Migrating RC32332/RC32334 Software to the RC32438 Device

RC32438 Application Note AN-368

Notes

By Harpinder Singh and Nebojsa Bjegovic

Introduction

Operating system kernels, board support packages, and other processor-aware system software created for systems based on RC3233x devices can be modified to execute in systems based on the RC32438, permitting significant code reuse. This application note offers specific guidelines to software developers undertaking such a porting effort. It is assumed that the individual undertaking such an effort is familiar with the architecture of both parts, since this note is intended to supplement existing technical documentation. The intent of this note is to offer the developer an opportunity to estimate the extent of the porting effort prior to starting. It may also serve as a checklist during the actual porting process.

Porting from RC3233x Devices to RC32438

The 79RC3233x components consist of the RISCore32300 CPU core and a number of peripheral interfaces, including an SDRAM memory controller, local memory controller, PCI interface, direct memory access (DMA) controller and a number of other generic peripherals including serial port(s), an interrupt controller, and timers. The RC32438 also incorporates a similar set of peripheral modules, some of which incorporate enhanced capabilities when compared to their counterparts in the RC3233x devices, as well as new modules not present in either of the RC3233x components. In addition, the CPU core used in the RC32438 integrated processor is the 4Kc core, based on the MIPS32 instruction set architecture (ISA), supplied by MIPS Technologies.

The table below summarizes the key feature differences between the RC32332, RC32334 and RC32438 integrated processors.

Feature	Integrated Processor		
	RC32438	RC32334	RC32332
Maximum CPU Frequency (MHz)	266	150	133
Theoretical Main Memory Bandwidth (MBps)	1064	300	266
On-chip Cache Size [I/D] (kB)	16/16	8/2	8/2
SDRAM Memory Controller	No	Yes (x32)	Yes (x32)
DDR Memory Controller	Yes (x32 / x16)	No	No
Dedicated Pins for Memory I/O Controller	Yes	No	No
PCI Bus Interface	Yes	Yes	Yes
Number of Bus Masters Supported by Arbiter	6	3	2
Sustained Bandwidth Across PCI (MBps)	160	40	40
Number of Ethernet Ports	2	0	0
I^2C Hardware Module	Yes	No	No

Table 1 Feature Comparison Between RC32438, RC32334, and RC32332 (Page 1 of 2)

Notes

Feature	Integrated Processor		
	RC32438	RC32334	RC32332
SPI Interface	Yes	Yes	Yes
Number of Serial Ports	2	2	1
Package Description	416BGA	256BGA	208QFP

Table 1 Feature Comparison Between RC32438, RC32334, and RC32332 (Page 2 of 2)

A major part of the software migration effort is expected to be related to "initialization" code, both for the CPU core and the peripheral interfaces. For the peripheral interfaces that were not present in the RC32332 or RC32334 devices, IDT will provide device drivers. IDT offers drivers for both VxWorks® and Linux operating systems and is also able to provide software written for its low-level monitor program (IDT SIM). In addition, IDT's products are supported by a number of other real-time operating systems (RTOS). For a complete list of supported operating environments, please contact your local IDT sales representative or visit IDT's web site at www.idt.com.

Guidelines to help porting software related to both the CPU core as well as the peripheral interfaces are covered within this application note.

CPU Core Initialization

Instruction Set Differences between RISCore32300 CPU Core and MIPS 4Kc

The primary differences between the two are in the following instructions:

- *RISCore32300 CPU Core does not include the SSNOP instruction*
- *Slight differences in the capabilities of the prefetch instruction*

The SSNOP instruction on RISCore32300 CPU Core is treated as a NOP because it is a single-issue processor. For Prefetch, the MIPS32 specification has a hint field for load, store, load_streamed, store_streamed, load_retained, store_retained writeback_invalidate, and other reserved hint fields. The RISCore32300 CPU core only implements a hint field for load, store, and ignore_hit.

Effectively therefore, the ISA in the 4Kc is a slightly larger superset from the ISA implemented in the RISCore32300, so there should be no issues for legacy code.

