TxUDRUN
R/W	R	R	R	R	R	R	R	R
RTCRST	Undefined	Undefined	Undefined	0	0	0	0	0
After reset	Undefined	Undefined	Undefined	0	0	0	0	0

Bit	Name	Function
15:8	RFU	Write 0 to these bits. 0 is returned after a read.
7:6	AFESEL(1:0)	Indicates the AFESEL(1:0) signal (internal) state
5	IBYTE	Indicates the BYTE signal (internal) state
4	TO	Error-related timeout 1: Timeout occurred 0: No timeout
3	CFGCP	CODEC initialization complete 1: Complete 0: Not complete
2	IRQS	Pending interrupt request exists 1: Exists 0: No pending interrupt requests
1	RxOVRUN	Receive buffer overrun occurred 1: Occurred 0: No receive overruns
0	TxUDRUN	Transmit buffer underrun occurred 1: Occurred 0: No transmit overruns

This register is used to indicate various states during a communication when the index number is 1.

TO bit is set (to "1") when the timeout counter reaches the value specified by the TOC bit of HSPTOC.

CFGCP bit indicates whether or not CODEC initialization has been completed. Actually, this bit is set (to "1") when the START bit of HSPCNT has been set as active to reset the FIFO pointer and then 9-word data has been transmitted (1 word = 16 bits).

IRQS bit indicates whether or not any pending interrupt request exists. When an interrupt request from HSP to the CPU core is in pending, the request is cleared after this register is read.

IRQS, RxOVRUN, and TxUDRUN bits are cleared (to "0") when read.

(8) HSPID and HSPEXTIN (0x0C00 0022: Index 2, Read)

Bit	15	14	13	12	11	10	9	8
Name	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	1	0	0	1	1	0
After reset	0	0	1	0	0	1	1	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	ILCS	IRING
R/W	R	R	R	R	R	R	R	R
RTCRST	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined
After reset	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined

Bit	Name	Function
15:8	ID(7:0)	Indicates HSP unit's ID and revision number
7:2	RFU	Write 0 to these bits. 0 is returned after a read.
1	ILCS	ILCSENSE input pin state indication
0	IRING	IRING input pin state indication

When the index number is 2, this register indicates the HSP unit's ID and revision number in the higher byte and the HSP input signal's state in the lower byte.

ID(7:0) is divided into two parts. The higher 4 bits of ID(7:0) indicate the ID number of HSP, and the lower 4 bits indicate the revision number of HSP.

(9) HSPERRCNT (0x0C00 0022: Index 3, Read)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	ERRCNT11	ERRCNT10	ERRCNT9	ERRCNT8
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	ERRCNT7	ERRCNT6	ERRCNT5	ERRCNT4	ERRCNT3	ERRCNT2	ERRCNT1	ERRCNT0
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:12	RFU	0 is returned after a read.
11:0	ERRCNT(11:0)	Error count

This register indicates the number of errors when the index number is 3.

This register indicates the number of overrun or underrun errors that have occurred. This is used for synchronizing software and hardware.

26.2.3 HSP ID register, HSP I/O address program confirmation register

The specific values are displayed to HSPID(7:0) and HSPPCS(7:0) registers following normal access of HSPPCTEL register.

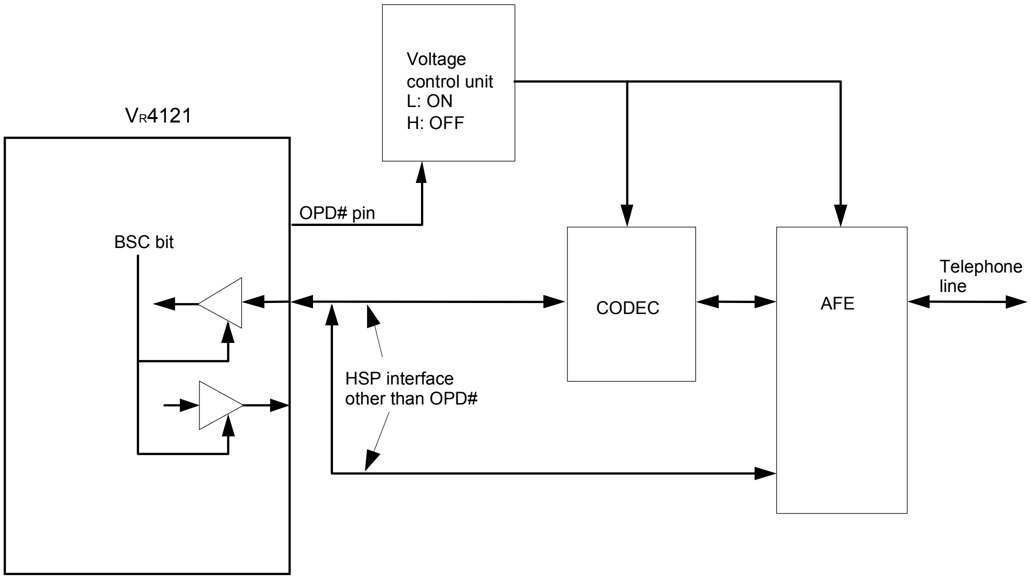
26.2.4 HSP signature checking port

HSPPCTEL(7:0) register must be accessed when to start using HSP unit. 0x5A can be read from the HSPPCS register by writing a certain value. Other HSP registers cannot be accessed unless this processing is executed. It must be executed during initialization.

26.3 Power Control

Power control of the CODEC and AFE can be performed using the OPD# pin and the BSC bit (HSPINIT). The following is an example of a control method using these units.

Figure 26-3. Block Diagram of HSP Interface Power Control



(1) After RTC reset

Item		OPD# Pin	BSC Bit	HSP Bus State	VR4121 Power	CODEC/AFE Power
1	When initialized	L	0	**Note**	ON	OFF
2	During power-on of CODEC or AFE	H	0	**Note**	ON	ON
3	When HSP bus’s gate is set to "ON"	H	1	Normal	ON	ON
4	Software modem control	H	1	Normal	ON	ON

Note Refer to **2.3.1 Pin status upon specific states.**

(2) During power-down (VR4121: Fullspeed/Standby/Suspend mode)

Item		OPD# Pin	BSC Bit	HSP Bus State	VR4121 Power	CODEC/AFE Power
1	Complete operation	H	1	Normal	ON	ON
2	When HSP bus's gate is set to "OFF"	H	0	**Note**	ON	ON
3	When CODEC or AFE power is set to "OFF"	L	0	**Note**	ON	OFF
4	If necessary, issue STANDBY/ SUSPEND command	L	0	**Note**	ON	OFF

Note Refer to **2.3.1 Pin status upon specific states.**

(3) During recovery from power-down (VR4121: Fullspeed/Standby/Suspend mode)

Item		OPD# Pin	BSC Bit	HSP Bus State	VR4121 Power	CODEC/AFE Power
1	Power down status	L	0	**Note**	ON	OFF
2	During power-on of CODEC or AFE	H	0	**Note**	ON	ON
3	When HSP bus's gate is set to "ON"	H	1	Normal	ON	ON
4	Use HSP unit	H	1	Normal	ON	ON

Note Refer to **2.3.1 Pin status upon specific states.**

(4) When changing to Hibernate mode (the following processing must occur before entering Hibernate mode)

Item		OPD# Pin	BSC Bit	HSP Bus State	VR4121 Power	CODEC/AFE Power
1	Complete operation	H	1	Normal	ON	ON
2	When HSP bus's gate is set to "OFF"	H	0	**Note**	ON	ON
3	When CODEC or AFE power is set to "OFF"	L	0	**Note**	ON	OFF
4	Issue HIBERNATE command	L	0	**Note**	ON	OFF

Note Refer to **2.3.1 Pin status upon specific states.**

(5) During recovery from Hibernate mode to use HSP unit

Item		OPD# Pin	BSC Bit	HSP Bus State	VR4121 Power	CODEC/AFE Power
1	During Hibernate mode	L	0	**Note**	ON	OFF
2	During power-on of CODEC or AFE	H	0	**Note**	ON	ON
3	When HSP bus's gate is set to "ON"	H	1	Normal	ON	ON
4	Use HSP unit	H	1	Normal	ON	ON

Note Refer to **2.3.1 Pin status upon specific states.**

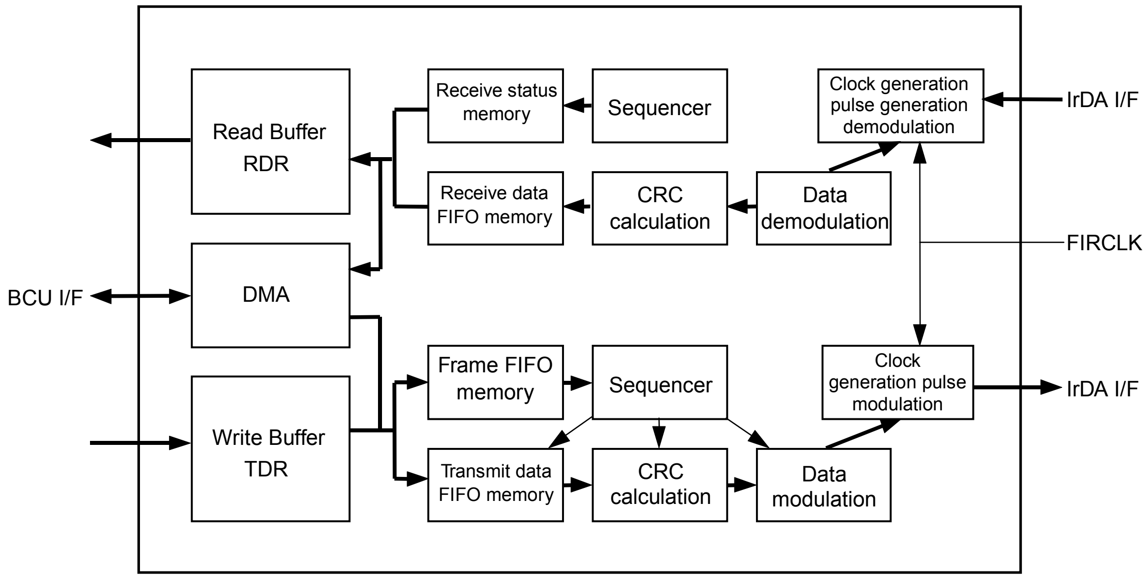
[MEMO]

CHAPTER 27 FIR (FAST IrDA INTERFACE UNIT)

27.1 General

This unit supports the IrDA 1.1 high-speed infrared communication physical layer standard.
Supported FIR (Fast SIR) transfer rates include 0.576 Mbps, 1.152 Mbps, and 4 Mbps.
SIR (up to 1.152 kbps) is not supported.

★ **Figure 27-1. Internal Block Diagram**



27.2 Register Set

The FIR registers are listed below.

Table 27-1. FIR Registers

Address	R/W	Register Symbols	Function
0x0C00 0040	R/W	FRSTR	FIR Reset register
0x0C00 0042	R/W	DPINTR	DMA Page Interrupt register
0x0C00 0044	R/W	DPCNTR	DMA Page Control register
0x0C00 0050	W	TDR	Transmit Data register
0x0C00 0052	R	RDR	Receive Data register
0x0C00 0054	R/W	IMR	Interrupt Mask register
0x0C00 0056	R/W	FSR	FIFO Setup register
0x0C00 0058	R/W	IRSR1	Infrared Setup register 1
0x0C00 005C	R/W	CRCSR	CRC Setup register
0x0C00 005E	R/W	FIRCR	FIR Control register
0x0C00 0060	R/W	MIRCR	MIR Control register
0x0C00 0062	R/W	DMACR	DMA Control register
0x0C00 0064	R/W	DMAER	DMA Enable register
0x0C00 0066	R	TXIR	Transmit Indication register
0x0C00 0068	R	RXIR	Receive Indication register
0x0C00 006A	R	IFR	Interrupt Flag register
0x0C00 006C	R	RXSTS	Receive Status
0x0C00 006E	R/W	TXFL	Transmit Frame Length
0x0C00 0070	R/W	MRXF	Maximum Receive Frame Length
0x0C00 0074	R	RXFL	Receive Frame Length

These registers are described in detail below.

27.2.1 FRSTR (0x0C00 0040)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	FRST
R/W	R	R	R	R	R	R	R	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:1	RFU	Write 0 to these bits. 0 is returned after a read.
0	FRST	FIR reset. Write 0 when releasing reset. 1: Reset 0: Normal

27.2.2 DPINTR (0x0C00 0042)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	FDPINT5	FDPINT4	FDPINT3	FDPINT2	FDPINT1
R/W	R	R	R	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:5	RFU	Write 0 to these bits. 0 is returned after a read.
4	FDPINT5	This bit indicates an FIR interrupt occurs. 1: Occurred 0: Normal
3	FDPINT4	This bit indicates that the DMA buffer (receive side) becomes full (2 pages). Cleared to 0 when 1 is written. 1: Occurred 0: Normal
2	FDPINT3	This bit indicates that the DMA buffer (transmit side) becomes full (2 pages). Cleared to 0 when 1 is written. 1: Occurred 0: Normal
1	FDPINT2	This bit indicates that the DMA buffer (receive side) becomes full (1 page). Cleared to 0 when 1 is written. 1: Occurred 0: Normal
0	FDPINT1	This bit indicates that the DMA buffer (transmit side) becomes full (1 page). Cleared to 0 when 1 is written. 1: Occurred 0: Normal

This register is used to generates DMA page interrupts of FIR.

FDPINT4 bit indicates that the DMA buffer (receive side) becomes full (2 pages). DMARQ is stopped when this bit is set to 1.

FDPINT3 bit indicates that the DMA buffer (transmit side) becomes full (2 pages). DMARQ is stopped when this bit is set to 1.

FDPINT2 bit indicates that the DMA buffer (receive side) becomes full (1 page). DMARQ is stopped when this bit is set to 1 and bit 0 of DPCNTR is 1.

FDPINT1 bit indicates that the DAM buffer (transmit side) becomes full (1 page). DMARQ is stopped when this bit is set to 1 and bit 0 of DPCNTR is 1.

Cautions 1. If FDPINT(4:3) is set to 1, the last data of transmit data is not guaranteed.

2. If FDPINT(2:1) is set to 1 while bit 0 of DPCNTR register is set to 1, the last data of transfer data is not guaranteed.

27.2.3 DPCNTR (0x0C00 0044)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	FDPCNT
R/W	R	R	R	R	R	R	R	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:1	RFU	Write 0 to these bits. 0 is returned after a read.
0	FDPCNT	DMA transfer stopping boundary. 1: 1-page boundary 0: 2-page boundary

Cautions 1. When the FDPCNT bit is set to 1, the last data of the first page is not guaranteed.

2. When the FDPCNT bit is set to 0, the last data of the second page is not guaranteed.

27.2.4 TDR (0x0C00 0050)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	W	W	W	W	W	W	W	W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	TDR7	TDR6	TDR5	TDR4	TDR3	TDR2	TDR1	TDR0
R/W	W	W	W	W	W	W	W	W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:8	RFU	Write 0 to these bits. 0 is returned after a read.
7:0	TDR(7:0)	Transmit data FIFO

This register is used to store the address to which data is written to the transmit data store FIFO.

Up to 64- or 32-byte data (determined by bit 3 of FSR register) is stored to the transmit data store FIFO.

The transmit data store FIFO is used as follows.

(1) Write

Data is written to the transmit data store FIFO while the IrDA is operating.

When a write operation is completed, the write pointer of the transmit data store FIFO is incremented. However, if data is written when this write pointer is full, it is not incremented.

After the data of frame size is written to the TXFL register in a status other than the transmit busy status (start enable), if the data written to this register reaches frame size, data transfer starts even if the number of write to this register is short of the threshold. (This is Start 1.)

After that, data is always transferred if it reaches frame size, even if it is short of the threshold. (This is Start 2.)

(2) Read

After frame transmission is completed, the sequencer reads the transmit data during the data transfer sequence, and the read pointer is incremented.

If read is done while the transmit FIFO is empty, a transmit underrun error occurs. This stops the current frame transmission and then starts the abort frame transmission. The following frames scheduled to be transmitted next are not transferred.

27.2.5 RDR (0x0C00 0052)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RDR7	RDR6	RDR5	RDR4	RDR3	RDR2	RDR1	RDR0
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:8	RFU	Write 0 to these bits. 0 is returned after a read.
7:0	RDR(7:0)	Receive data FIFO

This register is used to store the address from which data is read from the receive data store FIFO.

