Compatibility Between RC32334/ RC32332 and RC32364/RC32134 Based Solutions

Technical Note
TN-47

Notes

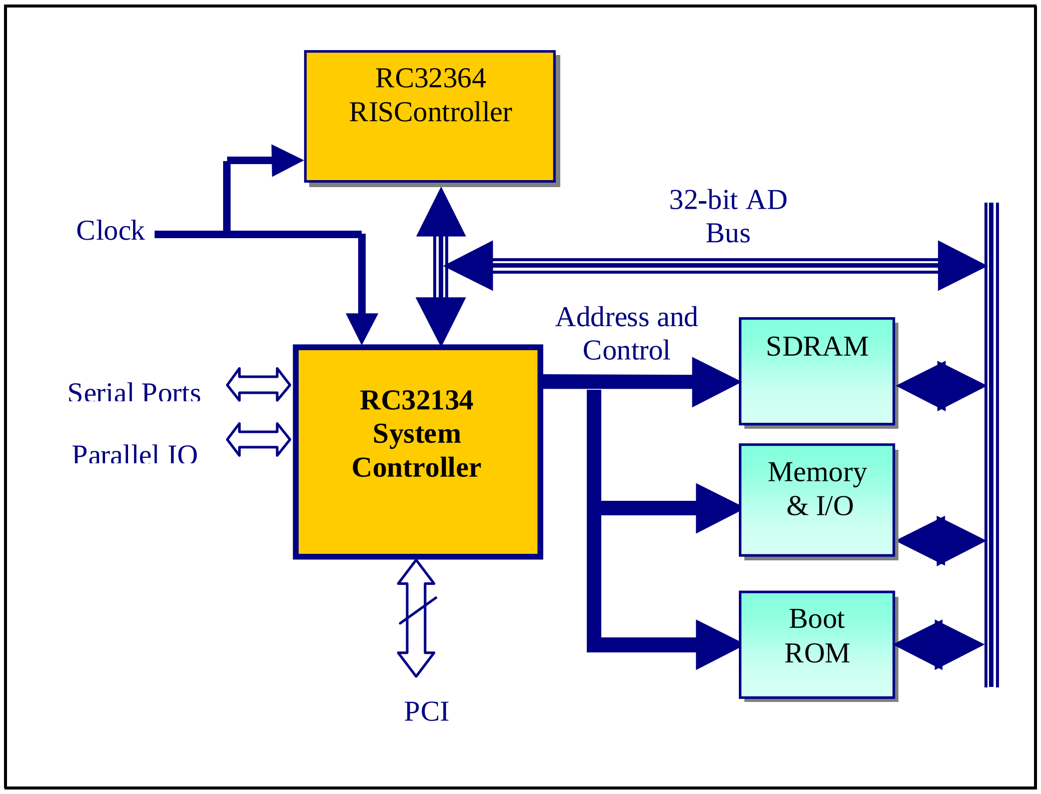
By Ian Ferguson

Introduction

Integrated Device Technology has developed a series of RISC processors to target embedded communications applications. All implement the MIPS Instruction Set Architecture (ISA). One of these devices, the RC32364, is a standalone CPU which is based on IDT's RC32300 CPU core. This implements the MIPS32 ISA. A companion device, the RC32134, incorporates a number of interfaces to other functions required in the majority of embedded applications. These include

- **PCI Bridge, fully compatible with version 2.1 of the PCI specification and operational up to 33MHz, with built in arbitration logic for managing PCI bus master device**
- **Flexible memory controller supporting 8, 16 or 32-bit external memory devices such as ROM, Flash, SDRAM and EDO**
- **Two 16550 compatible serial ports**
- **Four DMA channels**
- **Three 32-bit timers**
- **Interrupt controller providing the interrupt logic to enable software to identify the peripheral asserting a system interrupt**
- **Programmable input/output pins**

The diagram below shows how a basic system using the two devices described above could look.

