MIPS32® M6200 Processor Core Family Datasheet December 15, 2015

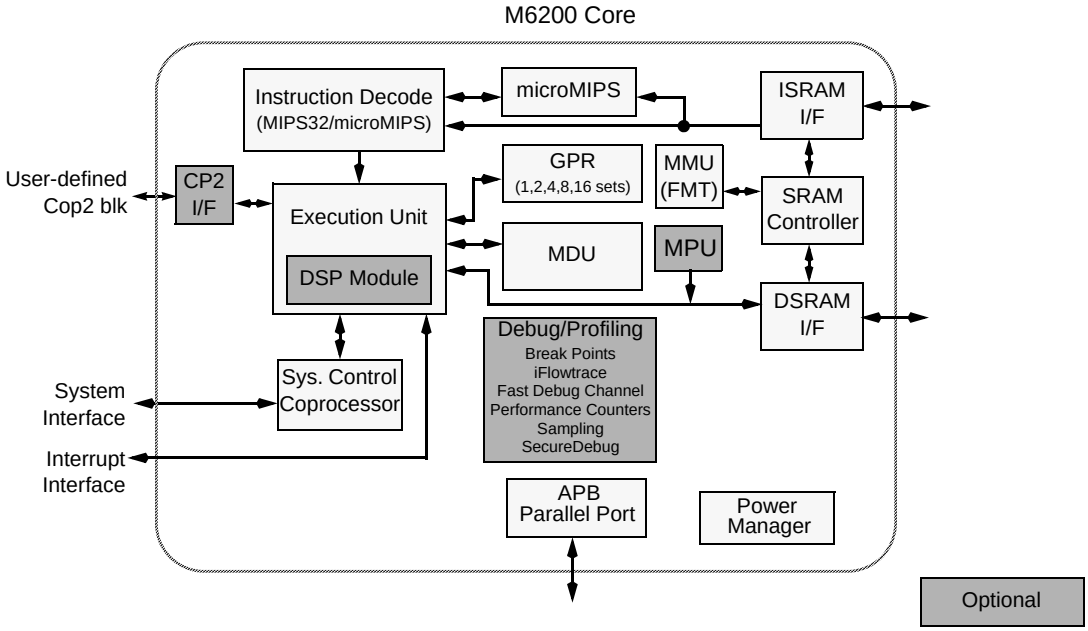
The MIPS32® M6200 core from Imagination Technologies is a member of the MIPS32 M6200 processor core family. This document refers to all members of the M6200 family.

The M6200 core is a configurable and synthesizable solution for use in microcontroller and real-time embedded applications requiring a high level of performance efficiency, small silicon area, and low power.

The M6200 processor is a 6-stage pipeline design that implements the MIPS32 Release 6 Architecture. It also includes support for the microMIPS32™ ISA, an Instruction Set Architecture with optimized MIPS32 16-bit and 32-bit instructions that provides a significant reduction in code size with a performance equivalent to MIPS32. The M6200 core is an enhancement of the microAptiv™ UC core, capable of operating up to 30% higher frequency. The M6200 core includes the MIPS Architecture DSP Module Revision 3 as a configurable option, providing a high level of digital signal processing capabilities and Single Instruction Multiple Data (SIMD) support.

Figure 1 shows a block diagram of the M6200 core. Optional blocks are shown as shaded.

Figure 1 MIPS32® M6200 Core Block Diagram



Features

- The M6200 builds on the feature set supported by the MIPS32® microAptiv™ UC core as outlined below, but adds the following key capabilities:
  - 6-stage pipeline implementation, enabling up to 30% higher frequencies
  - Support for MIPS32® R6 architecture
  - Support for microMIPS™ R6 ISA
  - 32-bit Address and 64-bit wide Data Bus