Up to 64- or 32-byte data (determined by bit 3 of FSR) is stored to the receive data store FIFO.

The receive data store FIFO is used as follows.

(1) Write

During a frame data reception, the sequencer writes the receive data during the data transfer sequence, and the write pointer is incremented.

If data is written when the unread data in the receive FIFO reaches the maximum volume, the receive overrun error occurs and the current frame reception is ended.

The write pointer is not incremented.

After the receive FIFO is cleared, if the number of received frames is less than 7 frames, it is possible to continue frame reception.

To receive 8 or more frames, read all the data and frames that are already received from the receive FIFO, then clear the receive FIFO and restart reception.

(2) Read

Data is read from the receive data store FIFO while the IrDA is operating.

When a read operation is completed, the read pointer of the receive data store FIFO is incremented. However, it is not incremented when the receive FIFO is empty.

When the number of read frames reaches the receive frame size, an interrupt occurs and bit 7 of the RXSTS register is set.

Cautions 1. If data is read when the receive FIFO is empty (read pointer = write pointer), it may contend with the sequencer's write operation. This may cause undefined data.

2. The error generated by read underrun is not reported in the V_R4121.

27.2.6 IMR (0x0C00 0054)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	IMR7	IMR6	IMR5	IMR4	IMR3	IMR2	IMR1	IMR0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:8	RFU	Write 0 to these bits. 0 is returned after a read.
7:0	IMR(7:0)	These bits are used to enable/prohibit interrupt output. 1: Enable 0: Disable

This register sets whether to notify to external when interrupt occurs. Each bit corresponds to the equivalent IFR register bit. When interrupt output is enabled and the corresponding bit is 1, interrupt output becomes active.

Caution The IFR register is set irrespective of this register's setting.

27.2.7 FSR (0x0C00 0056)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RX_TH1	RX_TH0	TX_TH1	TX_TH0	F_SIZE	TXF_CLR	RXF_CLR	TX_STOP
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:8	RFU	Write 0 to these bits. 0 is returned after a read.
7:6	RX_TH(1:0)	These bits are used to specify the receive FIFO's threshold. RX_TH 1, 0 — F_SIZE = 0 — F_SIZE = 1 11 — 26 bytes — 48 bytes 10 — 16 bytes — 32 bytes 01 — 4 bytes — 8 bytes 00 — 1 byte — 1 byte
5:4	TX_TH(1:0)	These bits are used to specify the transmit FIFO's threshold. TX_TH 1, 0 — F_SIZE = 0 — F_SIZE = 1 11 — 26 bytes — 48 bytes 10 — 16 bytes — 32 bytes 01 — 8 bytes — 16 bytes 00 — 1 byte — 1 byte
3	F_SIZE	This bit is used to specify the maximum size of transmit/receive FIFO. 1: 64 bytes 0: 32 bytes
2	TXF_CLR	Transmit FIFO clear trigger (read value = 0)
1	RXF_CLR	Receive FIFO clear trigger (read value = 0)
0	TX_STOP	Transmission stop trigger (read value = 0)

This register is used to make various settings for the transmit/receive FIFO.

When the TXF_CLR bit is set, the pointers of the transmit data FIFO and transmit frame size FIFO are initialized.

When the RXF_CLR bit is set, the pointers of the receive data FIFO, receive frame size FIFO, and receive status FIFO are initialized.

When the TX_STOP bit is set, the current frame transmission is stopped and the abort frame transmission starts.

The following frames scheduled to be transmitted next are not transferred. Also, setting this bit to 1 causes DMA operation to be stopped, and then a DMA completion interrupt to be output.

Cautions 1. During transmission/reception, the contents of bits 7 through 3 of the FSR register must not be changed (refresh is possible).

2. The data in the FIFO is not cleared even if the TXF_CLR or RXF_CLR bit is set (clearing the pointer).

3. Set the TX_STOP bit to stop DMA operation after data transfer is completed, regardless of whether the DMA operation is transmission or reception. In the case of reception, however, the DMA should be stopped after confirming the command bit of the transferred data.

27.2.8 IRSR1 (0x0C00 0058)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	IRDA_EN	RFU	RFU	RFU	RFU	RFU	IRDA_MD	MIR_MD
R/W	R/W	R	R	R	R	R	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

<table>
<tr><th>Bit</th><th>Name</th><th colspan="5">Function</th></tr>
<tr><td>15:8</td><td>RFU</td><td colspan="5">Write 0 to these bits. 0 is returned after a read.</td></tr>
<tr><td>7</td><td>IRDA_EN</td><td colspan="5">This bit is used to control (enable/prohibit) FIR operation.
1: Enable
0: Prohibit</td></tr>
<tr><td>6:2</td><td>RFU</td><td colspan="5">Write 0 to these bits. 0 is returned after a read.</td></tr>
<tr><td rowspan="5">1:0</td><td rowspan="5">IRDA_MD/
MIR_MD</td><td colspan="5">These bits are used to specify the IrDA/MIR mode.</td></tr>
<tr><td>IRDA_MD</td><td>MIR_MD</td><td>Operation mode</td><td>Frequency</td><td>Modulation method</td></tr>
<tr><td>1</td><td>1</td><td>MIR half mode</td><td>0.576 MHz</td><td>Bit stream/stuff</td></tr>
<tr><td>1</td><td>0</td><td>MIR full mode</td><td>1.152MHz</td><td>Bit stream/stuff</td></tr>
<tr><td>0</td><td>1 or 0</td><td>FIR mode</td><td>8 MHz</td><td>4PPM</td></tr>
</table>

When the IRDA_EN bit is set, the peripheral main block's reset is released and clock supply starts.

Pulse output level changes according to operation mode changes by IRDA_MD and MIR_MD.

The operation mode should be changed after changing the IrDA operation to prohibit state (by setting bit 7 = 0). The output level does not change because output latch is reset.

Once the mode is changed, be sure to switch bit inversion of I/O data ON/OFF by setting bit 0 of the CRCSR register.

Example) Sequence of changing operation mode from FIR mode to MIR full mode

```
clr1    0x7, IRSR1     Prohibit IrDA operation
set1    0x1, IRSR1     Change the mode
set1    0x0, CRCSR     Set bit inversion
set1    0x7, IRSR1     Enable IrDA operation
```

Caution During transmission/reception, this register must not be changed (refresh is possible).

27.2.9 CRCSR (0x0C00 005C)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	TX_EN	RX_EN	4PPM_DIS	DPLL_DIS	RFU	NON_CRC	CRC_INV	DATA_INV
R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:8	RFU	Write 0 to these bits. 0 is returned after a read.
7	TX_EN	This bit is used to control (enable/prohibit) masking of transmit start enable flag. 1: Enable 0: Prohibit
6	RX_EN	This bit is used to control (enable/prohibit) receive operation. 1: Enable 0: Prohibit
5	4PPM_DIS	This bit is used to control (enable/prohibit) the 4PPM modulation (for debugging). 1: Prohibit 0: Enable
4	DPLL_DIS	This bit is used to control (enable/prohibit) the bit correction (for debugging). 1: Prohibit 0: Enable
3	RFU	Write 0 to this bit. 0 is returned after a read.
2	NON_CRC	This bit is used to control whether or not a CRC is added (for debugging). 1: Do not add CRC 0: Add CRC
1	CRC_INV	This bit is used to set whether or not a CRC is inverted (for debugging). 1: Inverted CRC 0: Normal CRC (not inverted)
0	DATA_INV	This bit is used to set whether or not received/transmitted data I/O is inverted. 1: Inverted 0: Normal (not inverted)

TX_EN bit is used to set whether to mask sequence transition to transmission enable state entered by writing the TXFL register.

RX_EN bit is used to set whether to release masking of receive line, sampling data, and generating of receive clocks.

4PPM_DIS bit is used to set whether to disable 4PPM modulation of transmit data.

DPLL_DIS bit is used to set whether to disable bit correction of receive data.

NON_CRC bit is used to set whether to transmit with STO after the data transmit of transmit frame.

CRC_INV bit is used to set whether to invert CRC to create an incorrect CRC in the normal routine.

DATA_INV bit is used to set whether to invert receive/transmitted data I/O. Be sure to set as 0 (normal) in FIR and 1 (invert) in MIR.

Caution During transmission/reception, the data in this register must not be changed (refresh is possible).

27.2.10 FIRCR (0x0C00 005E)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	PA_LEN2	PA_LEN1	PA_LEN0	W_PULSE1	W_PULSE0	F_WIDTH2	F_WIDTH1	F_WIDTH0
R/W	R/W	R/W	R/W	R	R	R/W	R/W	R/W
RTCRST	1	0	0	0	0	1	0	1
After reset	1	0	0	0	0	1	0	1

Bit	Name	Function
15:8	RFU	Write 0 to these bits. 0 is returned after a read.
7:5	PA_LEN(2:0)	These bits are used to specify the number of PA (preamble). 111: 32 100: 16 (default) 011: 4 010: 2 001: 1 Others: RFU
4:3	W_PULSE(1:0)	These bits are used to specify the undefined receive pulse width area. 11: 10 to 11 clocks 10: 9 to 10 clocks 01: 8 to 9 clocks 00: 7 to 8 clocks
2:0	F_WIDTH(2:0)	These bits are used to specify the FIR pulse modulation width. F_WIDTH(2:0) / Single pulse / Double pulse 101 (default) / 6 clocks / 12 clocks 100 / 5 clocks / 11 clocks 011 / 4 clocks / 10 clocks 010 / 3 clocks / 9 clocks 001 / 2 clocks / 8 clocks 000 / 1 clock / 7 clocks Others / Setting prohibited / Setting prohibited

This register is used to control FIR operations. The PA_LEN bits are used to specify the number of PA added to the FIR transmit frame.

The W_PULSE bits are used to specify the undefined receive pulse width area. A pulse width within the undefined receive pulse width area is recognized as a single pulse. A pulse width outside the undefined receive pulse width area is recognized as a double pulse.

The F_WIDTH bits are used to specify the width in terms of reference clock (48 MHz) cycles to which the FIR output pulse into be modulated to.

Caution During transmission/reception, the contents of this register must not be changed.

27.2.11 MIRCR (0x0C00 0060)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	STA_LEN2	STA_LEN1	STA_LEN0	M_WIDTH4	M_WIDTH3	M_WIDTH2	M_WIDTH1	M_WIDTH0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	1	0	0	1	0	0	1
After reset	0	1	0	0	1	0	0	1

Bit	Name	Function
15:8	RFU	Write 0 to these bits. 0 is returned after a read.
7:5	STA_LEN(2:0)	These bits are used to specify the number of STA (start flag). 111: 32 100: 16 011: 4 010: 2 (default) 001: 1 Others: RFU
4:0	M_WIDTH(4:0)	These bits are used to specify the MIR pulse modulation width. 11111: 32 clocks : 10100: 21 clocks : 01001: 10 clocks (default) : 00001: 2 clocks 00000: 1 clock

This register controls the MIR operation.

The STA_LEN bits are used to specify the number of STA added to the MIR transmit frame.

The M_WIDTH bits are used to specify the MIR pulse modulation width. The MIR output pulse is modulated to a width consisting of the number of reference clocks (48 MHz) specified by these bits.

The nominal pulse width of MIR is 1/4. Therefore, the following setting should normally be made.

MIR full mode (1.152 MHz) = 01001 (rate 10/42)
MIR half mode (0.576 MHz) = 10100 (rate 21/83)

Caution During transmission/reception, the contents of this register must not be changed.

27.2.12 DMACR (0x0C00 0062)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	ACES_MD	TRANS_MD	RFU	RFU	RFU	DEMAND2	DEMAND1	DEMAND0
R/W	R/W	R/W	R	R	R	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:8	RFU	Write 0 to these bits. 0 is returned after a read.
7	ACES_MD	This bit is used to select the access mode. Write 0 when writing. 0 is returned after a read.
6	TRANS_MD	This bit is used to specify the transfer direction. 1: RDR → Memory 0: Memory → TDR
5:3	RFU	Write 0 to these bits. 0 is returned after a read.
2:0	DEMAND(2:0)	These bits are used to specify the demand size. 111: Free size 110: 7 101: 6 100: 5 011: 4 010: 3 001: 2 000: 1

Caution During the DMA operation (both the master side and IrDA side), the contents of this register must not be changed.

27.2.13 DMAER (0x0C00 0064)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RFU	RFU	RFU	RFU	RFU	RFU	DMA_BUSY	DMA_EN
R/W	R	R	R	R	R	R	R	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:2	RFU	Write 0 to these bits. 0 is returned after a read.
1	DMA_BUSY	This bit is used to indicate the busy status. 1: DMA operation in progress 0: DMA operation not in progress
0	DMA_EN	This bit is used as a DMA operation enable trigger. 1: Enable 0: Disable

The DMA_BUSY bit is set automatically by setting the DMA_EN bit to 1, and is cleared by setting bit 0 of the FSR register (TX_STOP) to 1.

Even if 0 is written to DMA_EN during DMA operation, DMA ignores this and continues its operation.

27.2.14 TXIR (0x0C00 0066)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	TX_BUSY	RFU	LAST_TFL	TX_TH_OV	RFU	TXF_UNDR	TXF_FULL	TXF_EMP
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:8	RFU	Write 0 to these bits. 0 is returned after a read.
7	TX_BUSY	Transmission busy. 0: Not in transmission 1: In transmission
6	RFU	Write 0 to this bit. 0 is returned after a read.
5	LAST_TFL	Last transmission frame status. 0: Normal 1: Exists
4	TX_TH_OV	Transmission FIFO threshold over status. 0: Normal 1: Excesses
3	RFU	Write 0 to this bit. 0 is returned after a read.
2	TXF_UNDR	Transmission FIFO underrun status. 0: Normal 1: Data is read
1	TXF_FULL	Transmission FIFO full status. 0: Normal 1: FIFO is full
0	TXF_EMP	Transmission FIFO empty status. 0: Normal 1: Exists

The TX_BUSY bit is set to 1 during the period between PA (in FIR)/STA (in MIR) transmission and abort transmission.

The LAST_TFL bit indicates whether or not data exists in the transmit frame size FIFO. This bit changes when the STA transmission sequence ends. Its default value is 1.

The TX_TH_OV bit indicates whether or not the data size within the transmission FIFO exceeds the threshold.

The TXF_UNDR bit is set when data is read when no data is in the transmission FIFO.

The TXF_FULL bit is set when there is no writable space in the transmission FIFO.

The TXF_EMP bit is set when there is no more data to be read in the transmission FIFO.

27.2.15 RXIR (0x0C00 0068)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RX_BUSY	END_DATA	LAST_RFL	RX_TH_OV	RFU	RFU	RXF_FULL	RXF_EMP
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:8	RFU	Write 0 to these bits. 0 is returned after a read.
7	RX_BUSY	Reception busy. 0: Not in reception 1: In reception
6	END_DATA	Frame last data status. 0: Normal 1: Exists
5	LAST_RFL	Last reception frame status. 0: Normal 1: Result is stored
4	RX_TH_O	Reception FIFO threshold over status. 0: Normal 1: Excesses
3:2	RFU	Write 0 to these bits. 0 is returned after a read.
1	RXF_FULL	Reception FIFO full status. 0: Normal 1: FIFO is full
0	RXF_EMP	Reception FIFO empty status. 0: Normal 1: Exists

The RX_BUSY bit is set to 1 during the period between when PA (in FIR) or STA (in MIR) is detected and when reception ends.

The END_DATA bit is set when the last data of the frame that is received completely exists in the FIFO.

The LAST_RFL bit is set when the reception result (frame size and status) of the 7th frame is stored.

The RX_TH_O bit is set when the data size within the reception FIFO exceeds the threshold.

The RXF_FULL bit is set when there is no writable space in the reception FIFO.

The RXF_EMP bit is set when there is no data to be read in the reception FIFO.

This register's initial value is the value immediately after the IrDA operation is enabled or after the reception FIFO is cleared. 0x00 is also read while the operation is stopped.

Caution This register can be read only in IrDA mode.