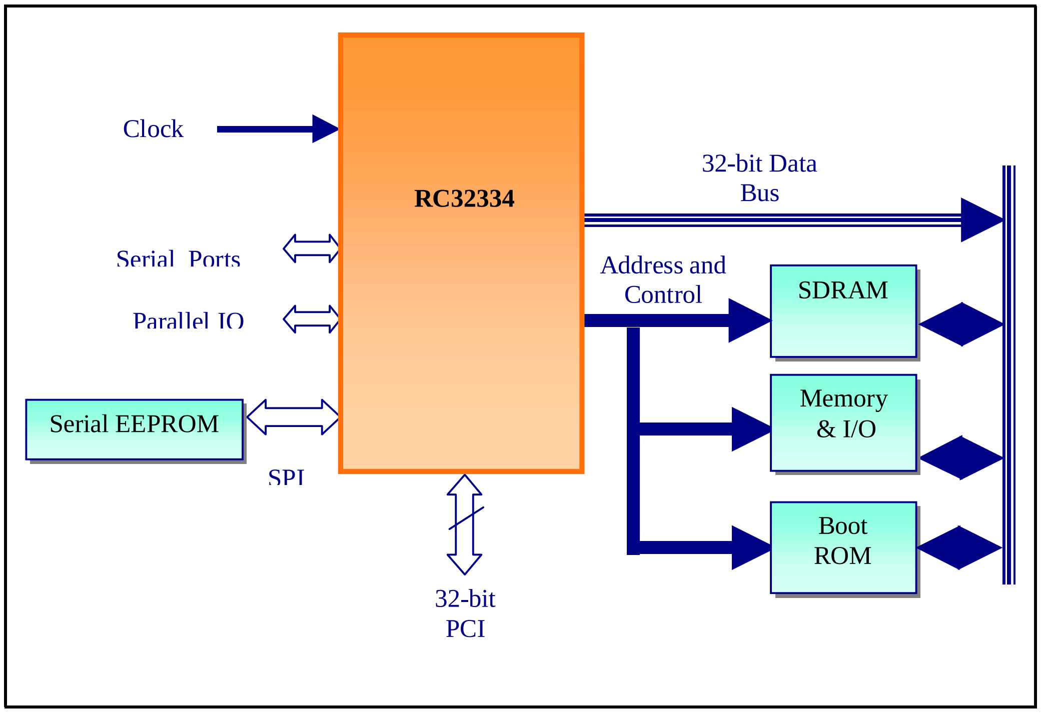


DSC 5916

Notes

The RC32334/RC32332 is an integrated processor that integrates the RC32300 CPU core, identical to the one present in the RC32364 and the peripheral modules found in the RC32134 system controller onto a single piece of silicon.

The diagram below shows a simple system implementation of a RC32334-based platform.



Therefore there is a large degree of compatibility between the discrete two-chip solution and the integrated processor. The aim of this document is to provide information about features where there are minor hardware and software differences between the solutions, and provide some guidelines as to the impact on the system designer and software programmer. The areas of differences are grouped below by the interface module affected.

External Memory Support

SO-DIMM Support

In the RC32334/RC32332, the SDRAM controller has been modified to allow direct connection with SODIMM-144.

A register bit that was previously reserved in RC32134 is used as a control bit. The setting of bit 15 in the SDRAM control register will enable SO-DIMM mode. Logically, the existing SDRAM chip select signals are OR'd onto two new pins. The existing chip selects are then used for the odd chip select byte masks. In this case, the SDRAM signals should be connected to the SO-DIMM pins as follows:-

- **sdram_s_n[1] corresponds to banks 3&2.**
- **sdram_s_n[0] corresponds to banks 1&0.**
- **sdram_cs_n[3:0] correspond to the byte enable DQM signals for banks 3&1.**
- **sdram_bemask_n[3:0] correspond to the byte enable DQM signals for banks 2&0.**

Impact: The SDRAM controller included on the RC32334/RC32332 is backwards compatible to the RC32134 but adds SO-DIMM support as an additional feature. Customers wishing to reuse standard DIMM modules will be able to reuse code and schematics developed for the RC32134-based system.

