

MMU and Coprocessor specification

CoProcessor Specification

All registers are RW except where otherwise noted (RO or WO). Items in *italics* are indicated and are provided as placeholders only and are not implemented. Items in *italic-bold* indicate that they are only implemented in the software simulator. Red indicates not implemented yet. Blue indicates implemented, but not verified. Green indicates implemented with test-code. White indicates un-implemented, and should respond as a bounced coprocessor instruction.

Table 1: CoProcessor Register Summary

| R# | CoProcessor 13 | CoProcessor 14 | CoProcessor 15 |
|----|------------------------------------|------------------------------|-------------------------------|
| 0 | Access Cycle Count (RO) | Real Time Counter Low (RO) | MMU ID (RO) |
| 1 | Idle Cycle Count (RO) | Real Time Counter High (RO) | System Control |
| 2 | Sleep Cycle Count (RO) | Timer Interrupt | <i>Translation Table Base</i> |
| 3 | Wait Cycles (RO) | Interrupt Suspend | <i>Domain Access Control</i> |
| 4 | Hit Count (RO) | Internal Dynamic Clock Speed | |
| 5 | Cached Miss Count (RO) | External Pin Control | <i>Fault Status</i> |
| 6 | Cache Writeback Count (RO) | Hw. Control Tweaks | <i>Fault Address</i> |
| 7 | Uncached Access Count (RO) | Instruction Count (RO) | Cache Operations (WO) |
| 8 | <i>External Idle Count (RO)</i> | | <i>TLB Operations</i> |
| 9 | <i>External Address Count (RO)</i> | | |
| 10 | <i>External Data Count (RO)</i> | | |
| 11 | <i>External Wait Count (RO)</i> | | |
| 12 | <i>External Other Count (RO)</i> | | |
| 13 | | | |
| 14 | | | |
| 15 | | | |

Register 13.0: Access Cycle Count

This read-only register returns the number of processor cycles (subject to the dynamic clock) since system startup where nConfirm is high (processor active) and the processor is performing a memory action (ARequest != AREQ_NONE or RRequestI != 0).

Register 13.1: Idle Cycle Count

This read-only register returns the number of processor cycles (subject to the dynamic clock) since system startup where nConfirm is high (processor active) and the processor is not performing a memory action (ARequest == AREQ_NONE).

Register 13.2: Sleep Cycle Count

This read-only register returns the number of processor cycles (subject to the dynamic clock) since system startup where nConfirm is held low (processor gated) due to an interrupt suspend (see Register 14.2). Currently undercounts by 1 cycle per time going to sleep.

Register 13.3: Wait Cycles

This read-only register returns the number of cycles the core has been stalled by the cache controller. An extra cycle is counted at the end of each sleep period.

Register 13.4: Cache Hit Counter

This read-only register returns the number of accesses to regions marked as cacheable that resulted in a cache hit.

Register 13.5: Cache Miss Counter

This read-only register returns the number of accesses to regions marked as cacheable (see Register 14.7) that resulted in a cache miss..

Register 13.6: Cache Writeback Counter

This read-only register returns the number of cache misses that resulted in a write-back.

Register 13.7: Uncached Access Count

This read-only register returns the number of accesses (reads or write) that were made to uncached areas of memory. (Buffered and non-buffered).

Register 13.8: External Idle Counter (Simulator Only)

This read-only register returns the number of cycles that the external bus was idle.

Register 13.9: External Address Counter (Simulator Only)

This read-only register returns the number of generated address cycles on the external memory bus. This does not include cycles originating from external sources (e.g. DMA).

Register 13.10: External Data Counter (Simulator Only)

This read-only register returns the number of generated data cycles on the external memory bus. This does not include cycles originating from external sources (e.g. DMA).

Register 13.11: External Wait Counter (Simulator Only)

This read-only register returns the number of external memory cycles spent waiting for access to the external memory bus because some other device has the bus locked. The implementation of this register may be infeasible depending on the chosen implementation of the external low-power memory bus architecture; in this case, the results of reading this register are undefined.