27.2.16 IFR (0x0C00 006A)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	TX_ABORT	TX_ERR	RX_VALID	DMA_END	RX_END	TX_END	TX_WR_RQ	RX_RD_RQ
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:8	RFU	Write 0 to these bits. 0 is returned after a read.
7	TX_ABORT	Abort frame transmission end interrupt. 0: Normal 1: Cancelled
6	TX_ERR	Transmission error interrupt. 0: Normal 1: Occurs
5	RX_VALID	Reception result valid interrupt. 0: Normal 1: Valid
4	DMA_END	DMA end interrupt. 0: Normal 1: Ends
3	RX_END	Reception end interrupt. 0: Normal 1: Detected
2	TX_END	Transmission end interrupt. 0: Normal 1: Detected
1	TX_WR_RQ	Transmission data write request interrupt. 0: Normal 1: Occurs
0	RX_RD_RQ	Reception data read request interrupt. 0: Normal 1: Occurs

This register indicates the occurrence of FIR interrupt requests.

The TX_ABORT bit is set when the abort frame is transmitted and the following frame's transfer reservation is canceled.

The TX_ERR bit is set when a transmission error (such as a forcible stop) occurs.

The RX_VALID bit is set when the last data of the frame is read from the reception FIFO and the received status becomes valid.

The DMA_END bit is set when the DMA operation ends.

The RX_END bit is set when the STO is detected for each reception frame.

The TX_END bit is set when the STO is transmitted for each transmission frame.

The TX_WR_RQ bit is set when the transmission data write request interrupt occurs.

The RX_RD_RQ bit is set when the reception data read request interrupt occurs.

If bits 7 through 2 of the IFR register are set, the flags that are set to 1 before a read are all cleared to 0.

27.2.17 RXSTS (0x0C00 006C)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RFU	RFU	RFU	RFU	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	VALID	RFU	RFU	RXF_OV	CRC_ERR	ABORT	MRXF_OV	RFU
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:8	RFU	Write 0 to these bits. 0 is returned after a read.
7	VALID	Valid status in the indication status. 0: Not completed 1: Completed
6:5	RFU	Write 0 to these bits. 0 is returned after a read.
4	RXF_OV	Receive FIFO overrun error. 0: Normal 1: Overrun
3	CRC_ERR	CRC Error. 0: Normal 1: CRC error
2	ABORT	Abort detection error. 0: Normal 1: Abort error
1	MRXF_OV	Maximum receive frame size error. 0: Normal 1: Overrun
0	RFU	Write 0 to this bit. 0 is returned after a read.

This register indicates an error during transmission/reception.

The VALID bit is set to 1 when receive data of one frame is read completely.

The RXF_OV bit is set when a receive operation is stopped due to receive FIFO overrun.

The CRC_ERR bit is set when the receive result CRC does not match the expected value.

The ABORT bit is set when the receive operation is stopped by abort frame detection.

The MRXF_OV bit is set when the receive operation is stopped by maximum receive frame size overrun.

Data of up to 7 frames can be stored in the receive status store FIFO.

The FIFO is initialized by setting bit 1 of the FSR register.

The receive status FIFO is used as follows.

(1) Write (bits 4 to 1)

The receive status is written to this register at the same timing of writing data to the receive frame length register.

This register shares the write pointer with the receive frame length register.

(2) Write (bit 7)

This bit is set to 1 when the data of receive frame size is read from the FIFO. While this bit is 1, data is recognized as valid.

(3) Read

This register shares the read pointer with the receive frame length register.

The read pointer is incremented by reading the RXFL (receive frame length) register after valid data is read from this register.

27.2.18 TXFL (0x0C00 006E)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	TXFL12	TXFL11	TXFL10	TXFL9	TXFL8
R/W	R	R	R	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	TXFL7	TXFL6	TXFL5	TXFL4	TXFL3	TXFL2	TXFL1	TXFL0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:13	RFU	Write 0 to these bits. 0 is returned after a read.
12:0	TXFL(12:0)	Transmit frame size. 0x1FFF: RFU : 0x0802: RFU 0x0801: 2,050 bytes : 0x0001: 2 bytes 0x0000: 1 byte

This register is used to set the address for data write to the transmit frame size data store FIFO. Set the value of ★ transmit size-1 to this register. The setting range is 1 to 2,050 bytes. Operation is not guaranteed if a value exceeding 2,051 bytes is set.

Data of up to 7 frames can be stored in the transmit frame size data store FIFO.

The FIFO is initialized by setting bit 2 of the FSR register.

The transmit frame size FIFO is used as follows.

(1) Write

The data transmit size of frames to be transferred is written to this register.

Transmission is enabled when data is written to this register in the state other than transmission busy state (after FIFO initialization and after transmission completion).

The frames whose number is specified by this register are transferred continuously (back-to-back transfer).

During the single frame transfer, FIFO should be initialized at each 1-frame transfer completion to restart transmit operation.

(2) Read

The sequencer reads the transmission size from this register after the STA flag of transmission frame is transmitted completed. Then, the read pointer is incremented.

Caution If data exists in the FIFO when the STO transmit sequence is completed, continuous transfer mode is entered. When multiple frames are transferred, be sure to write data to the TXFL register before the STO transmit sequence is completed.

27.2.19 MRXF (0x0C00 0070)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	MRXF12	MRXF11	MRXF10	MRXF9	MRXF8
R/W	R	R	R	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	MRXF7	MRXF6	MRXF5	MRXF4	MRXF3	MRXF2	MRXF1	MRXF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:13	RFU	Write 0 to these bits. 0 is returned after a read.
★ 12:0	MRXF(12:0)	Specifies receivable maximum frame size. 0x1FFF: RFU : 0x0802: RFU 0x0801: 2,050 bytes : 0x0001: 2 bytes 0x0000: 1 byte

The maximum receive frame size is set in this register. Operation is not guaranteed if a value exceeding 2,051 bytes is set.

When a 1-frame receive data reaches the setup value of this register, transfer to receive data store FIFO is masked. In this state, the receive of the current frame stops halfway upon arrival of the transfer timing of the next receive data store FIFO. At the same time, maximum frame size is stored in the RXFL register and the MRXF_OV and CRC_ERR bits are set.

If the number of received frames is less than 7 frames, it is possible to continue frame reception.

To receive 8 or more frames, read all the data and frames that are already received from the receive FIFO, then clear the receive FIFO and restart reception.

Remark When receiving data via the DMA operation, set the transfer size value by the following expression:

DMA receivable capacitance = set value × 7 frames

27.2.20 RXFL (0x0C00 0074)

Bit	15	14	13	12	11	10	9	8
Name	RFU	RFU	RFU	RXFL12	RXFL11	RXFL10	RXFL9	RXFL8
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Name	RXFL7	RXFL6	RXFL5	RXFL4	RXFL3	RXFL2	RXFL1	RXFL0
R/W	R	R	R	R	R	R	R	R
RTCRST	0	0	0	0	0	0	0	0
After reset	0	0	0	0	0	0	0	0

Bit	Name	Function
15:13	RFU	Write 0 to these bits. 0 is returned after a read.
12:0	RXFL(12:0)	Receive frame size. 0x1FFF: RFU : 0x0802: RFU 0x0801: 2,050 bytes : 0x0001: 2 bytes 0x0000: 1 byte

★ This register is used to set the address for data read from the receive frame size data store FIFO. The value of transmit size -1 is set to this register. The value to be stored is from 1 to 2,050 bytes.

Data up to 7 frames can be stored in the receive size frame data store FIFO. After 7 frames of data are stored, the receive line is automatically masked. Accordingly, a frame that cannot store a receive result is not transferred to FIFO. FIFO initialization releases this masked status.

The FIFO is initialized by setting bit 1 of the FSR register.

The receive frame size FIFO is used as follows.

(1) Write

When the frame reception is completed after its data is transferred (even if only 1 byte) to the receive FIFO, the sequencer writes the current transfer data size to this register, and the write pointer is incremented.

When the frame reception is completed before its data is transferred to the receive FIFO, write operation is not performed (lost frame).

(2) Read

The read pointer is enabled to be incremented by reading valid data from the RXSTS register, and the next data can be read.

Caution If a receive operation ends abnormally, the data size transferred to the receive FIFO at that time is written to this register.

The update condition of the read pointer of the receive frame size store FIFO is also valid in the test mode.

[MEMO]

CHAPTER 28 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.

28.1 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 (**28.6 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 VR4121 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 28-1.

Table 28-1. CPU Instruction Operation Notations

Symbol	Meaning
←	Assignment
||	Bit string concatenation
x^y	Replication of bit value *x* into a *y*-bit string. *x* is always a single-bit value
$x_{y:z}$	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 VR4121 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 VR4121 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 VR4121 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:

$\text{GPR [rt]} \leftarrow \text{immediate} \mid\mid 0^{16}$

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:

$(\text{immediate}_{15})^{16} \mid\mid \text{immediate}_{15...0}$

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.

28.2 Load and Store Instructions

In the VR4121 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 28-2 are used to summarize the handling of virtual addresses and physical memory.

Table 28-2. Load and Store Common Functions

Function	Meaning
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 28-3, 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 VR4121 since it supports the little-endian order only.

Table 28-3. Access Type Specifications for Loads/Stores

Access Type	Meaning
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.

28.3 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.

28.4 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.

28.5 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 **CHAPTER 7 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:   $\text{GPR [rd]} \leftarrow \text{GPR [rs]} + \text{GPR [rt]}$

64   T:   $\text{temp} \leftarrow \text{GPR [rs]} + \text{GPR [rt]}$
          $\text{GPR [rd]} \leftarrow (\text{temp}_{31})^{32} \,||\, \text{temp}_{31...0}$
```

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:   $\text{GPR [rt]} \leftarrow \text{GPR [rs]} + (\text{immediate}_{15})^{16} \,||\, \text{immediate}_{15...0}$

64   T:   $\text{temp} \leftarrow \text{GPR [ rs]} + (\text{immediate}_{15})^{48} \,||\, \text{immediate}_{15...0}$
          $\text{GPR [rt]} \leftarrow (\text{temp}_{31})^{32} \,||\, \text{temp}_{31...0}$
```

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:    $GPR [rt] \leftarrow 0^{16}$ || (immediate and GPR [rs]$_{15...0}$)

64    T:    $GPR [rt] \leftarrow 0^{48}$ || (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 ← $(\text{offset}_{15})^{14}$ || offset || $0^2$
     T+1:   if condition then
                 PC ←  PC + target
            endif

64   T-1:   condition ← not $SR_{18}$
     T:     target ← $(\text{offset}_{15})^{46}$ || offset || $0^2$
     T+1:   if condition then
                 PC ← PC + target
            endif
```

Exceptions:

Coprocessor unusable exception

Note See the opcode table below, or **28.6 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 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 ← $(offset_{15})^{14}$ || offset || $0^2$
     T+1:  if condition then
                PC ← PC + target
           else
                NullifyCurrentInstruction
           endif

64   T-1:  condition ← not SR
     T:    target ← $(offset_{15})^{46}$ || 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 **28.6 CPU Instruction Opcode Bit Encoding**.

Opcode Table:

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	0
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 **28.6 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 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 ← $(\text{offset}_{15})^{14}$ || offset || $0^2$
     T+1:   if condition then
                PC ← PC + target
            else
                NullifyCurrentInstruction
            endif

64   T-1:   condition ← $SR_{18}$
     T:     target ← $(\text{offset}_{15})^{46}$ || 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 **28.6 CPU Instruction Opcode Bit Encoding**.

Opcode Table:

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	0
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 ← $(offset_{15})^{14}$ || offset || $0^{2}$
              condition ← (GPR [rs] = GPR [rt])
      T+1:    if condition then
                        PC ← PC + target
              else
                        NullifyCurrentInstruction
              endif

64    T:      target ← $(offset_{15})^{46}$ || 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:    $target \leftarrow (offset_{15})^{14} \mid\mid offset \mid\mid 0^{2}$
           $condition \leftarrow (GPR\ [rs]_{31} = 0)$
     T+1:  if condition then
                $PC \leftarrow PC + target$
           endif

64   T:    $target \leftarrow (offset_{15})^{46} \mid\mid offset \mid\mid 0^{2}$
           $condition \leftarrow (GPR\ [rs]_{63} = 0)$
     T+1:  if condition then
                $PC \leftarrow PC + target$
           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 ← $(offset_{15})^{14}$ || offset || $0^2$
             condition ← $(GPR\ [rs]_{31} = 0)$
      T+1:   if condition then
                     PC ← PC + target
             else
                     NullifyCurrentInstruction
             endif

64    T:     target ← $(offset_{15})^{46}$ || offset || $0^2$
             condition ← $(GPR\ [rs]_{63} = 0)$
      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 ← (GPR [rs] $_{31}$ = 0) and (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]$_{63}$ = 0) and (GPR [rs] ≠ $0^{64}$)
     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 0 0 0 0 0 1	rs	BLTZ 0 0 0 0 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, 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 0 0 0 0 0 1	rs	BLTZAL 1 0 0 0 0	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 ← (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 ← (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 ← $(\text{offset}_{15})^{14}$ || 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

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 VR4121, 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, $vAddr_{CACHEBITS...LINEBITS}$ specifies the block.

Index_Load_Tag also uses $vAddr_{LINEBITS...3}$ 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

$op_{4..2}$	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 ← ((offset15)48 || offset15...0) + GPR [base]
            (pAddr, uncached) ← 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 mnimonics 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	0x0000 0000 7FFF FFFF	0xFFFF FFFF 8000 0000
1	1	0xFFFF FFFF FFFF FFFF	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 ← $((\text{GPR}[rs]_{31})^{32} \,||\, \text{GPR } [rs]) * ((\text{GPR}[rt]_{31})^{32} \,||\, \text{GPR } [rt])$
          temp2 ← temp1 + LO
          LO ← temp2
          GPR[rd] ← LO

64, sat=0, us=1 (DMACCU instruction)
    T:    temp1 ← $(0^{32} \,||\, \text{GPR } [rs]) * (0^{32} \,||\, \text{GPR } [rt])$
          temp2 ← temp1 + LO
          LO ← temp2
          GPR[rd] ← LO

64, sat=1, us=0 (DMACCS instruction)
    T:    temp1 ← $((\text{GPR}[rs]_{31})^{32} \,||\, \text{GPR } [rs]) * ((\text{GPR}[rt]_{31})^{32} \,||\, \text{GPR } [rt])$
          temp2 ← saturation(temp1 + LO)
          LO ← temp2
          GPR[rd] ← LO

64, sat=1, us=1 (DMACCUS instruction)
    T:    temp1 ← $(0^{32} \,||\, \text{GPR } [rs]) * (0^{32} \,||\, \text{GPR } [rt])$
          temp2 ← saturation(temp1 + LO)
          LO ← temp2
          GPR[rd] ← 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:    $s \leftarrow 0 \parallel sa$
            $GPR [rd] \leftarrow GPR [rt]_{(63 - s)..0} \parallel 0^{s}$
```

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 ← 1 || sa
            GPR [rd] ← 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 0 0 0 0 0 0	rs	rt	rd	0 0 0 0 0 0	DSRAV 0 1 0 1 1 1
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:    $s \leftarrow 1 \,||\, sa$
            $GPR\,[rd] \leftarrow (GPR\,[rt]_{63})^{s} \,||\, GPR\,[rt]_{63..s}$
```

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:    $s \leftarrow GPR[rs]_{5..0}$
            $GPR[rd] \leftarrow 0^{s} \| GPR[rt]_{63..s}$
```

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 0 0 0 0 0 0	rs	rt	rd	0 0 0 0 0 0	DSUB 1 0 1 1 1 0
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 (SR_2 = 1), then load the PC from the ErrorEPC register and clear the *ERL* bit of the Status register (SR_2). Otherwise (SR_2 = 0), load the PC from the EPC register, and clear the *EXL* bit of the Status register (SR_1 = 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

Remark Refer to **Chapter 16 PMU (POWER MANAGEMENT UNIT)** for details about the operation of the peripheral units at mode transition.