Notes

EDO Support

The RC32134 system controller supports EDO memories directly. The RC32334/RC32332 does not support EDO memory subsystems.

Impact: If the RC32364/RC32134 was designed to support an EDO memory subsystem, then both the hardware and the associated code will need to be rewritten for an SDRAM-based memory subsystem. In the RC32134, the registers that provide address and mask information for the four banks of memory are shared. The primary change will be the use of the SDRAM control register (address 1800_0300) to configure the memory subsystem, including SDRAM bank size, CAS latency and number of pre-charge clocks required. Refer to chapter 11 of the RC32334/RC32332 User Reference manual for more information.

Memory Output Clock

The RC32334/RC32332 allows the capability to drive a general purpose output clock (output_clk). This is enabled by setting bit 14 in the SDRAM control register. If the bit is set to 0, the output_clk pin is tri-stated. By default the pin is enabled. IDT recommends that this pin always be disabled by resetting this bit in the control register as part of the device initialization sequence, regardless of the existence of SDRAM in the system.

Impact: Users of the RC32334/RC32332 should ensure the device initialization code includes the resetting of the bit to tri-state this pin

Error Handling of Accesses to disabled DRAM locations

In the RC32134, if a DRAM memory access is made to a disabled DRAM area, the cycle will eventually timeout and a watchdog timer interrupt will be generated to the CPU. However, in the RC32334/RC32332, such an access will generate a bus error to the CPU.

Impact: The interrupt handler routine written for RC32134-based systems to manage this event will need to be rewritten to reflect the different method that the CPU is informed of such an event.

Operation of SDRAM interface at 75MHz

If the SDRAM memory subsystem is to be run at 75MHz, the address latch timing register included in the RC32334 needs to be set to 2 clocks (this does not apply to the RC32332 which supports system bus speeds up to 66MHz). In the RC32134, the same register would be set to a value of one clock to support the same 75MHz bus interface speed.

Impact: The initialization code programming the address latch timing register system will need to be modified if 75MHz bus operation is required. See Chapter 8 of the RC32334/RC32332 user manual for more information.

PCI Interface

66MHz PCI Bus Operation

The RC32334 PCI bus interface can be operated up to a maximum 66MHz clock frequency (50MHz for the RC32332). The RC32134 PCI bus has a maximum specified frequency of 33MHz

Impact: No impact. All PCI boards running in existing RC32134-based PCI subsystems at 33MHz will operate in a RC32334-based system, as the RC32334/RC32332 supports the lower bus frequency. This feature allows the possibility to connect up higher performance PCI-based cards to improve overall system bandwidth.

66MHz PCI Configuration Bit

When the PCI bridge is configured, the PCI target not ready bit (bit 2) in the PCI arbitration register is set. In the RC32334, either the RC32334 or an external PCI-based device configuring the PCI bus can set a bit in the PCI Status register (bit 5) to indicate that the bus is capable of operating at 66MHz (50MHz for the RC32332).

Notes

Impact: In the RC32134, this bit was not writeable. As a new feature, this feature has no impact on legacy code. It enables new application level code to be able to determine whether the PCI bus can be run at maximum speed, by accessing a single bit.

PCI_EEPROM_CS Signal

The RC32334/RC32332 includes a dedicated PIO pin for the pci_eeprom cs signal. This is used to enable an external EEPRROM to configure the PCI registers of the RC32334/RC32332, when it is to be used as a PCI slave. On the RC32134, the chip select was provided on a PIO pin that was shared with the pci_gnt_n[1] signal.

On the RC32334/RC32332, the chip select bit is located in a second PIO register, at physical address 0x18000610. The bit providing this function is bit 0.

Impact: Software written for the RC32134 that relies on the on-chip PCI bridge to be configured by an external EEPROM will need to be modified slightly, to program the new PIO bit that controls the chip select signal

PCI System Error Pin

This signal is used to inform the PCI host of an address error. Note that the RC32334/RC32332 does not connect the pci_serr_n signal inside the package to the RC32300 CPU core. Users will need to route this signal to either the non maskable interrupt signal (cpu_nmi_n) or a high priority interrupt signal.