Register 13.12: External Other Counter (Simulator Only)

This read-only register returns the number of external memory cycles that some other device is accessing the bus while the processor is *not* requesting access. The implementation of this register may be infeasible depending on the chosen implementation of the external low-power memory bus architecture; in this case, the results of reading this register are undefined.

Register 13.13 - 13.15: Unused

Register 14.0: Real Time Timer Low

This read-only register returns the lower 32 bits of the 64-bit real-time counter. This counter is independent of the current processor operating speed and is driven by the external reference clock. The time returned is in the time in 1 *us* increments since processor startup.

NOTE: A read of this register MAY be garbage due to asynchronous clock signals. Back-to-back reads may be recommended to ensure proper reading.

Register 14.1: Real Time Timer High

This read-only register returns the upper 32 bits of the 64-bit real-time counter (see Register 14.0).

NOTE: A read of this register MAY be garbage due to asynchronous clock signals. Back-to-back reads may be recommended to ensure proper reading.

Register 14.2: Timer Interrupt Control

A write to this register perform two functions. First, it clears any pending timer interrupt. Second, they set the time of the next timer interrupt. When the timer next transitions to the written value, a timer interrupt request (IRQ) will be generated. Writing the current timer value to the register will generate an interrupt immediately, so to safely clear it, write the current value - 1. (it will wait until the timer wraps around completely, which would take about 71 minutes with *us* resolution). There is no way to directly disable the timer interrupt; the interrupt signal may be blocked by the I bit in the PSR.

A read from this register simply returns the current timer interrupt value with no other effect.

Register 14.3: Interrupt Suspend/Information Register

A write to this register will assert the nConfirm line (pull it low) until either the FIQ or IRQ interrupt lines are asserted. This will halt the processor until there is an interrupt condition pending (the interrupt, however, may be blocked by the I and F bits in the internal PSR).

A read from this register returns the current state of the interrupt lines as indicated below. The EnIRQ bit as specified indicates a pending IRQ request from an external source, while the nTIQ line indicates a pending timer interrupt from the internal timer (see Register 14.1). The EnIRQ and nTIQ lines are merged into a single signal which is nIRQ (which is fed into the core). All bits are active-low (a 1 value indicates no interrupt condition pending).

Table 2: Interrupt Information Bitmap

| 31-10 | 9 | 8 | 7 | 6 | 5-0 |
|-------|------|-------|------|------|-----|
| x | nTIQ | EnIRQ | nIRQ | nFIQ | x |

NOTE: The suspend instruction MUST be followed by 2 null coproc writes. MCR p14, 0, X, c5, X

Register 14.4: Internal Dynamic Clock Speed

Writing to this register sets two target internal dynamic clock speeds: one for normal operation, and one for interrupts. The special interrupt clock speed can be enabled/disabled with a separate control bit for both IRQ and FIQ in C15R1.

Upon writing to this register with no pending interrupts, the desired clock value is sent to the regulation system. When either an FIQ or IRQ arrives (and the corresponding mask bit is enabled in C15R1), the interrupt clock speed is sent to the regulation system. Also, upon de-assertion of the interrupt, the normal clock rate is sent to the regulation system.

Table 3: Clock Speed Write Bitmap

| | | | |
|---------|-----------------------|---------|--------------------|
| 31-15 | 14-8 | 7 | 6-0 |
| ignored | interrupt clock speed | ignored | normal clock speed |

A read from this register returns the current sampled processor speed (not necessarily the same value written).

NOTE: To ensure a read of the correct value after a speed change, wait for two clock-ticks of the real-time counter before reading out the new value.

Register 14.5: External Pin Register

This register controls the state of 4 external pins. There is no other effect of writing to this register, and thus it is the recommended register to use when NULL co-proc writes are required (see Register 14.3). A read from this register returns the last value written.

Table 4: External Pin Mappings

| | |
|-----------|---------------|
| 31-4 | 3-0 |
| no effect | external pins |

A read from this register returns the current sampled processor speed (not necessarily the same value written).