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

31 26	25 0
JAL 0 0 0 0 1 1	target
6	26

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:    $temp \leftarrow target$
           $GPR [31] \leftarrow (PC + 8)_{31..1}$ || ISA MODE
     T+1:  $PC \leftarrow PC_{31..28}$ || temp || $0^2$
           ISA MODE toggle

64   T:    $temp \leftarrow target$
           $GPR [31] \leftarrow (PC + 8)_{63..1}$ || ISA MODE
     T+1:  $PC \leftarrow PC_{63..28}$ || 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 0 0 0 0 0 0	rs	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	JR 0 0 1 0 0 0
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:   $vAddr \leftarrow ((offset_{15})^{16} \| offset_{15..0}) + GPR[base]$
          $(pAddr, uncached) \leftarrow AddressTranslation(vAddr, DATA)$
          $pAddr \leftarrow pAddr_{PSIZE-1..3} \| (pAddr_{2..0} \text{ xor } ReverseEndian^{3})$
          $mem \leftarrow LoadMemory(uncached, BYTE, pAddr, vAddr, DATA)$
          $byte \leftarrow vAddr_{2..0} \text{ xor } BigEndianCPU^{3}$
          $GPR[rt] \leftarrow 0^{24} \| mem_{7+8*byte..8*byte}$

64   T:   $vAddr \leftarrow ((offset_{15})^{48} \| offset_{15..0}) + GPR[base]$
          $(pAddr, uncached) \leftarrow AddressTranslation(vAddr, DATA)$
          $pAddr \leftarrow pAddr_{PSIZE-1..3} \| (pAddr_{2..0} \text{ xor } ReverseEndian^{3})$
          $mem \leftarrow LoadMemory(uncached, BYTE, pAddr, vAddr, DATA)$
          $byte \leftarrow vAddr_{2..0} \text{ xor } BigEndianCPU^{3}$
          $GPR[rt] \leftarrow 0^{56} \| mem_{7+8*byte..8*byte}$
```

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:

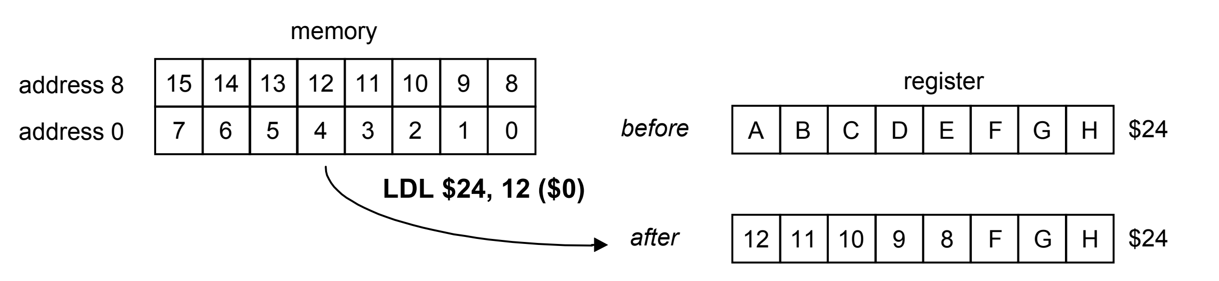
LDL rt, offset (base)

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 ← ((offset$_{15}$)$^{48}$ || offset$_{15..0}$) + GPR [base]
          (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
          pAddr ← pAddr$_{\text{PSIZE - 1..3}}$ || (pAddr$_{2..0}$ xor ReverseEndian$^3$)
          if BigEndianMem = 0 then
                  pAddr ← pAddr$_{\text{PSIZE - 1..3}}$ || 0$^3$
          endif
          byte ← vAddr$_{2..0}$ xor BigEndianCPU$^3$
          mem ← LoadMemory (uncached, byte, pAddr, vAddr, DATA)
          GPR [rt] ← mem$_{7 + 8 * \text{byte}..0}$ || GPR [rt]$_{55 – 8 * \text{byte}..0}$
```

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

$vAddr_{2..0}$	Destination	Type	Offset (LEM)
0	P B C D E F G H	0	0
1	O P C D E F G H	1	0
2	N O P D E F G H	2	0
3	M N O P E F G H	3	0
4	L M N O P F G H	4	0
5	K L M N O P G H	5	0
6	J K L M N O P H	6	0
7	I J K L M N O P	7	0

Remark *Type* AccessType (see **Figure 3-2 Byte Specification Related to Load and Store Instruction**) sent to memory

Offset $pAddr_{2..0}$ sent to memory

LEM Little-endian memory (BigEndianMem = 0)

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:

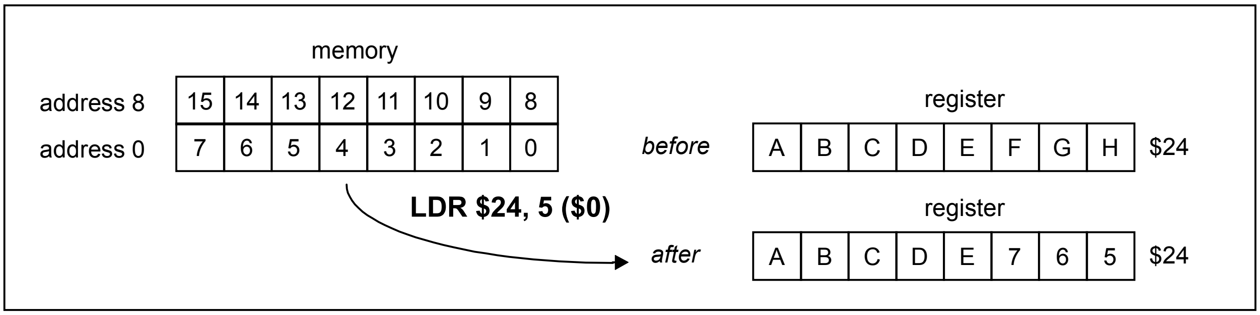
LDR rt, offset (base)

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 ← $((\text{offset}_{15})^{48} \,||\, \text{offset}_{15..0}) + \text{GPR [base]}$
          (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
          pAddr ← $\text{pAddr}_{\text{PSIZE - 1..3}} \,||\, (\text{pAddr}_{2..0} \text{ xor ReverseEndian}^3)$
          if BigEndianMem = 1 then
                pAddr ← $\text{pAddr}_{\text{PSIZE - 1..3}} \,||\, 0^3$
          endif
          byte ← $\text{vAddr}_{2..0} \text{ xor BigEndianCPU}^3$
          mem ← LoadMemory (uncached, DOUBLEWORD-byte, pAddr, vAddr, DATA)
          GPR [rt] ← $\text{GPR [rt]}_{63..64 - 8 * \text{byte}} \,||\, \text{mem}_{63..8 * \text{byte}}$
```

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

$vAddr_{2..0}$	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 *Type* AccessType (see **Figure 3-2 Byte Specification Related to Load and Store Instruction**) sent to memory

Offset $pAddr_{2..0}$ sent to memory

LEM Little-endian memory (BigEndianMem = 0)

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:   $vAddr \leftarrow ((offset_{15})^{16} \| offset_{15...0}) + GPR\ [base]$
          $(pAddr, uncached) \leftarrow AddressTranslation\ (vAddr, DATA)$
          $pAddr \leftarrow pAddr_{PSIZE - 1...3} \| (pAddr_{2...0}\ xor\ (ReverseEndian^{2} \| 0))$
          $mem \leftarrow LoadMemory\ (uncached, HALFWORD, pAddr, vAddr, DATA)$
          $byte \leftarrow vAddr_{2...0}\ xor\ (BigEndianCPU^{2} \| 0)$
          $GPR\ [rt] \leftarrow 0^{16} \| mem_{15 + 8 * byte...8 * byte}$

64   T:   $vAddr \leftarrow ((offset_{15})^{48} \| offset_{15...0}) + GPR\ [base]$
          $(pAddr, uncached) \leftarrow AddressTranslation\ (vAddr, DATA)$
          $pAddr \leftarrow pAddr_{PSIZE - 1...3} \| (pAddr_{2...0}\ xor\ (ReverseEndian^{2} \| 0))$
          $mem \leftarrow LoadMemory\ (uncached, HALFWORD, pAddr, vAddr, DATA)$
          $byte \leftarrow vAddr_{2...0}\ xor\ (BigEndianCPU^{2} \| 0)$
          $GPR\ [rt] \leftarrow 0^{48} \| mem_{15 + 8 * byte...8 * byte}$
```

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:    $GPR [rt] \leftarrow immediate \,||\, 0^{16}$

64    T:    $GPR [rt] \leftarrow (immediate_{15})^{32} \,||\, immediate \,||\, 0^{16}$
```

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 ← ((offset$_{15}$)$^{16}$ || offset$_{15...0}$) + GPR [base]
          (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
          pAddr ← pAddr$_{PSIZE - 1...3}$ || (pAddr$_{2...0}$ xor (ReverseEndian || 0$^{2}$))
          mem ← LoadMemory (uncached, WORD, pAddr, vAddr, DATA)
          byte ← vAddr$_{2...0}$ xor (BigEndianCPU || 0$^{2}$)
          GPR [rt] ← mem$_{31 + 8 * byte...8 * byte}$

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 || 0$^{2}$))
          mem ← LoadMemory (uncached, WORD, pAddr, vAddr, DATA)
          byte ← vAddr$_{2...0}$ xor (BigEndianCPU || 0$^{2}$)
          GPR [rt] ← (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:

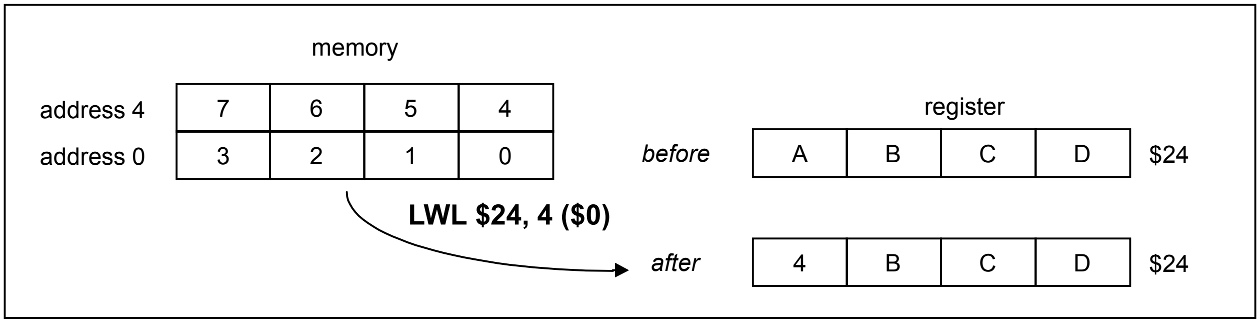
LWL rt, offset (base)

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:   $vAddr \leftarrow ((offset_{15})^{16} \| offset_{15...0}) + GPR\ [base]$
          $(pAddr, uncached) \leftarrow AddressTranslation\ (vAddr, DATA)$
          $pAddr \leftarrow pAddr_{PSIZE-1...3} \| (pAddr_{2...0}$ xor $ReverseEndian^3)$
          if BigEndianMem = 0 then
               $pAddr \leftarrow pAddr_{PSIZE-1...2} \| 0^2$
          endif
          $byte \leftarrow vAddr_{1...0}$ xor $BigEndianCPU^2$
          $word \leftarrow vAddr_2$ xor BigEndianCPU
          $mem \leftarrow LoadMemory\ (uncached, byte, pAddr, vAddr, DATA)$
          $temp \leftarrow mem_{32*word+8*byte+7...32*word} \| GPR\ [rt]_{23-8*byte...0}$
          $GPR\ [rt] \leftarrow temp$

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 = 0 then
               $pAddr \leftarrow pAddr_{PSIZE-1...2} \| 0^2$
          endif
          $byte \leftarrow vAddr_{1...0}$ xor $BigEndianCPU^2$
          $word \leftarrow vAddr_2$ xor BigEndianCPU
          $mem \leftarrow LoadMemory\ (uncached, 0 \| byte, pAddr, vAddr, DATA)$
          $temp \leftarrow mem_{32*word+8*byte+7...32*word} \| GPR\ [rt]_{23-8*byte...0}$
          $GPR\ [rt] \leftarrow (temp_{31})^{32} \| temp$
```

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

$vAddr_{2..0}$	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 *Type* AccessType (see **Figure 3-2 Byte Specification Related to Load and Store Instruction**) sent to memory

Offset $pAddr_{2..0}$ sent to memory

LEM Little-endian memory (BigEndianMem = 0)

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:

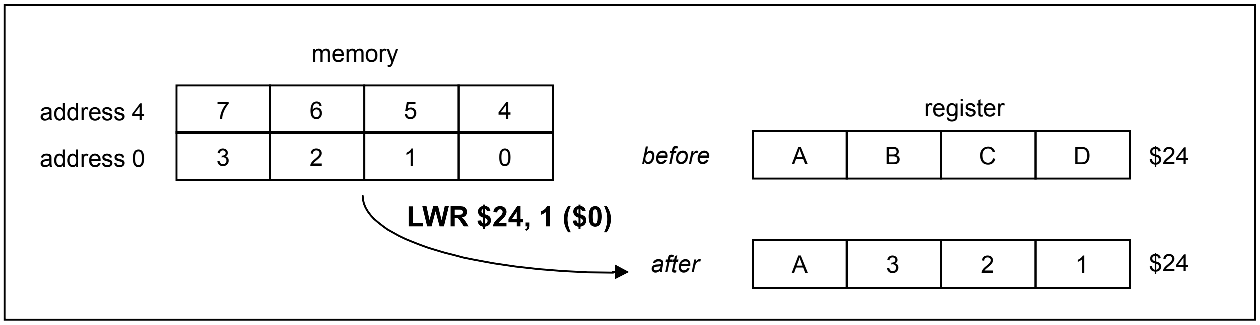
LWR rt, offset (base)

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
          $byte \leftarrow vAddr_{1...0}\ xor\ BigEndianCPU^2$
          $word \leftarrow vAddr_2\ xor\ BigEndianCPU$
          $mem \leftarrow LoadMemory\ (uncached, WORD\text{-}byte, pAddr, vAddr, DATA)$
          $temp \leftarrow GPR\ [rt]_{31...32 - 8 * byte} \| mem_{31 + 32 * word...32 * word + 8 * byte}$
          $GPR\ [rt] \leftarrow (temp_{31})^{32} \| temp$
```

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

$vAddr_{2..0}$	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 *Type* AccessType (see **Figure 3-2 Byte Specification Related to Load and Store Instruction**) sent to memory

Offset $pAddr_{2..0}$ sent to memory

LEM Little-endian memory (BigEndianMem = 0)

S sign-extend of $destination_{31}$

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 mnimonics 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	0x0000 0000 7FFF FFFF	0xFFFF FFFF 8000 0000
1	1	0xFFFF FFFF FFFF FFFF	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:  $temp1 \leftarrow GPR[rs] * GPR[rt]$
        $temp2 \leftarrow temp1 + (HI \,||\, LO)$
        $LO \leftarrow temp2_{63..32}$
        $HI \leftarrow temp2_{31..0}$
        $GPR[rd] \leftarrow LO$
32, sat=0, hi=0, us=1 (MACCU instruction)
    T:  $temp1 \leftarrow (0 \,||\, GPR[rs]) * (0 \,||\, GPR[rt])$
        $temp2 \leftarrow temp1 + ((0 \,||\, HI) \,||\, (0 \,||\, LO))$
        $LO \leftarrow temp2_{63..32}$
        $HI \leftarrow temp2_{31..0}$
        $GPR[rd] \leftarrow LO$
32, sat=0, hi=1, us=0 (MACCHI instruction)
    T:  $temp1 \leftarrow GPR[rs] * GPR[rt]$
        $temp2 \leftarrow temp1 + (HI \,||\, LO)$
        $LO \leftarrow temp2_{63..32}$
        $HI \leftarrow temp2_{31..0}$
        $GPR[rd] \leftarrow HI$
32, sat=0, hi=1, us=1 (MACCHIU instruction)
    T:  $temp1 \leftarrow (0 \,||\, GPR[rs]) * (0 \,||\, GPR[rt])$
        $temp2 \leftarrow temp1 + ((0 \,||\, HI) \,||\, (0 \,||\, LO))$
        $LO \leftarrow temp2_{63..32}$
        $HI \leftarrow temp2_{31..0}$
        $GPR[rd] \leftarrow HI$
32, sat=1, hi=0, us=0 (MACCS instruction)
    T:  $temp1 \leftarrow GPR[rs] * GPR[rt]$
        $temp2 \leftarrow saturation(temp1 + (HI \,||\, LO))$
        $LO \leftarrow temp2_{63..32}$
        $HI \leftarrow temp2_{31..0}$
        $GPR[rd] \leftarrow LO$
32, sat=1, hi=0, us=1 (MACCUS instruction)
    T:  $temp1 \leftarrow (0 \,||\, GPR[rs]) * (0 \,||\, GPR[rt])$
        $temp2 \leftarrow saturation(temp1 + ((0 \,||\, HI) \,||\, (0 \,||\, LO)))$
        $LO \leftarrow temp2_{63..32}$
        $HI \leftarrow temp2_{31..0}$
        $GPR[rd] \leftarrow LO$
```