Impact: This implementation is identical to the RC32134. The rationale for highlighting it here is that even though the RC32334/RC32332 integrates both the CPU core and the PCI bridge onto a single piece of silicon, this signal is not connected internally to the CPU core.

Handling of PCI Host Mode Vendor ID Polling Error

If any PCI cycle ends in error a data value of 0xffffffff will be returned to the PCI host. This is turn will generate a PCI Bus Error, which in turn is used to assert a Bus Error signal to the CPU itself. This signal is external on a RC32134 system implementation and internal to the RC32334/RC32332. The system includes an exception handler routine to cope with this event. However, during the initialization phase of the PCI interface, the PCI host (RC32134 or RC32334/RC32332) will typically scan the PCI bus to determine which cards are present in the system. When a slot is not populated with a target, a config_read cycle will return a data value of 0xffffffff. In the RC32334/RC32332, a new bit has been specified in the Bus Error Control register which allows Bus Error exceptions to be masked from generating the exception to the CPU Core. By setting this bit prior to scanning the PCI bus, the BIOS software can avoid special exception handler software for this period. Once the scan is completed, the bit can be reset, to ensure that bus errors that do occur during normal PCI bus transactions are recognized by the CPU.

Impact: This bit has been defined to be 100% backwards compatible with the RC32134 system controller. That is, the default value for this bit (0) in the RC32334/RC32332 register has BusError enabled. Therefore existing initialization routines accessing this register will not need to be modified for the RC32334/RC32332. The new feature can be used to simplify the startup code.

Illegal Disabling of Enable Bit in Configuration Address Register

It is possible that code originally developed for PCI peripherals for personal computer type environments will try to disable the enable bit in the PCI Configuration Address Register (as there is no analogous PC-AT I/O address space in the MIPS architecture). On the RC32134, the BusError exception handler was required to run a dummy cycle internally to clear the condition. In the RC32334/RC32332, the above situation will cause the PCI slave state machine to be reset.

Impact: The exception handler written to workaround this condition for the RC32134 can be removed for RC32334 and RC32332-based systems

PCI Vendor ID and Device ID Differences

For systems booting from local memory, there are some minor differences in the values programmed into the Device ID and PCI vendor ID registers:

Notes

RC32334/RC32332 device ID is 0x0204. RC32134 device ID set at 0x0000

RC32334/RC32332 vendor ID is set to 0x111D. In the RC32134, the PCI vendor ID is 0x0000.

Impact: Any application code that uses these registers will need to be updated to reflect the codes programmed for the RC32334/RC32332.

PCI DUAL ADDRESS MODE DISABLED

In the RC32334/RC32332, dual Address bits [7:0] hardwired to 0x00. Dual Address Mode configuration Type bit for Memory Base Address Register Bit 2 is set to zero which disables this function.

Impact: These settings for the RC32334/RC32332 are consistent with the recommendations for the RC32134. No impact for legacy code.

PCI I/O SPACE MAPPING TO LEGACY PC DEVICES

In the RC32334/RC32332, CPU to PCI accesses are able to address the 0x0000_0000 address space of legacy PC devices. Change CPU to PCI I/O mapping register from 4-bits to 12-bits.

Impact: Current RC32134 users should write 0x-88---- to the PCI IO register for forward compatibility.

PCI TARGET 0X0 ADDRESS DECODING

The RC32334/RC32332 allows conventional debug and non-PC system usage of address space. On PCI Target accesses to memory and I/O, 0x0xxx_xxxx and 0x0000_00xx spaces are decoded.

Impact: Code previously written for the RC32134 will run unmodified on the RC32334/RC32332. This new feature provides an opportunity for customers to rewrite certain portions of code to simplify their debugging process and use of the 0x0 address and IO spaces.