Register 14.6: Hardware Control Tweaks

This register controls various aspects of the hardware system, as defined in the following tables:

Table 5: HCR Mappings

| | | |
|-----------|-----------|--------|
| 31-6 | 6-5 | 4-0 |
| undefined | Clk Ratio | to VCO |

Table 6: Bus Clock Ratio Settings

| bits 6:5 | bus clock ratio |
|----------|-----------------------|
| 00 | 8x |
| 01 | 4x |
| 10 | 2x |
| 11 | not used (same as 2x) |

Bits 4:0 are used to tune the exact frequency of the VCO. The bits all set to one gives approximately 1/2 x nominal frequency. All zero gives approximately 1 1/2 x nominal frequency. Upon reset, the internal state of the VCO will come up in a 60% nominal designed frequency. (but will still lock to the value in Register 13.4... that is, it will run at a higher than designed to supply voltage)

Register 14.7: Instruction Count

This read-only register returns the number of instruction executed since processor startup.

Register 14.8 - 14.15: Unused

Register 15.0: MMU ID

This read-only register returns the value 0x42018110.

Implementor: 0x42 ('B' for Berkeley).

Architecture Version: 0x01

Part-Number: 0x811

Revision: 0x0

Register 15.1: Processor Configuration

This read-write register controls system level aspects as described below. All values are set to zero on reset, except bits 0, 4 and 5, which are reset to one. Reading from this register simply returns the previous value written. Values in *italics* are not supported and are ignored. The **C**, and **W** bits effect the function of the cache memory system. The **B** and **Z** bits are fed back into the processor core to alter core functionality. The **A** bit is sent to the cache controller to alter response to non-aligned memory accesses. All others bits are read as 0 or 1, and are unalterable.

Table 7: Configuration Register Bitmap

| | | | | | | | | | | | | | | | | |
|-----------------|----------|-----------|-----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| | 31-15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | not used | IE | FE | <i>I</i> | Z | <i>F</i> | <i>R</i> | <i>S</i> | B | <i>L</i> | <i>D</i> | <i>P</i> | W | C | A | <i>M</i> |
| <i>init val</i> | 0.....0 | 0 | 0 | <i>0</i> | 0 | <i>0</i> | <i>0</i> | <i>0</i> | 0 | <i>0</i> | <i>1</i> | <i>1</i> | 0 | 0 | 0 | <i>1</i> |

- A:** Alignment Fault Enable
- C:** Cache Enable
- W:** Write-buffer Enable
- B:** Big Endian Select (else Little Endian)
- Z:** Branch Prediction Enable
- FE:** Enable different clock speed for FIQ (set with C14R4).
- IE:** Enable different clock speed for IRQ (set with C14R4).

The **C** and **W** bits are combined with the address bits 26 & 27 as described below to define the cache and write-buffer behaviour for the access. The cacheable, not-writebufferable region is useless from an application point of view, but is included for symmetry and failure-mode (i.e. if the write-buffer is faulty); the implication is that lines ejected from the cache due to the writeback nature are NOT passed through the write-buffer, and instead cause lengthy wait states while the line is written out.

Table 8: Address Bit <27:26> Access Encodings

| Binary Value (27:26) | Description |
|-------------------------|---|
| 00 | Cacheable and Write-bufferable (subject to the C and W bits above). |
| 01 | Cacheable, not write-bufferable |
| 10 | Write-bufferable, not cacheable |
| 11 | Neither cacheable or write-bufferable |

Register 15.2 - 15.6: Unused/Unimplemented

Register 15.7: Cache Control

Writing to this register will flush or clean cache blocks, as indicated in the table below. The Flush operation will invalidate all indicated cache data. The Clean operation write out dirty data held in the writeback cache. Operations which operate on a single cache entry may flush or clean more than a single entry, up to the entire cache.

These operations also require subsequent operations to the NULL coprocessor register to work properly with the cache system. The sequences are shown in the table below. (*X*=don't care) For Clean single entry, only the block enable bits of the VA are examined to determine which block to query for a dirty line.

Table 9: Cache Control Operations

| Function | opcode_2 value | CRm value | Data | Instruction |
|-----------------------|----------------|-----------|------|---|
| Flush ID cache(s) | 000 | 0111 | SBZ | MCR p15, 0, <i>X</i> , c7, c7, 0 MCR p14, 0, <i>X</i> , c5, <i>X</i> |
| Clean ID single entry | 001 | 1011 | VA | MCR p15, 0, Rd, c7, c11, 1 MCR p14, 0, Rd, c5, <i>X</i> |

(SBZ = Should Be Zero, VA = Virtual Address)

NOTE: For Clean ID single entry... Rd of *BOTH* instructions *MUST BE THE SAME*.