MACC Multiply and Accumulate (4/5) MACC

```
32, sat=1, hi=1, us=0 (MACCHIS instruction)
    T:  $\text{temp1} \leftarrow \text{GPR[rs]} * \text{GPR[rt]}$
        $\text{temp2} \leftarrow \text{saturation(temp1} + (\text{HI} \,||\, \text{LO}))$
        $\text{LO} \leftarrow \text{temp2}_{63..32}$
        $\text{HI} \leftarrow \text{temp2}_{31..0}$
        $\text{GPR[rd]} \leftarrow \text{HI}$
32, sat=1, hi=1, us=1 (MACCHIUS instruction)
    T:  $\text{temp1} \leftarrow (0 \,||\, \text{GPR[rs]}) * (0 \,||\, \text{GPR[rt]})$
        $\text{temp2} \leftarrow \text{saturation(temp1} + ((0 \,||\, \text{HI}) \,||\, (0 \,||\, \text{LO})))$
        $\text{LO} \leftarrow \text{temp2}_{63..32}$
        $\text{HI} \leftarrow \text{temp2}_{31..0}$
        $\text{GPR[rd]} \leftarrow \text{HI}$
64, sat=0, hi=0, us=0 (MACC instruction)
    T:  $\text{temp1} \leftarrow ((\text{GPR[rs]}_{31})^{32} \,||\, \text{GPR[rs]}) * ((\text{GPR[rt]}_{31})^{32} \,||\, \text{GPR[rt]})$
        $\text{temp2} \leftarrow \text{temp1} + (\text{HI}_{31..0} \,||\, \text{LO}_{31..0})$
        $\text{LO} \leftarrow ((\text{temp2}_{63})^{32} \,||\, \text{temp2}_{63..32})$
        $\text{HI} \leftarrow ((\text{temp2}_{31})^{32} \,||\, \text{temp2}_{31..0})$
        $\text{GPR[rd]} \leftarrow \text{LO}$
64, sat=0, hi=0, us=1 (MACCU instruction)
    T:  $\text{temp1} \leftarrow (0^{32} \,||\, \text{GPR[rs]}) * (0^{32} \,||\, \text{GPR[rt]})$
        $\text{temp2} \leftarrow \text{temp1} + (\text{HI}_{31..0} \,||\, \text{LO}_{31..0})$
        $\text{LO} \leftarrow ((\text{temp2}_{63})^{32} \,||\, \text{temp2}_{63..32})$
        $\text{HI} \leftarrow ((\text{temp2}_{31})^{32} \,||\, \text{temp2}_{31..0})$
        $\text{GPR[rd]} \leftarrow \text{LO}$
64, sat=0, hi=1, us=0 (MACCHI instruction)
    T:  $\text{temp1} \leftarrow ((\text{GPR[rs]}_{31})^{32} \,||\, \text{GPR[rs]}) * ((\text{GPR[rt]}_{31})^{32} \,||\, \text{GPR[rt]})$
        $\text{temp2} \leftarrow \text{temp1} + (\text{HI}_{31..0} \,||\, \text{LO}_{31..0})$
        $\text{LO} \leftarrow ((\text{temp2}_{63})^{32} \,||\, \text{temp2}_{63..32})$
        $\text{HI} \leftarrow ((\text{temp2}_{31})^{32} \,||\, \text{temp2}_{31..0})$
        $\text{GPR[rd]} \leftarrow \text{HI}$
64, sat=0, hi=1, us=1 (MACCHIU instruction)
    T:  $\text{temp1} \leftarrow (0^{32} \,||\, \text{GPR[rs]}) * (0^{32} \,||\, \text{GPR[rt]})$
        $\text{temp2} \leftarrow \text{temp1} + (\text{HI}_{31..0} \,||\, \text{LO}_{31..0})$
        $\text{LO} \leftarrow ((\text{temp2}_{63})^{32} \,||\, \text{temp2}_{63..32})$
        $\text{HI} \leftarrow ((\text{temp2}_{31})^{32} \,||\, \text{temp2}_{31..0})$
        $\text{GPR[rd]} \leftarrow \text{HI}$
```

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 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0	rd	0 0 0 0 0 0	MFHI 0 1 0 0 0 0
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 **CHAPTER 30 VR4121 CPROCESSOR 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 *LO* 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 *HI* 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 0 0 0 0 0 0	rs	rt	0 0 0 0 0 0 0 0 0 0 0	MULT 0 1 1 0 0 0
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:   $GPR [rt] \leftarrow GPR [rs]_{31...16} \| (immediate \text{ or } GPR [rs]_{15...0})$

64   T:   $GPR [rt] \leftarrow GPR [rs]_{63...16} \| (immediate \text{ or } GPR [rs]_{15...0})$
```

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:   $vAddr \leftarrow ((offset_{15})^{48} \mid\mid offset_{15...0}) + GPR[base]$
          (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)

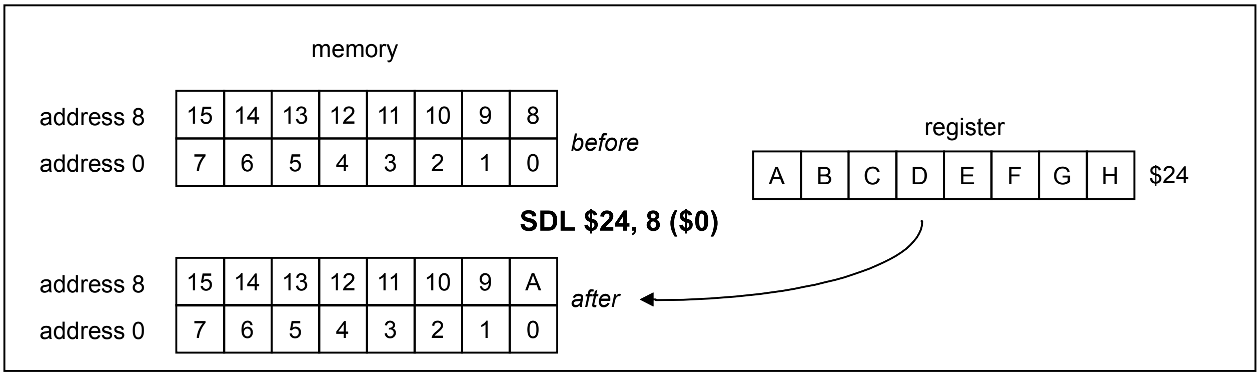
Description:

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 ← $((\mathrm{offset}_{15})^{48} \,||\, \mathrm{offset}_{15...0})$ + GPR [base]
            (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
            pAddr ← $\mathrm{pAddr}_{\mathrm{PSIZE - 1...3}} \,||\, (\mathrm{pAddr}_{2...0}$ xor $\mathrm{ReverseEndian}^{3})$
            if BigEndianMem = 0 then
                    pAddr ← $\mathrm{pAddr}_{\mathrm{PSIZE - 1...3}} \,||\, 0^{3}$
            endif
            byte ← $\mathrm{vAddr}_{2...0}$ xor $\mathrm{BigEndianCPU}^{3}$
            data ← $0^{56 - 8 * \mathrm{byte}} \,||\, \mathrm{GPR}\,[\mathrm{rt}]_{63...56 - 8 * \mathrm{byte}}$
            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

$vAddr_{2..0}$	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 *Type* AccessType (see **Figure 3-2 Byte Specification Related Load and Store Instruction**) sent to memory

Offset $pAddr_{2..0}$ sent to memory

LEM Little-endian memory (BigEndianMem = 0)

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:

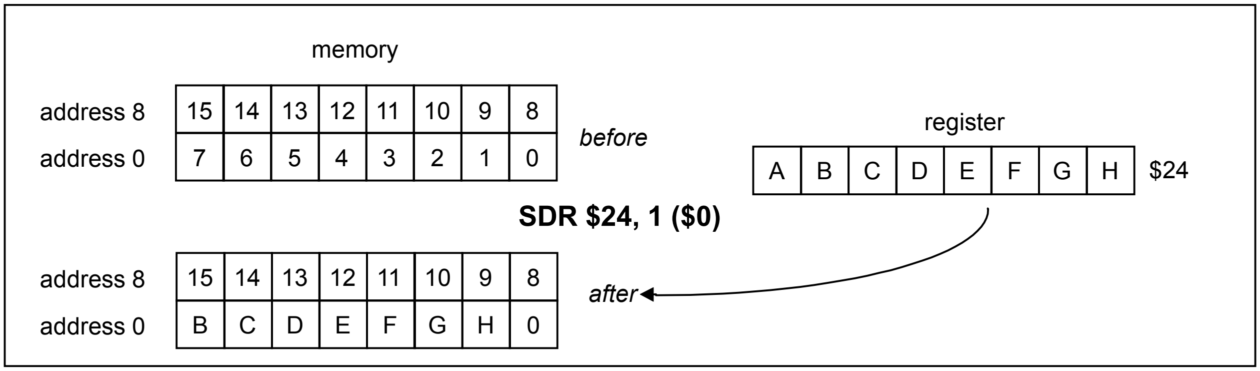
SDR rt, offset (base)

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

$vAddr_{2..0}$	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 *Type* AccessType (see **Figure 3-2 Byte Specification Related Load and Store Instruction**) sent to memory

Offset $pAddr_{2..0}$ sent to memory

LEM Little-endian memory (BigEndianMem = 0)

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 ← $((\text{offset}_{15})^{16} \,||\, \text{offset}_{15...0}) + \text{GPR [base]}$
          (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
          pAddr ← $\text{pAddr}_{\text{PSIZE - 1...3}} \,||\, (\text{pAddr}_{2...0} \text{ xor } (\text{ReverseEndian}^2 \,||\, 0))$
          byte ← $\text{vAddr}_{2...0} \text{ xor } (\text{BigEndianCPU}^2 \,||\, 0)$
          data ← $\text{GPR [rt]}_{63 - 8 * \text{byte}...0} \,||\, 0^{8 * \text{byte}}$
          StoreMemory (uncached, HALFWORD, data, pAddr, vAddr, DATA)

64   T:   vAddr ← $((\text{offset}_{15})^{48} \,||\, \text{offset}_{15...0}) + \text{GPR [base]}$
          (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
          pAddr ← $\text{pAddr}_{\text{PSIZE - 1...3}} \,||\, (\text{pAddr}_{2...0} \text{ xor } (\text{ReverseEndian}^2 \,||\, 0))$
          byte ← $\text{vAddr}_{2...0} \text{ xor } (\text{BigEndianCPU}^2 \,||\, 0)$
          data ← $\text{GPR [rt]}_{63 - 8 * \text{byte}...0} \,||\, 0^{8 * \text{byte}}$
          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 0 0 0 0 0 0	0 0 0 0 0 0	rt	rd	sa	SLL 0 0 0 0 0 0
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:   $GPR [rd] \leftarrow GPR [rt]_{31 - sa...0} || 0^{sa}$

64   T:   $s \leftarrow 0 || sa$
          $temp \leftarrow GPR [rt]_{31 - s...0} || 0^{s}$
          $GPR [rd] \leftarrow (temp_{31})^{32} || temp$
```

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 0 0 0 0 0 0	rs	rt	rd	0 0 0 0 0 0	SLLV 0 0 0 1 0 0
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:   $s \leftarrow \text{GPR}[rs]_{4...0}$
          $\text{GPR}[rd] \leftarrow \text{GPR}[rt]_{(31-s)...0} \,||\, 0^s$

64   T:   $s \leftarrow 0 \,||\, \text{GPR}[rs]_{4...0}$
          $temp \leftarrow \text{GPR}[rt]_{(31-s)...0} \,||\, 0^s$
          $\text{GPR}[rd] \leftarrow (temp_{31})^{32} \,||\, temp$
```

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] < $(\mathrm{immediate}_{15})^{16}$ || $\mathrm{immediate}_{15...0}$ then
              GPR [rt] $\leftarrow 0^{31}$ || 1
          else
              GPR [rt] $\leftarrow 0^{32}$
          endif

64   T:   if GPR [rs] < $(\mathrm{immediate}_{15})^{48}$ || $\mathrm{immediate}_{15...0}$ then
              GPR [rt] $\leftarrow 0^{63}$ || 1
          else
              GPR [rt] $\leftarrow 0^{64}$
          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 || (immediate$_{15}$)$^{16}$ || immediate$_{15...0}$) then
                GPR [rt] $\leftarrow$ 0$^{31}$ || 1
          else
                GPR [rt] $\leftarrow$ 0$^{32}$
          endif

64   T:   if (0 || GPR [rs]) < (0 || (immediate$_{15}$)$^{48}$ || immediate$_{15...0}$) then
                GPR [rt] $\leftarrow$ 0$^{63}$ || 1
          else
                GPR [rt] $\leftarrow$ 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:   $s \leftarrow \text{GPR}[rs]_{4...0}$
          $\text{GPR}[rd] \leftarrow (\text{GPR}[rt]_{31})^{s} \,||\, \text{GPR}[rt]_{31...s}$

64   T:   $s \leftarrow \text{GPR}[rs]_{4...0}$
          $temp \leftarrow (\text{GPR}[rt]_{31})^{s} \,||\, \text{GPR}[rt]_{31...s}$
          $\text{GPR}[rd] \leftarrow (temp_{31})^{32} \,||\, temp$
```

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] ← $0^{sa}$ || GPR $[rt]_{31...sa}$

64    T:    s ← 0 || sa
            temp ← $0^{s}$ || GPR $[rt]_{31...s}$
            GPR [rd] ← $(temp_{31})^{32}$ || 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:   $s \leftarrow GPR[rs]_{4...0}$
          $GPR[rd] \leftarrow 0^{s} \| GPR[rt]_{31...s}$

64   T:   $s \leftarrow GPR[rs]_{4...0}$
          $temp \leftarrow 0^{s} \| GPR[rt]_{31...s}$
          $GPR[rd] \leftarrow (temp_{31})^{32} \| temp$
```

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 **CHAPTER 16 PMU (POWER MANAGEMENT UNIT)** 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 0 0 0 0 0 0	rs	rt	rd	0 0 0 0 0 0	SUB 1 0 0 0 1 0
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] ← $(\text{temp}_{31})^{32}$ || $\text{temp}_{31...0}$
```

Exceptions:

Integer overflow exception

SUBU Subtract Unsigned SUBU

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	SUBU 1 0 0 0 1 1
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]
          $\text{GPR [rd]} \leftarrow (\text{temp}_{31})^{32} \,||\, \text{temp}_{31\ldots0}$
```

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 **CHAPTER 16 PMU (POWER MANAGEMENT UNIT)** 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)

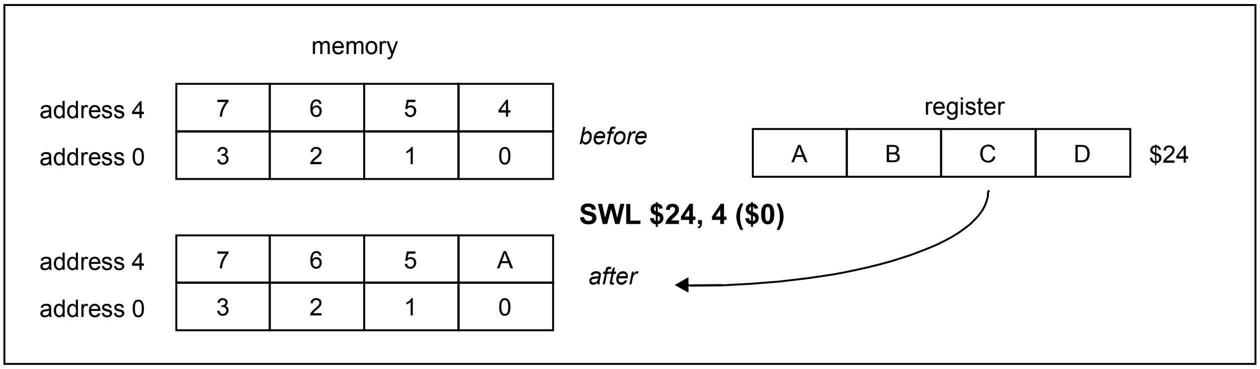
Description:

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:   $vAddr \leftarrow ((offset_{15})^{16} \| offset_{15...0}) + GPR [base]$
          $(pAddr, uncached) \leftarrow AddressTranslation (vAddr, DATA)$
          $pAddr \leftarrow pAddr_{PSIZE - 1...3} \| (pAddr_{2...0}$ xor $ReverseEndian^{3})$
          if BigEndianMem = 0 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
               $data \leftarrow 0^{32} \| 0^{24 - 8 * byte} \| GPR [rt]_{31...24 - 8 * byte}$
          else
               $data \leftarrow 0^{24 - 8 * byte} \| GPR [rt]_{31...24 - 8 * byte} \| 0^{32}$
          endif
          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 $ReverseEndian^{3})$
          if BigEndianMem = 0 then
               $pAddr \leftarrow pAddr_{PSIZE - 1...2} \| 0^{2}$
          endif
          $byte \leftarrow vAddr_{1...0}$ xor $BigEndianCPU^{2}$
          if (vAddr2 xor BigEndianCPU) = 0 then
               $data \leftarrow 0^{32} \| 0^{24 - 8 * byte} \| GPR [rt]_{31...24 - 8 * byte}$
          else
               $data \leftarrow 0^{24 - 8 * byte} \| GPR [rt]_{31...24 - 8 * byte} \| 0^{32}$
          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

$vAddr_{2..0}$	Destination	Type	Offset (LEM)
0	I J K L M N O E	0	0
1	I J K L M N E F	1	0
2	I J K L M E F G	2	0
3	I J K L E F G H	3	0
4	I J K E M N O P	0	4
5	I J E F M N O P	1	4
6	I E F G M N O P	2	4
7	E F G H M N O P	3	4

Remark *Type* AccessType (see **Figure 3-2 Byte Specification Related Load and Store Instruction**) sent to memory

Offset $pAddr_{2..0}$ sent to memory

LEM Little-endian memory (BigEndianMem = 0)

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)

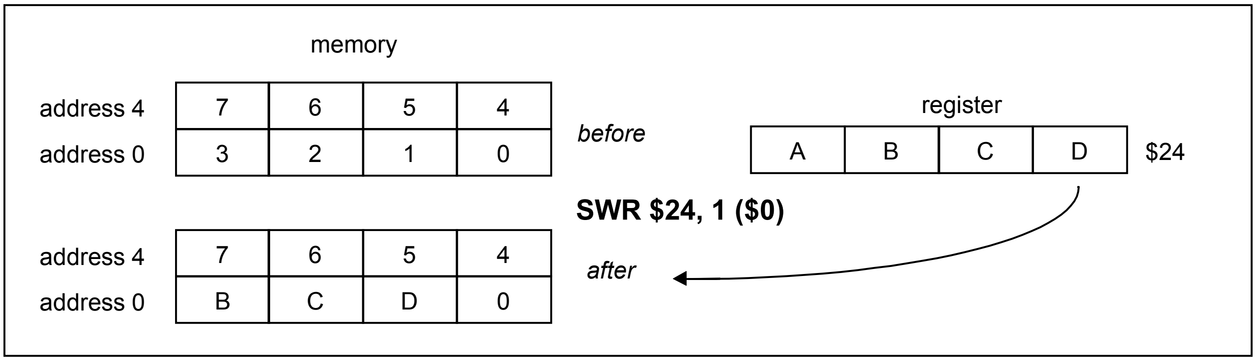
Description:

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

vAddr$_{2..0}$	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 *Type* AccessType (see **Figure 3-2 Byte Specification Related Load and Store Instruction**) sent to memory

Offset pAddr$_{2..0}$ sent to memory

LEM Little-endian memory (BigEndianMem = 0)

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 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	SYNC 0 0 1 1 1 1
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] = $(\text{immediate}_{15})^{16} \mid\mid \text{immediate}_{15...0}$ then
                TrapException
            endif

64    T:    if GPR [rs] = $(\text{immediate}_{15})^{48} \mid\mid \text{immediate}_{15...0}$ 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 0 0 0 0 0 1	rs	TGEIU 0 1 0 0 1	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 $(0 \| \text{GPR [rs]}) \geq (0 \| (\text{immediate}_{15})^{16} \| \text{immediate}_{15...0})$ then
                TrapException
          endif

64   T:   if $(0 \| \text{GPR [rs]}) \geq (0 \| (\text{immediate}_{15})^{48} \| \text{immediate}_{15...0})$ 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 $[i]_{95...77}$ = $\text{EntryHi}_{31...13}$) and (TLB $[i]_{76}$ or
                (TLB $[i]_{71...64}$ = $\text{EntryHi}_{7...0}$)) 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 $[i]_{216...205}$))
                = ($\text{EntryHi}_{39...13}$) and not ($0^{15}$ || TLB $[i]_{216...205}$)) and
                (TLB $[i]_{140}$ or (TLB $[i]_{135...128}$ = $\text{EntryHi}_{7...0}$)) 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:   PageMask ← TLB [Index$_{5...0}$]$_{127...96}$
          EntryHi ← TLB [Index$_{5...0}$]$_{95...64}$ and not TLB [Index$_{5...0}$]$_{127...96}$
          EntryLo1 ← TLB [Index$_{5...0}$]$_{63...33}$ || TLB [Index$_{5...0}$]$_{76}$
          EntryLo0 ← TLB [Index$_{5...0}$]$_{31...1}$ || TLB [Index$_{5...0}$]$_{76}$

64   T:   PageMask ← TLB [Index$_{5...0}$]$_{255...192}$
          EntryHi ← TLB [Index$_{5...0}$]$_{191...128}$ and not TLB [Index$_{5...0}$]$_{255...192}$
          EntryLo1 ← TLB [Index$_{5...0}$]$_{127...65}$ || TLB [Index$_{5...0}$]$_{140}$
          EntryLo0 ← TLB [Index$_{5...0}$]$_{63...1}$ || TLB [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 [$\text{Index}_{5...0}$] ←
                         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 [$Random_{5...0}$] ←
                  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 GPR [rs] $\neq$ $(\text{immediate}_{15})^{16}$ || $\text{immediate}_{15...0}$ then
               TrapException
          endif

64   T:   if 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

28.6 CPU Instruction Opcode Bit Encoding

Figure 28-1 lists the VR4121 Opcode Bit Encoding.

Figure 28-1. VR4121 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 28-1. VR4121 Opcode 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	HIBERNATE	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.

[MEMO]

CHAPTER 29 MIPS16 INSTRUCTION SET FORMAT

This chapter describes the format of each MIPS16 instruction, and the format of the MIPS instructions that are made by converting MIPS16 instructions in alphabetical order. For details of MIPS16 instruction conversion and opcode, refer to **CHAPTER 4 MIPS16 INSTRUCTION SET**.

Caution For some instructions, their format or syntax may become ineffective after they are converted to a 32-bit instruction. For details of formats and syntax of 32-bit instructions, refer to CHAPTER 3 MIPS III INSTRUCTION SET SUMMARY and CHAPTER 28 MIPS III INSTRUCTION SET DETAILS.

ADDIU

Add Immediate Unsigned

(1/2)

ADDIU ry, rx, immediate

15 11	10 8	7 5	4	3 0
RRI-A 0 1 0 0 0	rx	ry	A D D I U 0	immediate
5	3	3	1	4

31 26	25 21	20 16	15 4	3 0
ADDIU 0 0 1 0 0 1	trx	try	sign	immediate
6	5	5	12	4

ADDIU rx, immediate

15 11	10 8	7 0
ADDIU8 0 1 0 0 1	rx	immediate
5	3	8

31 26	25 21	20 16	15 8	7 0
ADDIU 0 0 1 0 0 1	trx	trx	sign	immediate
6	5	5	8	8

ADDIU sp, immediate

15 11	10 8	7 0
I8 0 1 1 0 0	ADJSP 0 1 1	immediate
5	3	8

31 26	25 21	20 16	15 11	10 3	2 0
ADDIU 0 0 1 0 0 1	sp 1 1 1 0 1	sp 1 1 1 0 1	sign	immediate	0 0 0 0
6	5	5	5	8	3

ADDIU

Add Immediate Unsigned

(2/2)

ADDIU rx, pc, immediate

15 11	10 8	7 0
ADDIUSP 0 0 0 0 1	rx	immediate
5	3	8

31 26	25 21	20 16	15 10	9 2	1 0
ADDIU 0 0 1 0 0 1	0 Note 0 0 0 0 0	trx	0 0 0 0 0 0 0	immediate	0 0 0
6	5	5	6	8	2

Note Zeros are shown in the field of bits 21 to 25 as placeholders. The 32-bit PC-relative instruction format shown above is provided here only to make the description complete; it is not a valid 32-bit MIPS instruction. Please see **CHAPTER 4 for MIPS16 INSTRUCTION SET** for a complete definition of the semantics of the MIPS16 PC-relative instructions.

ADDIU rx, sp, immediate

15 11	10 8	7 0
ADDIUSP 0 0 0 0 0	rx	immediate
5	3	8

31 26	25 21	20 16	15 10	9 2	1 0
ADDIU 0 0 1 0 0 1	sp 1 1 1 0 1	trx	0 0 0 0 0 0 0	immediate	0 0 0
6	5	5	6	8	2

ADDU

Add Unsigned

ADDU rz, rx, ry

15 11	10 8	7 5	4 2	1 0
RRR 1 1 1 0 0	rx	ry	rz	ADDU 0 1
5	3	3	3	2

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	trx	try	trz	0 0 0 0 0 0	ADDU 1 0 0 0 0 1
6	5	5	5	5	6

AND

AND

AND rx, ry

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	ry	AND 0 1 1 0 0
5	3	3	5

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	trx	try	trx	0 0 0 0 0 0	AND 1 0 0 1 0 0
6	5	5	5	5	6

B

Branch Unconditional

B immediate

15 11	10 0
B 0 0 0 1 0	immediate
5	11

31 26	25 21	20 16	15 11	10 0
BEQ 0 0 0 1 0 0	zero 0 0 0 0 0	zero 0 0 0 0 0	sign	immediate[Note]
6	5	5	5	11

Note In MIPS16 mode, the branch offset is interpreted as halfword aligned. This is unlike 32-bit MIPS mode which interprets the offset value as word aligned. The 32-bit branch instruction format shown above is provided here only to make the description complete; it is not a valid 32-bit MIPS instruction. Please see **CHAPTER 3 MIPS III INSTRUCTION SET SUMMARY** and **CHAPTER 28 MIPS III INSTRUCTION SET DETAILS** for a complete definition of the semantics of the MIPS16 branch instructions.

BEQZ

Branch On Equal to Zero

BEQZ rx, immediate

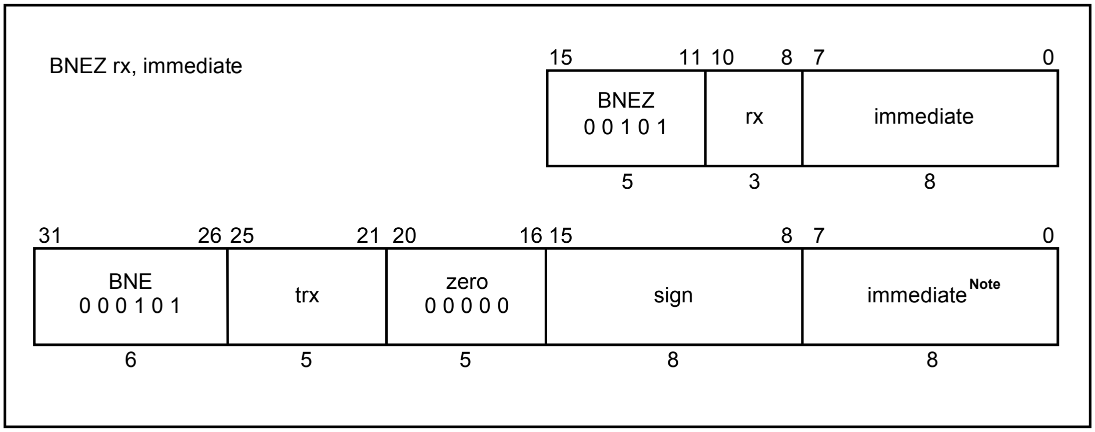
15 11	10 8	7 0
BEQZ 0 0 1 0 0	rx	immediate
5	3	8

31 26	25 21	20 16	15 8	7 0
BEQ 0 0 0 1 0 0	trx	zero 0 0 0 0 0	sign	immediate[Note]
6	5	5	8	8

Note In MIPS16 mode, the branch offset is interpreted as halfword aligned. This is unlike 32-bit MIPS mode which interprets the offset value as word aligned. The 32-bit branch instruction format shown above is provided here only to make the description complete; it is not a valid 32-bit MIPS instruction. Please see **CHAPTER 3 MIPS III INSTRUCTION SET SUMMARY** and **CHAPTER 28 MIPS III INSTRUCTION SET DETAILS** for a complete definition of the semantics of the MIPS16 branch instructions.

BNEZ

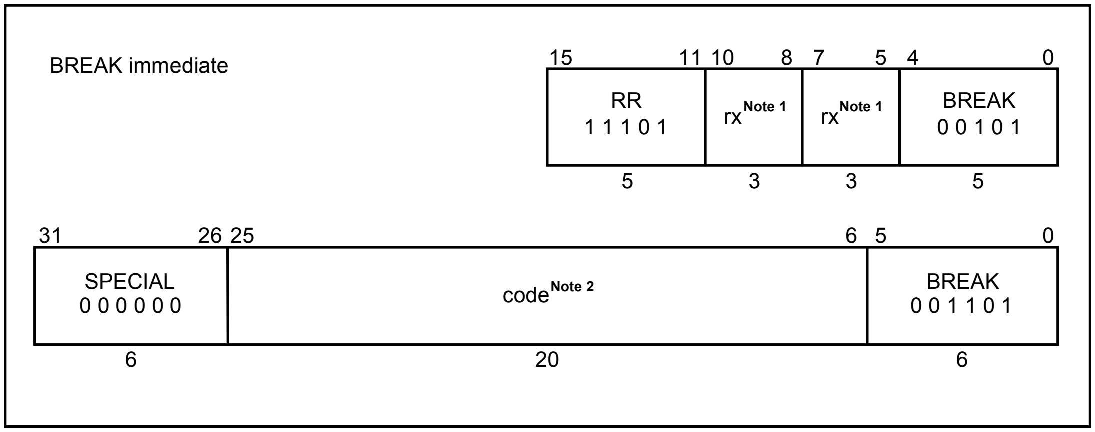
Branch on Not Equal to Zero



Note In MIPS16 mode, the branch offset is interpreted as halfword aligned. This is unlike 32-bit MIPS mode which interprets the offset value as word aligned. The 32-bit branch instruction format shown above is provided here only to make the description complete; it is not a valid 32-bit MIPS instruction. Please see **CHAPTER 3 MIPS III INSTRUCTION SET SUMMARY** and **CHAPTER 28 MIPS III INSTRUCTION SET DETAILS** for a complete definition of the semantics of the MIPS16 branch instructions.

BREAK

Breakpoint