Serial Ports

Modem Signals on UART0

Both the RC32334 integrated processor and the RC32134 device include two 16550 compatible UARTS. However for UART0 on the RC32334, four modem signals are mapped out to the external pins. (Note: the RC32332 has only 1 UART and does not support mapping to external pins.)

Signal Name	Description	PIO Pin
uart_dtr_n[0]	Data Terminal Ready	PIO[13]
uart_dsr_n[0]	Data Set Ready	PIO[14]
uart_rts_n[0]	Request to Send	PIO[12]
uart_cts_n[0]	Clear to Send	PIO[15]

The above signals default to PIO inputs at reset. The programmer will need to configure these bits through the programming of the appropriate fields in both the PIO direction register 1 and PIO data register 1 to select this signals for use with the UART. For more information, refer to the Programmable I/O Controller section in Chapter 15 of the RC32334/RC32332 user manual.

Impact: In the RC32134, the RTS and DTR bits in the Modem Control Register and the DSR and CTS bits in the modem status register could be ignored as they had no effect on external pins. For the RC32334/RC32332, care must be taken to ensure that bits are programmed appropriately.

UART Hardware Reset

The RC32334 provides the programmer the capability of generating a hardware reset to each UART channel. Specifically the following addresses should be used to reset each UART.

Notes

UART0	Address 0x1800_0840
UART1	Address 0x1800_0860

Note: The RC32332 does not include UART1.

Impact: This is a new feature, not previously available on the RC32134, so there is no impact on the porting of legacy code.

Serial Peripheral Interface

Addition of Serial Peripheral Interface (SPI) on RC32334

The RC32334/RC32332 includes an additional industry standard serial interface that is not included in the RC32134. External peripherals that support this bus standard include EEPROM and Analog to digital (A-D) converters. In a similar way to the modem control signals described previously, these pins default to PIO lines at reset (spi_miso defaulting to an input, the other three defaulting to be outputs). The table below shows the correlation between SPI pin functions and PIO pin values. The PIO direction register 0 and data register 0 need to be configured to setup these pins to operate as the SPI interface.

Signal Name	Description	PIO Pin
spi_mosi	SPI Data Output	PIO[10]
spi_miso	SPI Data Input	PIO[7]
spi_clk	SPI Clock	PIO[9]
spi_ss_n	SPI Chip select	PIO[8]

Impact: This is a new feature not incorporated in the RC32134. Any code written for the RC32134 that uses these pins as PIO will run unmodified. There is an additional set of registers used to control the four wire SPI interface. Refer to Chapter 18 of the RC32334/RC32332 user manual for more information

Local System Bus

RC32334/RC32332 External Bus Interface

The integrated core controller includes five slight differences to external system bus of the discrete solution

1. The external memory addressing has been extended to support a full 64Mbyte decode per chip select. Memory address bits 25:23 are now accessible to external system logic
2. The bus interface for the RC32334/RC32332 features de-multiplexed address and data busses. A separate 32-bit data bus is therefore present on the RC32334/RC32332.
3. With the exception of cpu_coldreset_n and cpu_dt_r_n signals, none of the other controls signals present on the RC32364 bus interface are accessible on the RC32334/RC32332.
4. On the RC32334/RC32332, the timer_tc_n[1] pin is not present.
5. The RC32334/RC32332 provides I/O mode burst reads address hold time after each datum. IOI and IOM burst reads hold the address for 1 more clock providing 1.0 clock address hold time after each datum is read.

Impact: The designer developing a system based around the RC32334/RC32332 will need to adjust any external hardware affected by the revised signals mapped out to the external pads of the RC32334.

Addressing of 8-bit and 16-bit External Memory Peripherals

The RC32334/RC32332 uses different signals to provide least significant address bits 1:0 on 8-bit and 16-bit port width accesses.