Privileged Resource Architecture

There are several differences between the CP0 registers of the RC3233x and RC32438 devices. However, the most frequently used bit fields in registers are located at the same locations within the same registers in both parts. If existing code follows the read/modify/write method for changing the contents of CP0 registers, and if the modifications to the bit settings are done using masks defined as macros in a single header file, code porting will be relatively straightforward since it will be primarily limited to the header file. Debug registers between the two parts are significantly different; however, this is only expected to impact a very small number of users including those developing EJTAG probes. Cache attributes are stored in a new register called CONFIG1.

Refer to Table 2 at the end of this application note for more specific information regarding the differences between register sets of RC3233x and RC32438.

Caches

Size: the RC32438 device incorporates 16kB each of data cache and instruction cache, as compared with 2kB data cache and 8kB instruction cache in the RC3233x devices. Any code written with hard-coded values of cache sizes will need to be modified. Any code written to employ intelligent algorithms to determine cache sizes may also be replaced completely with simple reading of the new CONFIG1 register in the RC32438.

Notes

Sets: The data cache in the RC32438 can be initialized in a single pass, as there is no address gap between associative sets. This is different from the RISCore32300 CPU core where a gap exists. Code for flushing or invalidating cache will need to address this difference as well. The caches in the RC32438 are 4-way-set-associative compared with the RC3233x caches, which are 2-way-set-associative. Therefore, the code will typically experience less cache thrashing in the RC32438. Any code specifically organized to minimize cache thrashing in the RC3233x parts will probably need to be reviewed to make it suitable for the caches in the RC32438.

Modes: For cache modes, the RC32438 only supports the "Write Through No Write Allocate" mode. However, the RC3233x devices support both "Write Through Write Allocate" and "Write Back" modes. Considering the fact that the highly integrated device interfaces within the RC32438 use DMA to transfer data to and from the CPU main memory in uncached space, the lack of a "Write Back" mode in the 79RC32438 is not seen as a huge handicap. However, code written with this mode in mind will need to be modified, which will result in simplified code. This is illustrated in the following example.

In the case of a device driver design that uses cached buffers for better performance, data handling will be different in the RC32438 compared to the RC3233x in the following areas:

Transmit

RC32438 device implementation: The device driver or the protocol stack prepares an outgoing packet using cached memory buffers and sets up a DMA channel to transmit the packet. Cache flush/invalidate operations are not required since the cache is "Write Through".

RC3233x devices implementation: The device driver or the protocol stack prepares the outgoing packet using cached buffers; but prior to setting up the DMA channel, the device driver is required to flush the cache to ascertain coherency with the memory. After the flush, the driver enables the DMA channel for transfer of the data.

Receive

For both the RC3233x and RC32438 devices, cache needs to be invalidated before processing the received packet in cached buffer mode, since the DMA engine always delivers the packet into uncached memory.

As seen above, the software design for the transmit path in the device drivers for the RC32438 integrated processor tends to be more efficient and simpler in the case of cached memory buffers.

Memory Management (TLB)

The 4Kc CPU core has a hidden bit, which flags a TLB entry for comparison only after that entry has been written to by software. On power-up or reset, none of the TLB entries will participate in memory mapping. Therefore, it is not necessary to initialize the TLB entries to a known state after reset. However, in the RC3233x devices, such an initialization is required.

Filling two entries with the same data generates a new Machine Check Exception in the RC32438, which prevents TLB shutdown. Change in cache policies impact the possible values for "cacheability" field in TLB entries in the RC32438.