Register 15.8 - 15.15: Unused/Unimplemented

Coprocessor Implementation Specifics:

| CP # | Reg # | Register Description | Implementation | | |
|------|-------|-------------------------|----------------|------|---|
| | | | Bits | Type | Description |
| 13 | 0 | Access cycle count | 32 | RO | Counter |
| | 1 | Idle Cycle count | 32 | RO | Counter |
| | 2 | Sleep cycle count | 32 | RO | Counter |
| | 3 | Wait cycle count | 32 | RO | Counter |
| | 4 | Hit count | 32 | RO | Counter |
| | 5 | Cache miss count | 32 | RO | Counter |
| | 6 | Cached writeback count | 32 | RO | Counter |
| | 7 | Uncached access count | 32 | RO | Counter |
| 14 | 0 | RT counter low | 32 | RO | Counter |
| | 1 | RT counter high | 32 | RO | Counter |
| | 2 | Timer interrupt | 32 | R/W | Register |
| | 3 | Interrupt suspend | 4 | R/W | Non-writable register (9:6). Bits (31:10) and (5:0) = 0. |
| | 4 | Internal clock speed | 7 | R/W | Register (6:0). Bits (31:7) = 0. |
| | 5 | External pins | 4 | R/W | Register (3:0). Bits (31:4) = 0. |
| | 6 | Hardware control tweaks | 7 | R/W | Register (6:0). Bits (31:7) = 0. |
| | 7 | Instruction Count | 32 | RO | Counter |
| 15 | 0 | MMUID | 32 | RO | Hard coded |
| | 1 | System Control | 15 | R/W | Register (14:0). Bits (31:15) = 0. |
| | 7 | Cache Operations | na | WO | Sets control signals. Only 2 different valid instructions. All others bounce. |

Coprocessor Testing Conditions:

| <u>Test Description</u> | <u>Tested Register</u> | |
|---|------------------------|------------|
| 1. Enable Cache & Write Buffer..... | CP(15)(1)(3:2) | write |
| 2. Set bus speed to 2x and write 0x1f to lower bits | CP(14)(6)(6:0) | write |
| 3. Initial CPSR check..... | | |
| 4. MMUID read | CP(15)(0) | read |
| 5. MMU control read (including align bit)..... | CP(15)(1) | read |
| 6. Verify clock frequency is 50 MHz | CP(14)(4) | read |
| 7. Check real-time clock at 50 MHz..... | CP(14)(0) | read |
| 8. Set clock frequency to 10 MHz (and verify) | CP(14)(4) | write/read |
| 9. Check real-time clock at 10 MHz..... | CP(14)(0) | read |
| 10. Check external bus ratio at 2x and 8x..... | CP(14)(6) | read |
| 11. Check timer interrupt | CP(14)(2) | read/write |
| 12. Check sleep register | CP(14)(3) | write |
| 13. Check interrupt status register (for timer interrupt) | CP(14)(3) | read |
| 14. Check control over branch predictor | CP(15)(1)(7) | write |
| 15. Check instruction count register | CP(14)(7) | read |
| 16. Bogus coprocessor operations (incl. all STC/LDC/CDP) | many undefined regs. | |
| 17. Bogus cache control operations | CP(15)(7) | write |
| 18. Failed conditional MCR/MRC instruction | | |
| 19. Failed conditional STC instruction. | | |
| 20. Failed speculative MCR/MRC instructions | | |
| 21. Back-to-back MCR/MRC and MRC/MCR | CP(14)(5) | read/write |
| 22. Load value from mem then MCR (LDR and LDRB)..... | CP(14)(5) | read/write |
| 23. Exercise all possible cache operations | CP(15)(7) | write |
| 24. Read all count registers..... | CP(13)(0-7) | read |
| 25. Check Timer High word | CP(14)(1) | read |

To Be Added:

26.