    - Data integrity features:
        - ECC (optional) on all ISRAMs and DSRAMs
        - Parity (optional) on address and data bus
- The M6200 retains features from the microAptiv UC, including:
    - 32-bit General Purpose Registers (GPR)
    - Instruction and Data SRAM interfaces
    - Memory Protection Unit
    - Microcontroller Application Specific Extension (MCU ASE)
    - Multiply/Divide Unit
    - DSP Module (optional)
    - Debug and profiling support
    - Secure Debug
    - Coprocessor interface
    - Power management
- MIPS32 Architecture Features
    - Vectored interrupts and support for external interrupt controller
    - Programmable exception vector base
    - Simple boot exception vector relocation via 2 externally controlled pins
    - GPR shadow registers (1, 2, 4, 8, or 16 additional shadows can be optionally added to minimize latency for interrupt handlers)
    - Bit field manipulation instructions
- microMIPS32 Instruction Set Architecture Release 6
    - microMIPS ISA reduces code size over MIPS32, while maintaining MIPS32 performance.
    - Combining both 16-bit and 32-bit opcodes, microMIPS supports all MIPS32 instructions, with new optimized encoding. Frequently used MIPS32 instructions are available as 16-bit instructions.
    - Stack pointer implicit in instruction.
    - MIPS32 assembly and ABI-compatible.
    - Supports MIPS architecture Modules and ASEs, including DSP and MCU.
- MCU™ ASE
    - Increases the number of interrupt hardware inputs from 6 to 8 for Vectored Interrupt (VI) mode, and from 63 to 255 for External Interrupt Controller (EIC) mode.
    - Separate priority and vector generation. 16-bit vector address is provided.
    - Hardware assist combined with the use of Shadow Register Sets to reduce interrupt latency during the prologue and epilogue of an interrupt.
    - An interrupt return with automated interrupt epilogue handling instruction (IRET) improves interrupt latency.
    - Supports optional interrupt chaining.
    - Two memory-to-memory atomic read-modify-write instructions (ASET and ACLR) eases commonly used semaphore manipulation in microcontroller applications. Interrupts are automatically disabled during the operation to maintain coherency.
- Memory Management Unit
    - Simple Fixed Mapping Translation (FMT) mechanism
- ECC Support
    - The ISRAM and DSRAM support optional single-error correction and double- error detection (SECDED), with correction in software.
- Transmission Parity Support
    - The ISRAM and DSRAM interfaces support optional parity detection on transactions between master and slave.
- Memory Protection Unit
    - Optional feature that improves system security by restricting access, execution, and trace capabilities from untrusted code in predefined memory regions.
- Simple SRAM-Style Interface
    - 32-bit address and 64-bit data
    - Single or multi-cycle latencies
    - Dual instruction and data interfaces
    - Partially registered interface
- MIPS DSP Module (Revision 3.0)
    - Support for MAC operations with 4 additional pairs of HI/LO accumulator registers (Ac0 - Ac3)
    - Fractional data types (Q15, Q31) with rounding support
    - Saturating arithmetic with overflow handling
    - SIMD instructions operate on 2x16-bit or 4x8-bit operands simultaneously
    - Separate MDU pipeline with full-sized hardware multiplier to support back-to-back operations
    - The DSP Module is build-time configurable.
- Multiply/Divide Unit (without DSP)
    - Maximum issue rate of one 32x16 multiply per clock via on-chip 32x16 hardware multiplier array.

    - Maximum issue rate of one 32x32 multiply every other clock
    - Early-in iterative divide. Minimum 11 and maximum 34 clock latency (dividend (*rs*) sign extension-dependent)
- Multiply/Divide Unit (with DSP configuration)
    - Maximum issue rate of one 32x32 multiply per clock via on-chip 32x32 hardware multiplier array
    - Maximum issue rate of one 32x32 multiply every clock
    - Early-in iterative divide. Minimum 12 and maximum 38 clock latency (dividend (*rs*) sign extension-dependent)
- Multi-Core Support
    - External lock indication enables multi-processor semaphores based on LL/SC instructions
    - External sync indication allows memory ordering
- Coprocessor 2 Interface
    - 64b data width interface to an external coprocessor
- Interrupt Controller Unit
    - An optional feature that provides supports for up to 256 interrupts, configurable at build-time in options of 8, 16, 32, 64, 128, and 256 sources
    - Interrupts are configurable as to polarity, level or edge sensitivity, and dual- or single-edge sampling
    - Support for MCU ISA
    - Includes 32-bit Watchdog timer
- Power Control
    - Minimum frequency: 0 MHz
    - Power-down mode (triggered by WAIT instruction)
- Debug/Profiling and iFlowtrace™ Mechanism
    - CPU control with start, stop, and single stepping
    - Virtual instruction and data address/value breakpoints
    - Hardware breakpoint supports both address match and address range triggering
    - Optional simple hardware breakpoints on virtual addresses; 8I/4D or 4I/2D breakpoints, or no breakpoints
    - Complex hardware breakpoints with 8I/4D simple breakpoints
    - iFlowtrace support for real-time instruction PC and special events
    - PC and/or load/store address sampling for profiling
    - Performance Counters
    - Support for Fast Debug Channel (FDC)
    - Support for trace conversion block that converts iFlowtrace signals to ATB interface signals.
- Secure Debug
    - An optional feature that disables debug access in an untrusted environment
- Testability
    - Full scan design achieves test coverage in excess of 99% (dependent on library and configuration options)

Architecture Overview

The M6200 core contains both required and optional blocks, as shown in Figure 1. Required blocks must be implemented to remain MIPS-compliant. Optional blocks can be added to the M6200 core based on the needs of the implementation.