Notes

Peripheral Initialization

Device Controller

The differences outlined below between the device controllers in the two parts will result in a few changes, primarily enhancements, to the initialization code:

- *Timing Control Register for programmable postread/postwrite and hold times. This is new in the RC32438.*
- *The Base and Mask registers in the RC32438 are in general more flexible and do not come up with fixed predefined values as in the case of the RC3233x.*
- *Decoupled access used for slow devices: Code for programming the device controller on the RC3233x components can be reused and can be enhanced to include the decoupled access feature offered by the RC32438 for slow devices. For decoupled read/write, you need to write to 3 registers to initialize transfer and then to read the device status register to check for a finished flag before initializing the next transfer. Multi-byte decoupled access must be within a word boundary.*

DDR SDRAM Controller

The RC32438 device supports 2GB of DDR SDRAM with 2 chip selects and 4 banks per chip select. Devices can be 8, 16, or 32 bits wide with densities of 64Mb, 128Mb, 256Mb, 512Mb, and 1Gb. New Alternate Base, Mask, and Map registers allow a board to boot via the PCI interface.

Interrupt Controller

Implementation of the interrupt controller is substantially different between the RC32438 device and the RC3233x devices. For the RC32438 device, interrupts from the integrated device interfaces are routed on distinct CPU interrupt lines IP[2-6], and a pair of interrupt pending/mask registers is allocated to each. This is different from the implementation in the RC3233x devices where a single line is multiplexed for a number of interrupts. Therefore, the interrupt handler, which de-multiplexes the interrupt sources, will need to be modified.

The RC32438 device provides an Interrupt Test Register to test the interrupt mechanism during software development. This should prove to be extremely valuable to a large majority of software developers.

Interrupts in the RC32438 device have a much higher priority (right after resets, EJTAG and Machine Check) than in the 79RC3233x devices (lowest). This allows interrupts to have precedence in being serviced, resulting in improved performance for time-critical applications. Some of the handler logic may be affected by this change in priority. TLB exceptions now have lower priority than interrupts and may require changes in the handling of TLB exceptions if TLB is used.

Due to these changes, this is an area where the best approach is probably one of rewriting new code optimized for the RC32438 device, rather than porting legacy code.

Timer Controller

The main difference between the two devices is in the control register, which now has a timeout bit used to clear an interrupt. Code written for the RC3233x devices is reusable for the RC32438 as long as this change is addressed.

UART Controller

No code change required.

GPIO Controller

The GPIO controller in the RC32438 device is the same as in the RC3233x devices with the following exceptions:

- *Different pin assignments for alternate functions*
- *Increased number of pins in the RC32438 (32 instead of 16), and with a*
- *New feature in the RC32438 whereby any GPIO pin, configured as an input, can generate an NMI if enabled.*

Notes

The difference in alternate functions can typically be addressed by making small changes to a header file if the original code is relatively well written and devoid of hard-coded values within the source code.

DMA Controller

The RC32438 device includes 10 dedicated DMA channels assigned as follows: Two for the PCI interface, 4 for the Ethernet interfaces, 2 for external peripherals, and 2 for memory-to-memory transactions. The RC3233x devices include 4 DMA channels. In addition, the RC32438 device provides the concept of DMA chaining that is not present in the RC3233x.

The DMA engine in the RC32438 device is substantially different and advanced compared to that in the RC3233x parts. Rewriting the code for the RC32438 will be prudent.

PCI Controller

The PCI bus interface has been enhanced in the RC32438 device, incorporating several new features. The PCI interface included in the RC32438 device has the capability of sustaining significantly higher throughput than in the RC3233x devices. Any code specifically fine-tuned for the RC3233x will probably have to be revisited if porting that code to the RC32438.

Even though the RC32438 device offers a significantly advanced PCI engine as compared to the RC3233x devices, there are enough similarities between the two architectures to allow an easy port of the PCI-related code. Some software modules, such as Messaging and decoupled CPU PCI master transactions, will need to be written from scratch in order to benefit from enhancements. Some of the other enhanced features that could be added to code during the porting effort are:

Arbiter parking: Allows parking the bus on the last master granted the bus.

Idle Grant Mode: Provides static and dynamic grant modes. In static grant mode, once a grant is asserted when the PCI bus is idle, the grant will remain asserted until the requested transaction completes. When using dynamic idle grant mode, when the PCI bus is idle, the arbiter may take away the grant from one master and pass it to another.