These specific signals used for this purpose are shown in the table below

Notes

Port Width	mem_we_n[3]	mem_we_n[2]	mem_we_n[1]	mem_we_n[0]
DMA (32-bit)	mem_we_n[3]	mem_we_n[2]	mem_we_n[1]	mem_we_n[0]
32-bit	mem_we_n[3]	mem_we_n[2]	mem_we_n[1]	mem_we_n[0]
16-bit	Byte High WrEn	mem_addr[1]	Not Used (lo)	Byte Low WrEn
8-bit	Not Used (hi)	mem_addr[1]	mem_addr[0]	Byte WrEn

Impact: For 8-bit and 16-bit peripherals, circuitry will need to be reworked to use the above signals to control the lower address bits.

DMA

DMA Bus Turnaround (BTA) Optimization

In the RC3134, BTA clock cycles were inserted whether they were required by the system or not. In the RC32334/RC32332, BTA clock cycles will only asserted if the DMA write after the DMA read is going to take less than the programmed BTA value.

Impact: No impact on existing code as it will operate unmodified. However, this feature allows the possibility to improve the performance of the system

DMA Descriptor Read BTA

In the RC32334/RC32332, after a DMA descriptor burst read, no Bus Turnaround (BTA) clocks are inserted. Consequently a CPU address may appear on the mem_data[] bus as soon as two clocks after the DMA descriptor read.

Impact: DMA descriptor memory buffers must reside in memory/ or transceivers which do not have critical BTA of more than 2.0 clocks.

DMA Done

In the RC32334/RC32332, the dma_done_n input signal is synchronized and held. After the DMA transaction in progress completes, the DMA channel is disabled and a newly defined interrupt (DMA Interrupt #3) is asserted. The DMA interrupt #3 is still asserted even if the DMA channel simultaneously completes or is disabled. The internal pending dma_done_n synchronization register is reset when the DMA channel is re-enabled.

Impact: The interrupt handler written for the DMA channels 0 and 1 of the RC32134 will need to be rewritten to take account of the new internal interrupt 3 which has been defined. For more information, refer to Chapter 13 of the RC32334/RC32332 user manual.

DMA Clear Enable Channel

The RC32334/RC32332 adds a new dma_clr_en0 interrupt. This was implemented as previously application software was not aware how long to wait until channel was completely disabled. This new interrupt #4 is asserted for each channel when DMA channel completes, including flushing the FIFO of a final transfer.

Impact: The interrupt handler written for the DMA channels 0 and 1 of the RC32134 will need to be rewritten to take account of the new internal interrupt 4 which has been defined. For more information, refer to Chapter 13 of the RC32334/RC32332 user manual.

Notes

JTAG/Debug Interface

EJTAG/JTAG Controllers on RC32334/RC32332

When the RC32364 and the RC32134 devices were integrated together to form the RC32334/RC32332 integrated processor, the implementation kept both TAP controllers, one in the CPU core and one for the system control logic. For the TAP controller in the RC32300 CPU core, only the BYPASS, EJTAG, IDCODE instructions can be used. The TAP controller present in the system controller logic should be used to perform the boundary scan function, and execute instructions like EXTEST AND SAMPLE/PRELOAD. Also note that the RC32300 TAP controller and the TAP in the system controller logic cannot scan out commands or data at the same time since they share the JTAG TDO output pin.

Impact: The JTAG boundary scan test patterns will need to be rewritten to support the RC32334/RC32332.

JTAG IDCODE Change

The RC32334/RC32332 includes a unique part number of 0x18. This compares with the RC32134, which has a part number value of 6h.

Impact: JTAG boundary scan test patterns will need to be updated to reflect the RC32334's part number value that will be returned when the Device ID register is read.

JTAG Clamp Instruction

The RC32334/RC32332 implements the Clamp instruction, defined as an optional instruction in the JTAG specification. This allows the user the option to bypass the on-chip JTAG controller to access further devices down the JTAG chain while keeping the outputs on the RC32334/RC32332 constant.

Impact: No impact to customers migrating JTAG code from a RC32364/RC32134 system. This is a new feature which can be used if it is desired to improve the efficiency of the JTAG test patterns.

Parallel IO Pins

PIO Registers

The RC32334/RC32332 includes four additional signal pins that can be used as general-purpose parallel IO signals. These are the four modem signals associated with UART0. Architecturally, the control of these four signals (as well as the pci_eeprom_cs signals described in section 2.3) reside in three new PIO registers.