The required blocks are as follows:

- Instruction Decode
- Execution Unit
- General Purpose Registers (GPR)
- Multiply/Divide Unit (MDU)
- System Control Coprocessor (CP0)
- Memory Management Unit (MMU)
- I/D SRAM Interfaces
- Power Management

Optional or configurable blocks include:

- DSP (integrated with MDU)
- Memory Protection Unit (MPU)
- Coprocessor 2 interface
- Interrupt Controller Unit (ICU)
- Debug/Trace/Profiling with optional APB Debug, Hardware Breakpoints, PC Sampling, Performance Counters, Fast Debug Channel, and iFlowtrace

The section "MIPS32® M6200 Core Required Logic Blocks" on page 5 discusses the required blocks. The section "MIPS32® M6200 Core Optional or Configurable Logic Blocks" on page 10 discusses the optional blocks.

Pipeline Flow

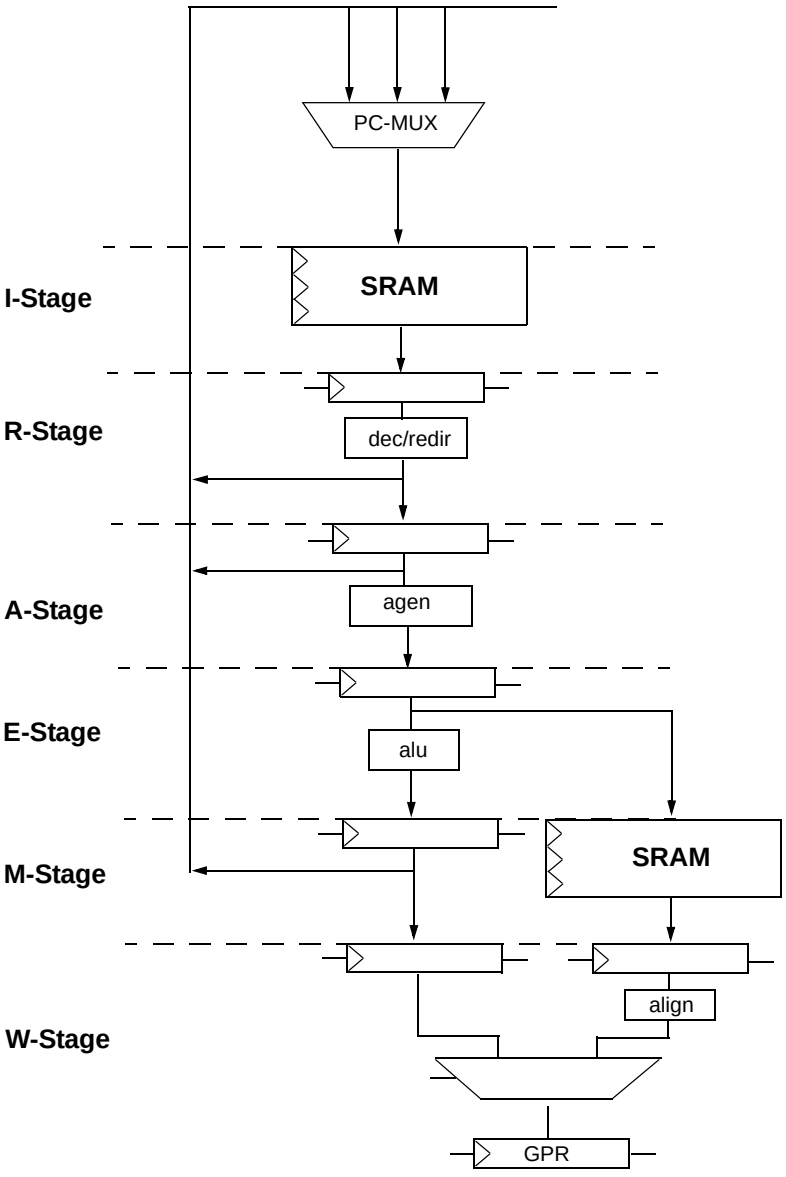
The M6200 core implements a 6-stage pipeline. The pipeline allows the processor to achieve high frequency while minimizing device complexity, reducing both cost and power consumption.