PCI operating modes selected at boot time are similar to RC3233x devices but with slightly different options. Code may need to be changed to accommodate these operations. PCI Base Address Control and Mapping registers are part of the PCI configuration register space, allowing the host to read/modify based on system-specific requirements. PCI configuration registers have a PCI Management register that can be used to reset or raise NMI exception to the CPU. PCI address space can also be mapped in Kseg1/Kseg0, requiring no memory mapping through TLB.

Ethernet Controller

The RC32438 device includes two identical and independent Ethernet interfaces with associated MAC and MII pin level interface. Since this interface did not exist in the RC3233x, code porting is not possible. IDT will provide sample Ethernet drivers for a few operating systems.

I^2C Interface

The I^2C interface is also new in the RC32438 device, with support for both Master and Slave modes. It complies with the I^2C bus specification v2.0. IDT will provide sample I^2C device drivers for a few operating systems.

SPI Interface

The SPI interface in the RC32438 device is slightly different from the module included in the RC3233x devices. In the RC32438, the chip select signal needs to be explicitly generated at the beginning of a transaction using one of the GPIO pins. Additionally, MICROWIRE transactions can be generated using the new dedicated registers in the RC32438. SPI code written for the RC3233x devices is largely reusable for the RC32438, with the above considerations.

Notes

On-Chip Memory

The RC32438 device includes a 4kB On-Chip memory module organized as 1k x 32 bits, which can be mapped using the TLB. This memory may be used for IPBus Monitor recordings, for storing intermediate results from application code, or as general-purpose memory.

Debugging And Performance Monitoring

The IPBus Monitor, a new feature in the RC32438 device, provides an on-chip "logic analyzer" for hardware and software enhanced debugging of transactions on the IPBus. It consists of 24-bit statistics counters, triggers, and filters. Recorded transactions are stored in On-Chip Memory in which data is not reset during a warm reset. Any code to take advantage of this new feature or to aid in the debug process will need to be written by the user, although there is a high probability that IDT may provide some low-level code along with an API that software developers may be able to use.

EJTAG/ICE Interface

The 4Kc CPU core incorporates an in-circuit emulator interface compatible with the EJTAG version 2.5 specification. The RISCore32300 supports version 1.5.3 of the EJTAG specification. A new CPU core register file specific to the 4Kc core must be used for existing EJTAG probes, after the probes are made compatible with version 2.5. Version 2.5 of the specification also includes slightly different implementation and control register bit fields.

Summary

The RC3233x and RC32438 integrated communications processors include a similar set of high level features. The strong compatibility between the CPU cores in each of the devices, and the substantial reuse of existing peripheral modules between the two devices, makes the software transition to the RC32438 device a straightforward process.

Features	Differences	
	RC32438	RC32334/RC32332
CP0 Registers		
Status (12)	27 - Reduced power mode 24 - Reserved (0) 23 - Reserved (0) 21 - TLB shut-down (TS) 19 - NMI 17 - Reserved (0) 16 - Reserved (0)	27 - Reserved (0) 24 - D-cache lock enable 23 - I-cache lock enable 21 - Reserved (0) 19 - Reserved (0) 17 - CE bit. When set, ECC contents modify check bit of the caches 16 - Cache parity exception enable
Cause (13)	26 - Reserved (0) 25 - Reserved (0) 24 - Reserved (0) 22 - WP: Watch exception 6:2 - New exception code (=24) called Machine check	26 - IPE, imprecise exception indicator 25 - DWatch matched 24 - IWatch matched 22 - Reserved (0)

Table 2 CP0 Registers: Bit level differences between RC3233x and RC32438 (Page 1 of 3)