Register	Effective Address	Function
PIO Data Register 1	1800_0610	Transmits/Receives data from external PIO pins
PIO Direction Register 1	1800_0614	Programs individual PIO signals as an input or output
PIO Function Register 1	1800_0618	Define data generation for PIO lines configured as outputs as either from data register or internal module

Impact: No impact on legacy code.

Use of PIO lines as interrupt sources

Similar to the RC32134, the RC32334/RC32332 allows the PIO signals to be configured as capable of initiating an interrupt. The PIO signals are aggregated into register groups inside the interrupt controller. In both the RC32134 and the RC32334/RC32332, when an interrupt is generated, the controller will first identify which register group asserted an interrupt, and will then access the specific register associated with that grouping in order to determine the exact signal that generated the interrupt request. Note that only PIO signals 11:0 can generate an active low interrupt and 6:0 can generate an active high interrupt request. The PIO lines are located in register groups 2 and 3. However, note that there are minor differences between

Notes

the assignments of the PIO signals inside these specific registers. The RC32334/RC32332 includes bit 2 in both registers as a reserved bit. Consequently bits 3 and above in the RC32334/RC32332 are displaced by one when compared with the RC32134 implementation. For example, in the RC32334, bit 3 represents PIO2, while in the RC32134 the same bit represents PIO3. Bit 4 in the RC32334 represents PIO3, while in the RC32134 it represents PIO4.

Impact: For systems using PIO signals to generate interrupts, the interrupt handler code itself will need to be rewritten

Miscellaneous

CPU CP0 Processor Revision Identifier (PRID) Value

The PRId register (CP0, register 15) contains a unique value representing the CPU core implementation level. For the RC32334/RC32332, the value programmed in the 8-bit field (bits 15:8) is 0x18. For the RC32364, this value 26h.

Impact: Any software previously written for the RC32364 that interrogates and uses information read from this register will need to be modified to operate in a RC32334 and RC32332-based system.

Pin Assignments for Mode Bits and Mode Bit Functionality on RC32334/RC32332

The RC32300 CPU core that is included in both the RC32364 and the RC32334/RC32332 processors, uses the state of certain pins to provide some basic configuration information and enable it to then start to read in code from external memory. For example, the pipeline clock to system clock ratio is determined at this time, along with the endianess of the system and the port width of the memory space that contains the initialization code. The removal of certain CPU signals present on the RC32364 has

Mode Bit	Description	RC32364 Pin Assignment	RC32334/32 Pin Assignment
2:0	Pipeline Clock/System bus clock multi-plier	PCST[2:0]	ejtag_pcst{2:0]
3	Little/Big Endian byte ordering	PCST3	debug_cpu_i_d_n
4	Reserved	PCST4	debug_cpu_ack_n
5	Reserved	BusGNT	debug_cpu_ads_n
6	Enable Timer interrupt	NT0	debug_cpu_dma_n
7	Reserved	INT1	mem_addr[17]
9:8	Boot-PROM width	INT[3:2]	mem_addr[19:18]

The RC32334/RC32332 uses mem_addr{22:20] is the same way as the RC32134, to define whether the memory device containing the boot code is managed by the on-chip memory controller or by the PCI bridge.

Impact: Users need to ensure that the pins listed above are pull high or low via resistors, to configure the system appropriately on reset.

Bus Interface Unit (BIU) System ID Register

IDT defines a unique ID for each system controller it manufactures and programs this value in the bit field 19:8 in the SysID register. This allows software to recognize the type of system controller being used to then determine the appropriate initialization code is used. The RC32334/RC32332 programs SYS ID register (address location 0x1800_0018) to a value 0x0000_2000. The RC32134 is programmed with 0x000. Note that this register is a read only.

Notes

Impact: If current code written for the RC32134 reads in this value to determine further programming steps that are needed, this code will need to modified to recognize the value programmed into this field for the RC32334/RC32332 integrated processor.