The M6200 core pipeline consists of six stages:

- Instruction (I Stage)
- Register (R Stage)
- Address Generation (A Stage)
- Execution (E Stage)
- Memory (M Stage)
- Write (W stage)

The M6200 core implements a bypass mechanism that allows the result of an operation to be forwarded directly to the instruction that needs it without having to write the result to the register and then read it back. Figure 2 shows a diagram of the M6200 core pipeline.

Figure 2 MIPS32® M6200 Core Pipeline



I Stage: Instruction Fetch

- An instruction is fetched from the instruction SRAM.

R Stage: Register File Access

- Instructions are partially decoded.
- Register file is read.
- Jump instructions are decoded. If conditions permit, the M6200 core will improve performance by performing early redirection and fetch the next target instruction before the jump-instruction has graduated.

A Stage: Address Generation

- Data access addresses are calculated.
- Instructions are fully decoded
- Multiplication Booth recode is performed.

E Stage: Execution

- Branch evaluation is performed, causing a redirect if the instruction branches.
- Arithmetic, logic, and multiplication operations are performed. Depending on the size of the operands, the multiplication operation may be double pumped.
- Division uses an iterative sequence, and is non-blocking until an instruction uses the division result.
- Fixed-mapping translation is used for cores without TLB support

M Stage: Memory Access

- Data access is performed for load and store instructions.
- On cacheable accesses, store instructions are written to internal write buffers, where the data will be written to external memory as soon as the interface is free or idle.
- On uncacheable accesses, store instructions obeys strongly-ordered memory consistency rules by stalling until the write is the oldest instruction in the pipeline before writing to the external memory.

W Stage: Write

- Exceptions are prioritized and flagged.
- Data reads are aligned before writing to the register file.

MIPS32® M6200 Core Required Logic Blocks

The required logic blocks of the M6200 core (Figure 1) are defined in the following subsections.

Execution Unit

The M6200 core execution unit implements a load/store architecture with single-cycle ALU operations (logical, shift, add, subtract) and an autonomous multiply/divide unit.

The execution unit includes:

- Single cycle Arithmetic Logic Unit (ALU) for performing arithmetic, bitwise logical operations and branch target calculation.
- Adder for load/store address calculation
- Address unit for calculating the next PC and next fetch address selection muxes.
- Load Aligner.
- Shifter and Store Aligner.
- Branch condition comparator.
- Bypass muxes to advance result between two instructions with data dependency.
- Leading Zero/One detect unit for implementing the CLZ and CLO instructions.
- Read-modify-write control logic implementing atomic instructions defined in the MCU ASE.
- Actual execution of the Atomic Instructions defined in the MCU ASE.
- A separate DSP ALU and Logic block for performing part of DSP Module instructions, such as arithmetic/shift/compare operations when the DSP function is configured.

General Purpose Registers

The M6200 core contains thirty-two 32-bit general-purpose registers used for integer operations and address calculation. Optionally, 1, 2, 4, 8, or 16 additional register file shadow sets (each containing thirty-two registers) can be added to minimize context switching overhead during interrupt/exception processing. The register file is flop-based and is fully bypassed to minimize operation latency in the pipeline.

Multiply/Divide Unit (MDU)

The M6200 core includes a multiply/divide unit (MDU) that contains a separate, dedicated pipeline for integer multiply/divide operations and DSP Module multiply instructions (with DSP option). This pipeline operates in parallel with the integer unit (IU) pipeline and does not stall when the IU pipeline stalls. This allows the long-running MDU operations to be partially masked by system stalls and/or other integer unit instructions.

The MIPS architecture defines that the result of a multiply or divide operation be placed in general-purpose registers (without DSP option) or one of four pairs of *HI* and *LO* registers (with DSP enabled).

MDU with 32x32 DSP Multiplier with DSP Option

With the DSP configuration option enabled, the MDU supports execution of one 16x16, 32x16, or 32x32 multiply or multiply-accumulate operation every clock cycle with the built-in 32x32 multiplier array. The multiplier is shared with DSP Module operations.

The MDU also implements various shift instructions operating on the HI/LO register and multiply instructions as defined in the DSP Module. It supports all the data types required for this purpose and includes four extra HI/LO registers as defined by the Module.

Table 1 lists the latencies (number of cycles for an instruction to propagate from the beginning to the end of the core's pipeline) and repeat rates (throughput without data dependency) for the DSP multiply and dot-product operations. The approximate latencies and repeat rates are listed in terms of pipeline clocks. For a more detailed discussion of latencies and repeat rates, refer to the *MIPS32 M6200 Processor Core Programmer's Guide*.