Notes

Features	Differences	
	RC32438	RC32334/RC32332
Config (16)	31 - Presence of Config1 (1) 30:28 - Reserved (011) 27:25 - Reserved (011) 23 - Reserved (0) 18:17 - Merge mode 16 - Burst order 14:13 - Reserved (00) 11:10 - Reserved (00) 9:7 - MMU Type (001) 6 - Reserved (0) 5 - Reserved (0) 4 - Reserved (0)	31 - ICE present 30:28 - External clock multiplier 27:25 - Reserved (000) 23 - Non-blocking load pending 18:17 - Reserved (01) 16 - Reserved (0) 14:13 - Reserved (11) 11:9 - I-cache size (100) 8:6 - D-cache size (010) 5 - I-cache line size (0) 4 - D-cache line size (0)
Config1 (16) select 1	30:25 - MMU size (001111) 24:22 - I-cache sets per way (010) 21:19 - I-cache line size (011) 18:16 - I-cache associativity (011) 15:13 - D-cache sets per way(010) 30:25 - MMU size (001111) 24:22 - I-cache sets per way (010) 21:19 - I-cache line size (011) 18:16 - I-cache associativity (011) 15:13 - D-cache sets per way (010) 12:10 - D-cache line size (011) 9:7 - D-cache associativity (011) 4:0 - Miscellaneous (01010)	NA
LLAddr (17)	27:0 - Physical address read by the most recent Load Linked instruction (Diagnostics purpose only)	NA
IWatch (18)	NA	31:2 - Instruction Virtual address that causes a Watch exception 0 - Instruction Watch enable
DWatch (19)	NA	31:3 - Data Virtual address that causes a Watch exception 2 - Data Watch enable for loads 1 - Data Watch enable for stores
WatchLo (18)	31:3 - Virtual address that causes a Watch exception 2 - Instruction Watch enable 1 - Data Watch enable for loads 0 - Data Watch enable for stores	NA
WatchHi (19)	30 - Global 23:16 - ASID 11:3 - Bit Mask address	NA
DebugEPC (23)	For RC32438, this is register #24	

Table 2 CP0 Registers: Bit level differences between RC3233x and RC32438 (Page 2 of 3)

Notes

Features	Differences	
	RC32438	RC32334/RC32332
Debug (24)	For RC32438, this is register #23 28 - Control access between dseg and remaining memory 27 - Indicates Low power mode when exception occurs 26 - Indicates Internal system bus clock stopped when exception occurs 25 - Indicates behavior of Count register in debug mode 24 - Instruction fetch Bus Error exception pending 21 - Data Bus Error exception Pending 20 - Imprecise Error exception Inhibit 17:15 - EJTAG version 14:10 - Exception Code (as is in Cause) 8 - Debug single step enabled 7 - Reserved (0) 6 - Reserved (0) 0 - Debug single step occurred	 28 - Reserved (0) 27 - Reserved (0) 26 - Reserved (0) 25 - Reserved (0) 24 - Reserved (0) 21 - Reserved (0) 20 - Reserved (0) 17 - Reserved (0) 16 - Interrupt when Cause.IV set 15 - Cache error status 14 - NMI status 13 - TLB refill miss status 12 - Other exception status 11 - TLB exception flag 10 - Bus error exception flag 8 - Reserved (0) 7 - JTAG reset 6 - Debug boot bit 0 - Reserved (0)
ECC (26)	NA	7:0 - Parity bits in primary cache
CacheErr (27)	NA	31 - Instruction or data 30 - Primary cache 29 - Data field error 28 - Tag field error 25 - Instruction AND data error 21:3 - Physical address 0:1 - Virtual address
TagLo (28)	31:10 - PA (31:10) 7:4 - Valid 1 - LRF 0 - Reserved (0)	30:8 - If D-cache PA (31:9) If I-cache PA (31:11) 7:6 - Cache state 5:4 - Reserved (00) 1 - FIFO refill 0 - Tag even parity
DataLo (28) select 1	31:0 - Data Read only for diagnostic purposes	NA
DeSave (31)		Implemented as register at 0xFFFF_E210

Table 2 CP0 Registers: Bit level differences between RC3233x and RC32438 (Page 3 of 3)

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.