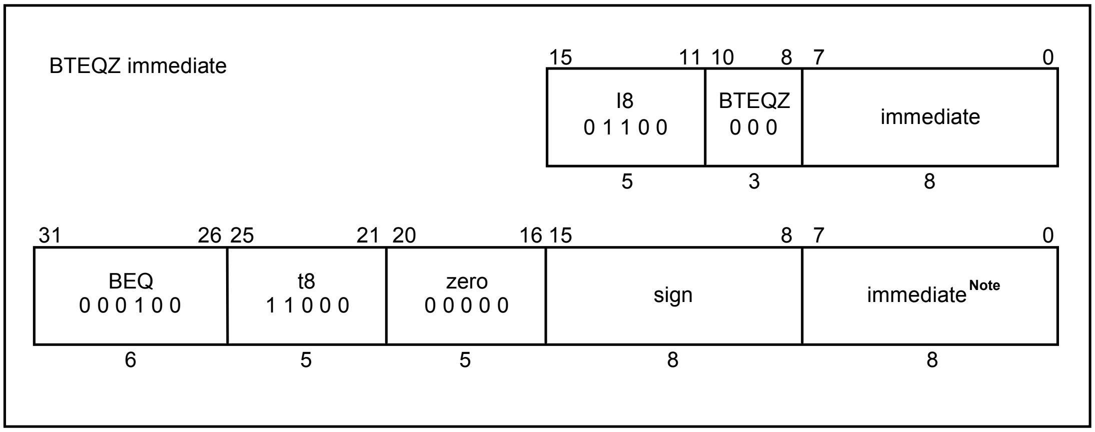


Notes 1. The two register fields in the MIPS16 break instruction may be used as a 6-bit code (immediate) field for software parameters. The 6-bit code can be retrieved by the exception handler.

2. The 32-bit break instruction format shown above is provided here only to make the description complete; it is not a valid 32-bit MIPS instruction. The code field is entirely ignored by the pipeline, and it is not visible in any way to the software executing on the processor.

BTEQZ

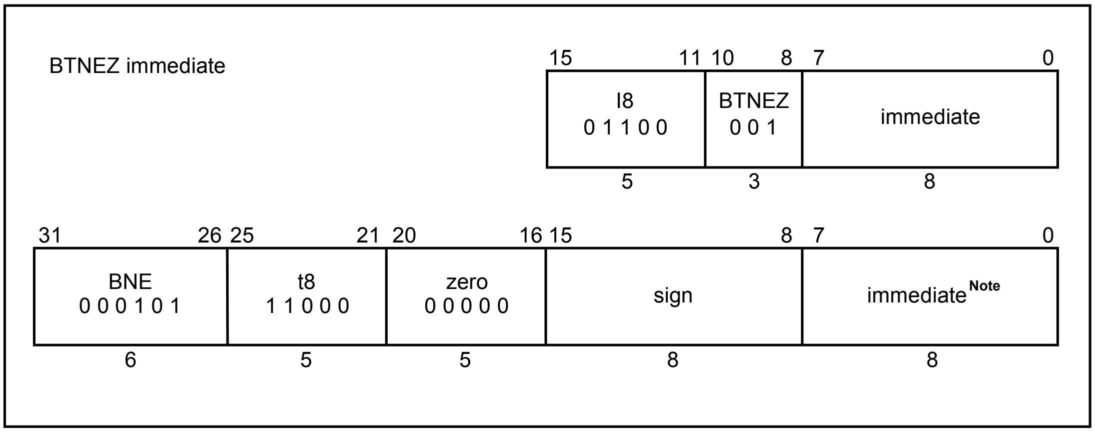
Branch on T Equal to Zero



Note In MIPS16 mode, the branch offset is interpreted as halfword aligned. This is unlike 32-bit MIPS mode which interprets the offset value as word aligned. The 32-bit branch instruction format shown above is provided here only to make the description complete; it is not a valid 32-bit MIPS instruction. Please see **CHAPTER 3 MIPS III INSTRUCTION SET SUMMARY** and **CHAPTER 28 MIPS III INSTRUCTION SET DETAILS** for a complete definition of the semantics of the MIPS16 branch instructions.

BTNEZ

Branch On T Not Equal to Zero



Note In MIPS16 mode, the branch offset is interpreted as halfword aligned. This is unlike 32-bit MIPS mode which interprets the offset value as word aligned. The 32-bit branch instruction format shown above is provided here only to make the description complete; it is not a valid 32-bit MIPS instruction. Please see **CHAPTER 3 MIPS III INSTRUCTION SET SUMMARY** and **CHAPTER 28 MIPS III INSTRUCTION SET DETAILS** for a complete definition of the semantics of the MIPS16 branch instructions.

CMP

Compare

CMP rx, ry

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	ry	CMP 0 1 0 1 0
5	3	3	5

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	trx	try	t8 1 1 0 0 0	0 0 0 0 0 0	XOR 1 0 0 1 1 0
6	5	5	5	5	6

CMPI

Compare Immediate

CMPI rx, immediate

15 11	10 8	7 0
CMPI 0 1 1 1 0	rx	immediate
5	3	8

31 26	25 21	20 16	15 8	7 0
XORI 0 0 1 1 1 0	trx	t8 1 1 0 0 0	0 0 0 0 0 0 0 0	immediate
6	5	5	8	8

DADDIU

Doubleword Add Immediate Unsigned

(1/2)

DADDIU ry, rx, immediate

15 11	10 8	7 5	4	3 0
RRI-A 0 1 0 0 0	rx	ry	D A D D I U 1	immediate
5	3	3	1	4

31 26	25 21	20 16	15 4	3 0
DADDIU 0 1 1 0 0 1	trx	try	sign	immediate
6	5	5	12	4

DADDIU ry, immediate

15 11	10 8	7 5	4 0
I64 1 1 1 1 1	DADD IU5 1 0 1	ry	immediate
5	3	3	5

31 26	25 21	20 16	15 5	4 0
DADDIU 0 1 1 0 0 1	try	try	sign	immediate
6	5	5	11	5

DADDIU ry, pc, immediate

15 11	10 8	7 5	4 0
I64 1 1 1 1 1	DADDIU PC 1 1 1	ry	immediate
5	3	3	5

31 26	25 21	20 16	15 7	6 2	1 0
DADDIU 0 1 1 0 0 1	0Note 0 0 0 0 0	try	0 0 0 0 0 0 0 0 0 0	immediate	0 0 0
6	5	5	9	5	2

Note Zeros are shown in the field of bits 21 to 25 as placeholders. The 32-bit PC-relative instruction format shown above is provided here only to make the description complete; it is not a valid 32-bit MIPS instruction. Please see **CHAPTER 4 MIPS16 INSTRUCTION SET** for a complete definition of the semantics of the MIPS16 PC-relative instructions.

DADDIU

Doubleword Add Immediate Unsigned

(2/2)

DADDIU ry, sp, immediate

15 11	10 8	7 5	4 0
I64 1 1 1 1 1	DADDIU SP 1 1 1	ry	immediate
5	3	3	5

31 26	25 21	20 16	15 7	6 2	1 0
DADDIU 0 1 1 0 0 1	sp 1 1 1 0 1	try	0 0 0 0 0 0 0 0 0 0	immediate	0 0 0
6	5	5	9	5	2

DADDIU sp, immediate

15 11	10 8	7 0
I64 1 1 1 1 1	DADJ SP 0 1 1	immediate
5	3	8

31 26	25 21	20 16	15 11	10 3	2 0
DADDIU 0 1 1 0 0 1	sp 1 1 1 0 1	sp 1 1 1 0 1	sign	immediate	0 0 0 0
6	5	5	5	8	3

DADDU

Doubleword Add Unsigned

DADDU rz, rx, ry

15 11	10 8	7 5	4 2	1 0
RRR 1 1 1 0 0	rx	ry	rz	DADDU 0 0
5	3	3	3	2

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	trx	try	trz	0 0 0 0 0 0	DADDU 1 0 1 1 0 1
6	5	5	5	5	6

DDIV

Doubleword Divide

DDIV rx, ry

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	ry	DDIV 1 1 1 1 0
5	3	3	5

31 26	25 21	20 16	15 6	5 0
SPECIAL 0 0 0 0 0 0	trx	try	0 0 0 0 0 0 0 0 0 0 0	DDIV 0 1 1 1 1 0
6	5	5	10	6

DDIVU

Doubleword Divide Unsigned

DDIVU rx, ry

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	ry	DDIVU 1 1 1 1 1
5	3	3	5

31 26	25 21	20 16	15 6	5 0
SPECIAL 0 0 0 0 0 0	trx	try	0 0 0 0 0 0 0 0 0 0 0	DDIVU 0 1 1 1 1 1
6	5	5	10	6

DIV

Divide

DIV rx, ry

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	ry	DIV 1 1 0 1 0
5	3	3	5

31 26	25 21	20 16	15 6	5 0
SPECIAL 0 0 0 0 0 0	trx	try	0 0 0 0 0 0 0 0 0 0 0	DIV 0 1 1 0 1 0
6	5	5	10	6

DIVU

Divide Unsigned

DIVU rx, ry

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	ry	DIVU 1 1 0 1 1
5	3	3	5

31 26	25 21	20 16	15 6	5 0
SPECIAL 0 0 0 0 0 0	trx	try	0 0 0 0 0 0 0 0 0 0 0	DIVU 0 1 1 0 1 1
6	5	5	10	6

DMULT

Doubleword Multiply

DMULT rx, ry

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	ry	DMULT 1 1 1 0 0
5	3	3	5

31 26	25 21	20 16	15 6	5 0
SPECIAL 0 0 0 0 0 0	trx	try	0 0 0 0 0 0 0 0 0 0 0	DMULT 0 1 1 1 0 0
6	5	5	10	6

DMULTU

Doubleword Multiply Unsigned

DMULTU rx, ry

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	ry	DMULTU 1 1 1 0 1
5	3	3	5

31 26	25 21	20 16	15 6	5 0
SPECIAL 0 0 0 0 0 0	trx	try	0 0 0 0 0 0 0 0 0 0 0	DMULTU 0 1 1 1 0 1
6	5	5	10	6

DSLL

Doubleword Shift Left Logical

DSLL rx, ry, immediate

15 11	10 8	7 5	4 2	1 0
SHIFT 0 0 1 1 0	rx	ry	shamt	DSLL 0 1
5	3	3	3	2

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	try	trx	sa	DSLL 1 1 1 0 0 0
6	5	5	5	5	6

DSLLV

Doubleword Shift Left Logical Variable

DSLLV ry, rx

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	ry	DSLLV 1 0 1 0 0
5	3	3	5

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	trx	try	try	0 0 0 0 0 0	DSLLV 0 1 0 1 0 0
6	5	5	5	5	6

DSRA

Doubleword Shift Right Arithmetic

DSRA ry, immediate

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	shamt	ry	DSRA 1 0 0 1 1
5	3	3	5

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	try	try	sa	DSRA 1 1 1 0 1 1
6	5	5	5	5	6

DSRAV

Doubleword Shift Right Arithmetic Variable

DSRAV ry, rx

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	ry	DSRAV 1 0 1 1 1
5	3	3	5

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	trx	try	try	0 0 0 0 0 0	DSRAV 0 1 0 1 1 1
6	5	5	5	5	6

DSRL

Doubleword Shift Right Logical

DSRL ry, immediate

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	shamt	ry	DSRL 0 1 0 0 0
5	3	3	5

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	try	try	sa	DSRL 1 1 1 0 1 0
6	5	5	5	5	6

DSRLV

Doubleword Shift Right Logical Variable

DSRLV ry, rx

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	ry	DSRLV 1 0 1 1 0
5	3	3	5

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	trx	try	try	0 0 0 0 0 0	DSRLV 0 1 0 1 1 0
6	5	5	5	5	6

DSUBU

Doubleword Subtract Unsigned

DSUBU rz, rx, ry

15 11	10 8	7 5	4 2	1 0
RRR 1 1 1 0 0	rx	ry	rz	DSUBU 1 0
5	3	3	3	2

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	trx	try	trz	0 0 0 0 0 0	DSUBU 1 0 1 1 1 1
6	5	5	5	5	6

JAL

Jump and Link

JAL target

15 11	10	9 5	4 0
JAL 0 0 0 1 1	0 0	immediate 20:16	immediate 25:21
5	1	5	5

15 0
immediate 15:0
16

31 26	25 0
JAL 0 0 0 0 1 1	target address
6	26

JALR

Jump and Link Register

JALR ra, rx

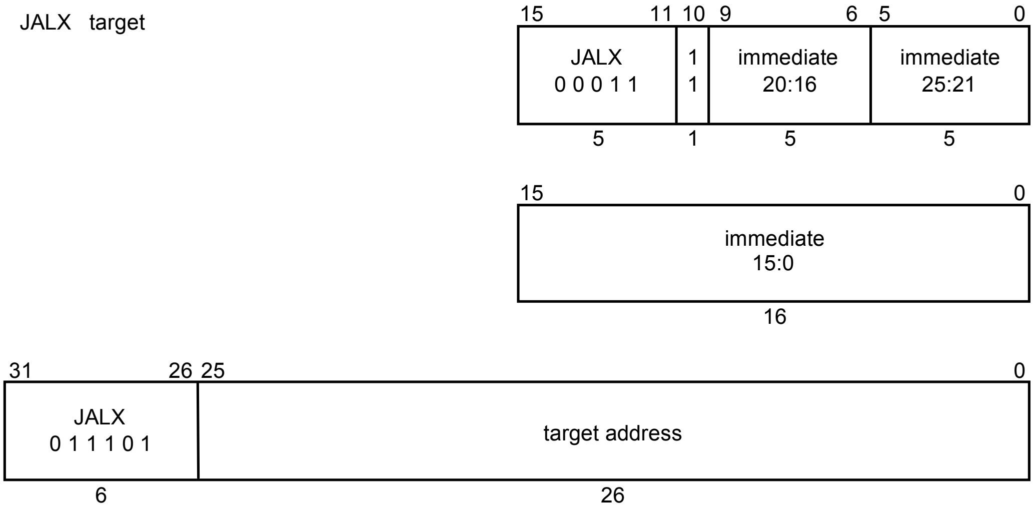
15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	0 1 0	JALR 0 0 0 0 0
5	3	3	5

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	trx	0 0 0 0 0 0	ra 1 1 1 1 1	0 0 0 0 0	JALR 0 0 1 0 0 1
6	5	5	5	5	6

JALX

Jump and Link Exchange

JALX target



JR

Jump Register

JR rx

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	0 0 0	JR 0 0 0 0 0
5	3	3	5

31 26	25 21	20 6	5 0
SPECIAL 0 0 0 0 0 0	trx	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	JR 0 0 1 0 0 0
6	5	15	6

JR ra

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	0 0 0	0 0 1	JR 0 0 0 0 0
5	3	3	5

31 26	25 21	20 6	5 0
SPECIAL 0 0 0 0 0 0	ra 1 1 1 1 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	JR 0 0 1 0 0 0
6	5	15	6

LB

Load Byte

LB ry, offset (rx)

15 11	10 8	7 5	4 0
LB 1 0 0 0 0	rx	ry	immediate
5	3	3	5

31 26	25 21	20 16	15 5	4 0
LB 1 0 0 0 0 0	trx	try	0 0 0 0 0 0 0 0 0 0 0 0	immediate
6	5	5	11	5

LBU

Load Byte Unsigned

LBU ry, offset (rx)

15 11	10 8	7 5	4 0
LBU 1 0 1 0 0	rx	ry	immediate
5	3	3	5

31 26	25 21	20 16	15 5	4 0
LBU 1 0 0 1 0 0	trx	try	0 0 0 0 0 0 0 0 0 0 0 0	immediate
6	5	5	11	5

LD

Load Doubleword

LD ry, offset (rx)

15 11	10 8	7 5	4 0
LD 0 0 1 1 1	rx	ry	immediate
5	3	3	5

31 26	25 21	20 16	15 8	7 3	2 0
LD 1 1 0 1 1 1	trx	try	0 0 0 0 0 0 0 0 0	immediate	0 0 0 0
6	5	5	8	5	3

LD ry, offset (pc)

15 11	10 8	7 5	4 0
I64 1 1 1 1 1	LDPC 1 0 0	ry	immediate
5	3	3	5

31 26	25 21	20 16	15 8	7 3	2 0
LD 1 1 0 1 1 1	0[Note] 0 0 0 0 0	try	0 0 0 0 0 0 0 0 0	immediate	0 0 0 0
6	5	5	8	5	3

Note Zeros are shown in the field of bits 21 to 25 as placeholders. The 32-bit PC-relative instruction format shown above is provided here only to make the description complete; it is not a valid 32-bit MIPS instruction. Please see **CHAPTER 4 MIPS16 INSTRUCION SET** for a complete definition of the semantics of the MIPS16 PC-relative instructions.

LD ry, offset (sp)

15 11	10 8	7 5	4 0
I64 1 1 1 1 1	LDSP 0 0 0	ry	immediate
5	3	3	5

31 26	25 21	20 16	15 8	7 3	2 0
LD 1 1 0 1 1 1	sp 1 1 1 0 1	try	0 0 0 0 0 0 0 0 0	immediate	0 0 0 0
6	5	5	8	5	3

LH

Load Halfword

LH ry, offset (rx)

15 11	10 8	7 5	4 0
LH 1 0 0 0 1	rx	ry	immediate
5	3	3	5

31 26	25 21	20 16	15 6	5 1	0
LH 1 0 0 0 0 1	trx	try	0 0 0 0 0 0 0 0 0 0 0	immediate	0 0
6	5	5	10	5	1

LHU

Load Halfword Unsigned

LHU ry, offset (rx)

15 11	10 8	7 5	4 0
LHU 1 0 1 0 1	rx	ry	immediate
5	3	3	5

31 26	25 21	20 16	15 6	5 1	0
LHU 1 0 0 1 0 1	trx	try	0 0 0 0 0 0 0 0 0 0 0	immediate	0 0
6	5	5	10	5	1

LI

Load Immediate

LI rx, immediate

15 11	10 8	7 0
LI 0 1 1 0 1	rx	immediate
5	3	8

31 26	25 21	20 16	15 8	7 0