Table 1 DSP-related Latencies and Repeat Rates

Opcode	Latency	Repeat Rate
Multiple and dot-product without saturation after accumulation	6	1
Multiple and dot-product with saturation after accumulation (word)	6	1
Multiply and dot-product with saturation after accumulation (doubleword)	7	1
Accumulator shifter uses immediately previous multiply result.	7	2
Multiply without accumulation	6	1

MDU with 32x16 High-Performance Multiplier

Without the DSP option, the high-performance MDU consists of a 32x16 Booth-recoded multiplier, a divide state machine, and the necessary multiplexers and control logic. The first number shown ('32' of 32x16) represents the *rs* operand. The second number ('16' of 32x16) represents the *rt* operand. The M6200 core only checks the value of the *rt* operand to determine how many times the operation must pass through the multiplier. The 16x16 and 32x16 operations pass through the multiplier once. A 32x32 operation passes through the multiplier twice.

The MDU supports execution of one 16x16 or 32x16 multiply or multiply-accumulate operation every clock cycle; 32x32 multiply operations can be issued every other clock cycle. Appropriate interlocks are implemented to stall the issuance of back-to-back 32x32 multiply operations. The multiply operand size is automatically determined by logic built into the MDU.

Table 2 and Table 3 list the repeat rate (how often the operation can be reissued when there is no data dependency) and latency (number of cycles until a result is available) for the multiply and divide instructions. The approximate latency and repeat rates are listed in terms of pipeline clocks. For a more detailed discussion of latencies and repeat rates, refer to the *MIPS32 M6200 Processor Core Family Programmer's Guide*.

Table 2 High-Performance Integer Multiply/Divide Unit Latencies and Repeat Rates with DSP

Opcode	Operand Size (mul rt) (div rs)	Latency	Repeat Rate
MUL, MUH, MULU, MUHU (GPR destination)	16 bits	6	1
	32 bits	6	1
DIV, MOD/ DIVU, MODU (GPR destination)	4 bits	11/10	8/7
	8 bits	15/14	12/11
	16 bits	23/22	20/19
	24 bits	31/30	28/27
	32 bits	39/38	36/35
MADD/MADDU, MSUB/MSUBU (with DSP)	GPR is 32-bit Accumulator is 64-bit	6	1

Table 3 High-Performance Integer Multiply/Divide Unit Latencies and Repeat Rates without DSP

Opcode	Operand Size (mul rt) (div rs)	Latency	Repeat Rate
MUL, MUH, MULU, MUHU (GPR destination)	16 bits	6 (I-R-A-E-M-W)	1
	32 bits	7 (I-R-A-E-E-M-W, 32x16 array is in E)	2
DIV, MOD/DIVU, MODU (GPR destination)	4	11/10	8/7
	8 bits	15/14	12/11
	16 bits	23/22	20/19
	24 bits	31/30	28/27
	32 bits	39/38	36/35

System Control Coprocessor (CP0)

In the MIPS architecture, CP0 is responsible for the virtual-to-physical address translation, the exception control system, the processor's diagnostics capability, the operating modes (kernel, user, and debug), and whether interrupts are enabled or disabled. Configuration information, such as presence of build-time options, such as microMIPS or Coprocessor 2 interface, is also available by accessing the CP0 registers.

Coprocessor 0 also contains the logic for identifying and managing exceptions. Exceptions can be caused by a variety of sources, including boundary cases in data, external events, or program errors.

Interrupt Handling

The M6200 core includes support for eight hardware interrupt pins, two software interrupts, and a timer interrupt. These interrupts can be used in any of three interrupt modes, as defined by Release 2 of the MIPS32 Architecture:

- Interrupt compatibility mode, which acts identically to that in an implementation of Release 1 of the Architecture.
- Vectored Interrupt (VI) mode, which adds the ability to prioritize and vector interrupts to a handler dedicated to that interrupt, and to assign a GPR shadow set for use during interrupt processing. The presence of this mode is denoted by the VInt bit in the *Config3* register. This mode is architecturally optional; but it is always present on the M6200 core, so the VInt bit will always read as a 1 for the M6200 core.
- External Interrupt Controller (EIC) mode, which redefines the way in which interrupts are handled to provide full support for an external interrupt controller handling prioritization and vectoring of interrupts. The presence of this mode is indicated by the *VEIC* bit in the *Config3* register. Again, this mode is architecturally optional. On the M6200 core, the *VEIC* bit is set externally by the static input *SI_EICPresent*, which allows system logic to indicate the presence of an external interrupt controller.

The reset state of the processor is interrupt compatibility mode, such that processors supporting Release 6 of the Architecture (such as the M6200 core) are fully compatible with implementations of Release 1 of the Architecture.

VI or EIC interrupt modes can be combined with the optional shadow registers to specify which shadow set should be used on entry to a particular vector. The shadow registers improve interrupt latency by avoiding the need to save context when invoking an interrupt handler.

In the M6200 core, interrupt latency is reduced by:

- Speculative interrupt-vector prefetching during the pipeline flush.
- Interrupt Automated Prologue (IAP) in hardware: Shadow Register Sets remove the need to save GPRs, and IAP removes the need to save specific Control Registers when handling an interrupt.
- Interrupt Automated Epilogue (IAE) in hardware: Shadow Register Sets remove the need to restore GPRs, and IAE removes the need to restore specific Control Registers when returning from an interrupt.
- Allow interrupt chaining. When servicing an interrupt and interrupt chaining is enabled, there is no need to return from the current Interrupt Service Routine (ISR) if there is another valid interrupt pending to be serviced. The control of the processor can jump directly from the current ISR to the next ISR without IAE and IAP.
- Simple exception vector relocation via the externally controlled pin *SI_Offset*.

GPR Shadow Registers

The MIPS32 Architecture optionally removes the need to save and restore GPRs on entry to high-priority interrupts or exceptions, and to provide specified processor modes with the same capability. This is done by introducing multiple copies of the GPRs, called *shadow sets*, and allowing privileged software to associate a shadow set with entry to kernel mode via an interrupt vector or exception. The normal GPRs are logically considered shadow set zero.

The number of GPR shadow sets is a build-time option. The M6200 core allows 1 (the normal GPRs), 2, 4, 8, or 16 shadow sets. The highest number actually implemented is indicated by the $SRSCtl_{HSS}$ field. If this field is zero, only the normal GPRs are implemented.

Shadow sets are new copies of the GPRs that can be substituted for the normal GPRs on entry to kernel mode via an interrupt or exception. Once a shadow set is bound to a kernel-mode entry condition, references to GPRs operate exactly as one would expect, but they are redirected to registers that are dedicated to that condition. Privileged software may need to reference all GPRs in the register file, even specific shadow registers that are not visible in the current mode, and the RDPGPR and WRPGPR instructions are used for this purpose. The *CSS* field of the *SRSCtl* register provides the number of the current shadow register set, and the *PSS* field of the *SRSCtl* register provides the number of the

previous shadow register set that was current before the last exception or interrupt occurred.

If the processor is operating in VI interrupt mode, binding of a vectored interrupt to a shadow set is done by writing to the *SRSMap* register. If the processor is operating in EIC interrupt mode, the binding of the interrupt to a specific shadow set is provided by the external interrupt controller and is configured in an implementation-dependent way. Binding of an exception or non-vectored interrupt to a shadow set is done by writing to the *ESS* field of the *SRSCtl* register. When an exception or interrupt occurs, the value of $SRSCtl_{CSS}$ is copied to $SRSCtl_{PSS}$, and $SRSCtl_{CSS}$ is set to the value taken from the appropriate source. On an ERET, the value of $SRSCtl_{PSS}$ is copied back into $SRSCtl_{CSS}$ to restore the shadow set of the mode to which control returns.