ORI 0 0 1 1 0 1	zero 0 0 0 0 0	trx	0 0 0 0 0 0 0 0 0	immediate
6	5	5	8	8

LW

Load Word

(1/2)

LW ry, offset (rx)

15 11	10 8	7 5	4 0
LW 1 0 0 1 1	rx	ry	immediate
5	3	3	5

31 26	25 21	20 16	15 7	6 2	1 0
LW 1 0 0 0 1 1	trx	try	0 0 0 0 0 0 0 0 0 0	immediate	0 0 0
6	5	5	9	5	2

LW

Load Word

(2/2)

LW rx, offset (pc)

15 11	10 8	7 0
LWPC 1 0 1 1 0	rx	immediate
5	3	8

31 26	25 21	20 16	15 10	9 2	1 0
LW 1 0 0 0 1 1	0 Note 0 0 0 0 0	trx	0 0 0 0 0 0 0	immediate	0 0 0
6	5	5	6	8	2

Note Zeros are shown in the field of bits 21 to 25 as placeholders. The 32-bit PC-relative instruction format shown above is provided here only to make the description complete; it is not a valid 32-bit MIPS instruction. Please see **CHAPTER 4 MIPS16 INSTRUCTION SET** for a complete definition of the semantics of the MIPS16 PC-relative instructions.

LW rx, offset (sp)

15 11	10 8	7 0
LWSP 1 0 0 1 0	rx	immediate
5	3	8

31 26	25 21	20 16	15 10	9 2	1 0
LW 1 0 0 0 1 1	sp 1 1 1 0 1	trx	0 0 0 0 0 0 0	immediate	0 0 0
6	5	5	6	8	2

LWU

Load Word Unsigned

LWU ry, offset (rx)

15 11	10 8	7 5	4 0
LWU 1 0 1 1 1	rx	ry	immediate
5	3	3	5

31 26	25 21	20 16	15 7	6 2	1 0
LWU 1 0 0 1 1 1	trx	try	0 0 0 0 0 0 0 0 0 0	immediate	0 0 0
6	5	5	9	5	2

MFHI

Move From HI Register

MFHI rx

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	0 0 0	MFHI 1 0 0 0 0
5	3	3	5

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	trx	0 0 0 0 0 0	MFHI 0 1 0 0 0 0
6	10	5	5	6

MFLO

Move From LO Register

MFLO rx

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	0 0 0	MFLO 1 0 0 1 0
5	3	3	5

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	trx	0 0 0 0 0 0	MFLO 0 1 0 0 1 0
6	10	5	5	6

MOVE

Move

MOVE ry, r32

15 11	10 8	7 5	4 0
I8 0 1 1 0 0	movr32 1 1 1	ry	r32
5	3	3	5

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	r32	zero 0 0 0 0 0	try	0 0 0 0 0 0	OR 1 0 0 1 0 1
6	5	5	5	5	6

MOVE r32 rz

15 11	10 8	7 3	2 0
I8 0 1 1 0 0	mov32r 1 0 1	r32 2:0, 4:3	rz
5	3	5	3

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	trz	zero 0 0 0 0 0	r32	0 0 0 0 0 0	OR 1 0 0 1 0 1
6	5	5	5	5	6

MULT

Multiply

MULT rx, ry

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	ry	MULT 1 1 0 0 0
5	3	3	5

31 26	25 21	20 16	15 6	5 0
SPECIAL 0 0 0 0 0 0	trx	try	0 0 0 0 0 0 0 0 0 0 0	MULT 0 1 1 0 0 0
6	5	5	10	6

MULTU

Multiply Unsigned

MULTU rx, ry

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	ry	MULTU 1 1 0 0 1
5	3	3	5

31 26	25 21	20 16	15 6	5 0
SPECIAL 0 0 0 0 0 0	trx	try	0 0 0 0 0 0 0 0 0 0 0	MULTU 0 1 1 0 0 1
6	5	5	10	6

NEG

Negate

NEG rx, ry

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	ry	NEG 0 1 0 1 1
5	3	3	5

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	zero 0 0 0 0 0	try	trx	0 0 0 0 0 0	SUBU 1 0 0 0 1 1
6	5	5	5	5	6

NOT

NOT

NOT rx, ry

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	ry	NOT 0 1 1 1 1
5	3	3	5

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	zero 0 0 0 0 0	try	trx	0 0 0 0 0 0	NOR 1 0 0 1 1 1
6	5	5	5	5	6

OR

OR

OR rx, ry

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	ry	OR 0 1 1 0 1
5	3	3	5

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	trx	try	trx	0 0 0 0 0 0	OR 1 0 0 1 0 1
6	5	5	5	5	6

SB

Store Byte

SB ry, offset (rx)

15 11	10 8	7 5	4 0
SB 1 1 0 0 0	rx	ry	immediate
5	3	3	5

31 26	25 21	20 16	15 5	4 0
SB 1 0 1 0 0 0	trx	try	0 0 0 0 0 0 0 0 0 0 0 0	immediate
6	5	5	11	5

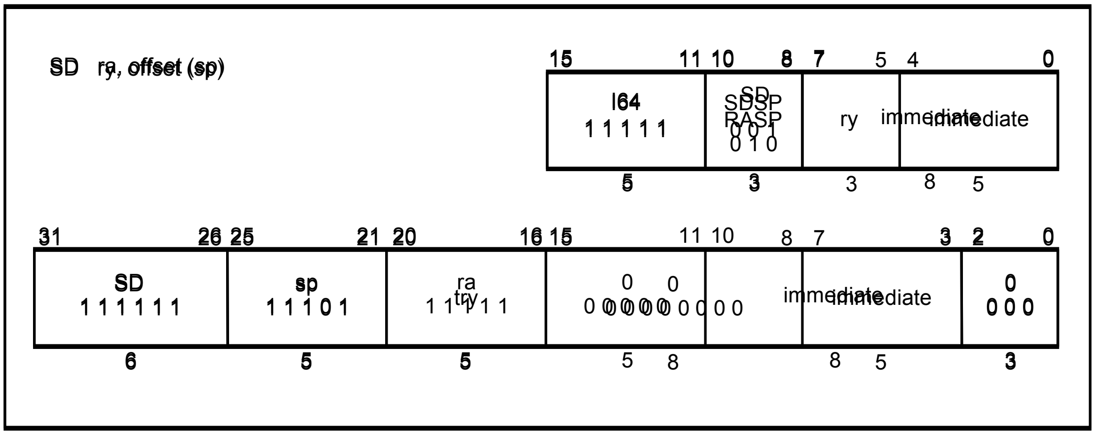
SD

Store Doubleword

SD ry, offset (rx)

15 11	10 8	7 5	4 0
SD 0 1 1 1 1	rx	ry	immediate
5	3	3	5

31 26	25 21	20 16	15 8	7 3	2 0
SD 1 1 1 1 1 1	trx	try	0 0 0 0 0 0 0 0 0	immediate	0 0 0 0
6	5	5	8	5	3



SH

Store Halfword

SH ry, offset (rx)

15 11	10 8	7 5	4 0
SH 1 1 0 0 1	rx	ry	immediate
5	3	3	5

31 26	25 21	20 16	15 6	5 1	0
SH 1 0 1 0 0 1	trx	try	0 0 0 0 0 0 0 0 0 0 0	immediate	0
6	5	5	10	5	1

SLL

Shift Left Logical

SLL rx, ry, immediate

15 11	10 8	7 5	4 2	1 0
SHIFT 0 0 1 1 0	rx	ry	shamt	SLL 0 0
5	3	3	3	2

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	try	trx	sa	SLL 0 0 0 0 0 0
6	5	5	5	5	6

SLLV

Shift Left Logical Variable

SLLV ry, rx

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	ry	SLLV 0 0 1 0 0
5	3	3	5

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	trx	try	try	0 0 0 0 0 0	SLLV 0 0 0 1 0 0
6	5	5	5	5	6

SLT

Set on Less Than

SLT rx, ry

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	ry	SLT 0 0 0 1 0
5	3	3	5

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	trx	try	t8 1 1 0 0 0	0 0 0 0 0 0	SLT 1 0 1 0 1 0
6	5	5	5	5	6

SLTI

Set on Less Than Immediate

SLTI rx, immediate

15 11	10 8	7 0
SLTI 0 1 0 1 0	rx	immediate
5	3	8

31 26	25 21	20 16	15 8	7 0
SLTI 0 0 1 0 1 0	trx	t8 1 1 0 0 0	0 0 0 0 0 0 0 0 0	immediate
6	5	5	8	8

SLTIU

Set on Less Than Immediate Unsigned

SLTIU rx, immediate

15 11	10 8	7 0
SLTIU 0 1 0 1 1	rx	immediate
5	3	8

31 26	25 21	20 16	15 8	7 0
SLTIU 0 0 1 0 1 1	trx	t8 1 1 0 0 0	0 0 0 0 0 0 0 0 0	immediate
6	5	5	8	8

SLTU

Set on Less Than Unsigned

SLTU rx, ry

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	ry	SLTU 0 0 0 1 1
5	3	3	5

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	trx	try	t8 1 1 0 0 0	0 0 0 0 0 0	SLTU 1 0 1 0 1 1
6	5	5	5	5	6

SRA

Shift Right Arithmetic

SRA rx, ry, immediate

15 11	10 8	7 5	4 2	1 0
SHIFT 0 0 1 1 0	rx	ry	shamt	SRA 1 1
5	3	3	3	2

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	try	trx	sa	SRA 0 0 0 0 1 1
6	5	5	5	5	6

SRAV

Shift Right Arithmetic Variable

SRAV ry, rx

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	ry	SRAV 0 0 1 1 1
5	3	3	5

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	trx	try	try	0 0 0 0 0 0	SRAV 0 0 0 1 1 1
6	5	5	5	5	6

SRL

Shift Right Logical

SRL rx, ry, immediate

15 11	10 8	7 5	4 2	1 0
SHIFT 0 0 1 1 0	rx	ry	shamt	SRL 1 0
5	3	3	3	2

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	try	trx	sa	SRL 0 0 0 0 1 0
6	5	5	5	5	6

SRLV

Shift Right Logical Variable

SRLV ry, rx

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	ry	SRLV 0 0 1 1 0
5	3	3	5

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	trx	try	try	0 0 0 0 0 0	SRLV 0 0 0 1 1 0
6	5	5	5	5	6

SW

Store Word

SW ry, offset (rx)

15 11	10 8	7 5	4 0
SW 1 1 0 1 1	rx	ry	immediate
5	3	3	5

31 26	25 21	20 16	15 7	6 2	1 0
SW 1 0 1 0 1 1	trx	try	0 0 0 0 0 0 0 0 0 0	immediate	0 0 0
6	5	5	9	5	2

SW rx, offset (sp)

15 11	10 8	7 0
SWSP 1 1 0 1 0	rx	immediate
5	3	8

31 26	25 21	20 16	15 10	9 2	1 0
SW 1 0 1 0 1 1	sp 1 1 1 0 1	trx	0 0 0 0 0 0 0	immediate	0 0 0
6	5	5	6	8	2

SW ra, offset (sp)

15 11	10 8	7 0
I8 0 1 1 0 0	SW RASP 0 1 0	immediate
5	3	8

31 26	25 21	20 16	15 10	9 2	1 0
SW 1 0 1 0 1 1	sp 1 1 1 0 1	ra 1 1 1 1 1	0 0 0 0 0 0 0	immediate	0 0 0
6	5	5	6	8	2

SUBU

Subtract Unsigned

SUBU rz, rx, ry

15 11	10 8	7 5	4 2	1 0
RRR 1 1 1 0 0	rx	ry	rz	SUBU 1 1
5	3	3	3	2

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	trx	try	trz	0 0 0 0 0 0	SUBU 1 0 0 0 1 1
6	5	5	5	5	6

XOR

Exclusive OR

XOR rx, ry

15 11	10 8	7 5	4 0
RR 1 1 1 0 1	rx	ry	XOR 0 1 1 1 0
5	3	3	5

31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 0 0 0 0 0 0	trx	try	trx	0 0 0 0 0 0	XOR 1 0 0 1 1 0
6	5	5	5	5	6

CHAPTER 30 VR4121 COPROCESSOR 0 HAZARDS

The VR4120 CPU 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 VR4120 CPU core are as or less stringent than those of the VR4000. Table 30-1 lists the Coprocessor 0 hazards of the VR4120 CPU 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 30-1. VR4121 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 30-2 indicates examples of calculation.

Table 30-2. 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)

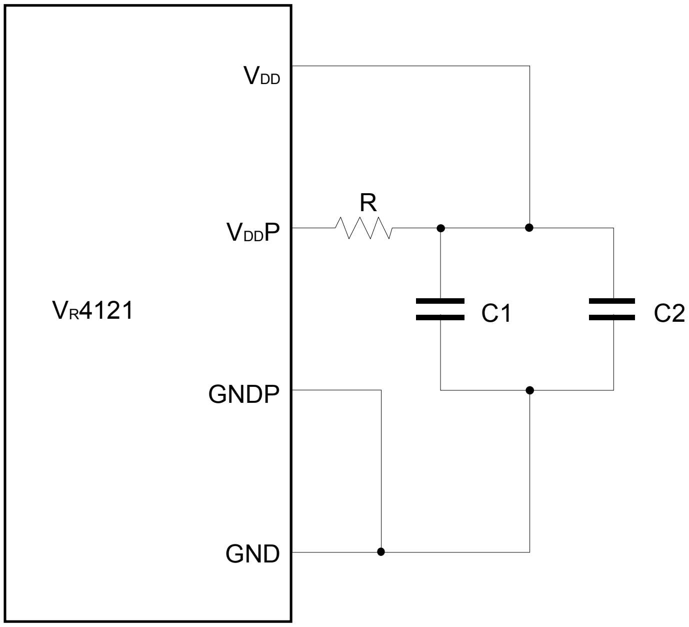
Note The number of hazards is undefined if the instruction execution sequence is changed by exceptions. In such a case, the minimum number of hazards until the IE bit value is confirmed may be the same as the maximum number of hazards until an interrupt request occurs that is pending and enabled.

[MEMO]

CHAPTER 31 PASSIVE COMPONENTS

The Phase Locked Loop circuit requires several passive components for proper operation, which are connected to $V_{DD}P$ as illustrated in Figure 31-1.

Figure 31-1. Example of Connection of PLL Passive Components



Remarks **1.** Capacitors C1 and C2 and resistor R are mounted on the printed circuit board.

2. Since the value for the components depends upon the application system, the optimum values for each system should be decided after repeated experimentation.

It is essential to isolate the analog power and ground for the PLL circuit ($V_{DD}P$) from the regular power and ground (V_{DD}/GND). Initial evaluations have yielded good results with the following values:

R = 100 Ω C1 = 0.1 *µ*F C2 = 1.0 *µ*F

Since the optimum values for the filter components depend upon the application and the system noise environment, these values should be considered as starting points for further experimentation within your specific application. In addition, the choke (inductor: *L*) can be considered for use as an alternative to the resistor (*R*) for use in filtering the power supply.

[MEMO]

INDEX

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
Technical Documentation Dept.
Fax: +49-211-6503-274

South America
NEC do Brasil S.A.
Fax: +55-11-6465-6829

Hong Kong, Philippines, Oceania
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

Korea
NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: 02-528-4411

Taiwan
NEC Electronics Taiwan Ltd.
Fax: 02-2719-5951

Asian Nations except Philippines
NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Japan
NEC Semiconductor Technical Hotline
Fax: 044-548-7900

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	❏	❏	❏	❏