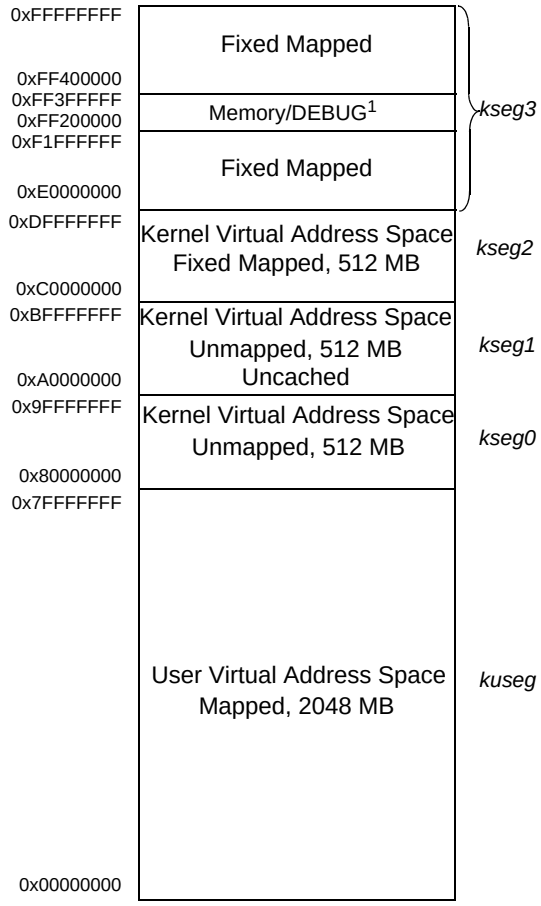
Modes of Operation

The M6200 core implements three modes of operation:

- *User mode* is most often used for applications programs.
- *Kernel mode* is typically used for handling exceptions and operating-system kernel functions, including CP0 management and I/O device accesses.
- *Debug mode* is used during system bring-up and software development.

Figure 3 shows the virtual address map of the MIPS Architecture.

Figure 3 M6200 Core Virtual Address Map



1. This space is mapped to memory in user or kernel mode, and by the Debug module in debug mode.

Memory Management Unit (MMU)

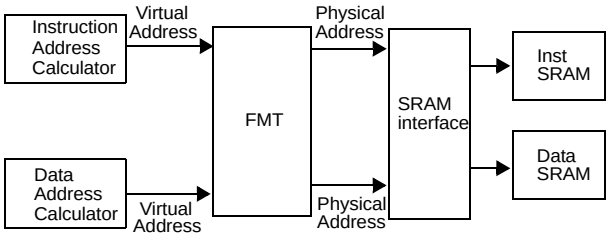
The M6200 core contains a simple Fixed Mapping Translation (FMT) MMU that interfaces between the execution unit and the SRAM controller.

Fixed Mapping Translation (FMT)

A FMT is smaller and simpler than the full Translation Lookaside Buffer (TLB) style MMU found in other MIPS cores. Like a TLB, the FMT performs virtual-to-physical address translation and provides attributes for the different segments. Those segments that are unmapped in a TLB implementation (kseg0 and kseg1) are translated identically by the FMT.

Figure 4 shows how the FMT is implemented in the M6200 core.

Figure 4 Address Translation During SRAM Access with FMT Implementation



SRAM Interface Controller

The M6200 core contains an interface to SRAM-style memories that is lightly-coupled to the core (an unregistered interface). This permits deterministic response time with less area than is typically required for caches. The SRAM interface includes separate uni-directional 32-bit busses for address, and 64-bit wide buses for read data and write data. All writes are non-posted.

The SRAM interface is a dual interface that enables independent connection to instruction and data devices. It generally yields the highest performance, because the pipeline can generate simultaneous I and D requests, which are then serviced in parallel.

Back-stalling

Typically, read and write transactions will complete in a single cycle. However, if multi-cycle latency is desired, the interface can be stalled to allow connection to slower devices.

Lock Mechanism

The SRAM interface includes two protocols to identify a locked sequence. One is used in conjunction with the LL/SC atomic read-modify-write semaphore instructions, and the other is used in conjunction with the MCU ASE Atomic instructions, namely, ASET and ACLR. In the former, the interface includes a protocol that externalizes the execution of the SYNC instruction. External logic may choose to use this information to enforce memory ordering between various elements in the system.

Sync Mechanism

The interface includes a protocol that externalizes the execution of the SYNC instruction. External logic might choose to use this information to enforce memory ordering between various elements in the system.

Hardware Reset

The M6200 core has two reset input signals: *SI_WarmResetN* and *SI_ColdResetN*. These two signals are used to initialize critical hardware state.

Both reset signals are active Low and can be asserted synchronously or asynchronously to the core clock, *SI_ClkIn*, and will trigger a Reset exception. The CPU will internally generate a reset pulse of 6 clock cycles, whereupon the core will exit reset synchronously.

The primary difference between the two reset signals is that *SI_WarmResetN* sets a bit in the *Status* register; this bit could be used by software to distinguish between the two reset signals, if desired. The reset behavior is summarized in Table 4.

Table 4 Reset Types

SI_WarmResetN	SI_ColdResetN	Action
1	1	Normal operation, no reset.
0	1	Reset exception; sets $Status_{SR}$ bit.
X	0	Reset exception.

One (or both) of the reset signals must be asserted at power-on or whenever hardware initialization of the core is desired. A power-on reset typically occurs when the machine is first turned on. A hard reset usually occurs when the machine is already on and the system is rebooted.

In debug mode, a Debug probe can request that a soft reset (via the *SI_WarmResetN* pin) be masked. It is system-dependent whether this functionality is supported. In normal mode, the *SI_WarmResetN* pin cannot be masked; the *SI_ColdResetN* pin is never masked.

Power Management

The M6200 core offers a number of power management features, including low-power design, active power management, and power-down modes of operation. The core is a static design that supports slowing or halting the clocks, which reduces system power consumption during idle periods.

The M6200 core provides two mechanisms for system-level low-power support:

- Register-controlled power management
- Instruction-controlled power management

Register-Controlled Power Management

Three bits—$Status_{EXL}$, $Status_{ERL}$, and $Debug_{DM}$—support the power-management function by allowing the user to change the power state if an exception or error occurs while the core is in a low-power state. Depending on what type of exception is taken, one of these three bits will be asserted and reflected on the *SI_EXL*, *SI_ERL*, or *EJ_DebugM* outputs. The external agent can look at these signals and determine whether to leave the low-power state to service the exception.

The following four power-down signals are part of the system interface and change state as the corresponding bits in the CP0 registers are set or cleared:

- The *SI_EXL* signal represents the state of the *EXL* bit (1) in the CP0 *Status* register.
- The *SI_ERL* signal represents the state of the *ERL* bit (2) in the CP0 *Status* register.
- The *EJ_DebugM* signal represents the state of the DM bit (30) in the CP0 *Debug* register.

Instruction-Controlled Power Management

The second mechanism for invoking power-down mode is by executing the WAIT instruction. When the WAIT instruction is executed, the internal clock is suspended; however, the internal timer and some of the input pins (*SI_Int[7:0]*, *SI_NMI*, *SI_WarmResetN*, and *SI_ColdResetN*) continue to run. Once the CPU is in instruction-controlled power management mode, any interrupt, NMI, or reset condition causes the CPU to exit this mode and resume normal operation.

The M6200 core asserts the *SI_Sleep* signal, which is part of the system interface bus, whenever the WAIT instruction is executed. The assertion of *SI_Sleep* indicates that the clock has stopped and the M6200 core is waiting for an interrupt.

MIPS32® M6200 Core Optional or Configurable Logic Blocks

The M6200 core contains several optional or configurable logic blocks, shown as shaded in the block diagram in Figure 1.

Data Integrity

The M6200 core optionally supports single-error correction and double-error detection (SECDED). Errors are reported in COP0 registers and rely on software correction. ECC is generated or checked for valid byte lanes.

In addition to ECC protection, the M6200 core also offers data integrity protection on uncached data transmissions. Parity is generated and checked for every 8 bits of data transferred, and every 32 bits of address transferred.

Memory Protection Unit

The Memory Protection Unit can be configured to have from 1 to 16 memory protection regions. Each region is enabled by registers that define the address, size, and protection of each memory region. The Memory Protection Unit control is implemented by CDMM (Common Device Memory Map) registers. After they have been programmed, these control registers can be locked to prohibit later modifications. Once programmed, a Protection Exception will be triggered when an Instruction Fetch or Data Access matches the address of the protected memory region or any modification of the EBase (base address of exception vectors) register was attempted. Each protected region can also disable the iFlowtrace capability. Typically, the Memory Protection Unit improves system security by disabling access to bootcode and preventing execution of non-trusted kernel mode code.

DSP Module

The M6200 core implements an optional DSP Module to benefit a wide range of DSP, Media, and DSP-like algorithms. The DSP module is highly integrated with the Execution Unit and the MDU in order to share common logic and to include support for operations on fractional data types, saturating arithmetic, and register SIMD operations. Fractional data types Q15 and Q31 are supported. Register SIMD operations can perform up to four simultaneous add, subtract, or shift operations and two simultaneous multiply operations.

In addition, the DSP Module includes some key features that efficiently address specific problems often encountered in DSP applications. These include, for example, support for complex multiply, variable-bit insert and extract, and

implementation and use of virtual circular buffers. The extension also makes available four additional sets of HI-LO accumulators to better facilitate common accumulate functions such as filter operation and convolutions.

Coprocessor 2 Interface

The M6200 core can be configured to have an interface for an on-chip coprocessor. This coprocessor can be tightly coupled to the processor core, allowing high-performance solutions integrating a graphics accelerator or DSP, for example.

The coprocessor interface is extensible and standardized on MIPS cores, allowing for design reuse. The M6200 core supports a subset of the full coprocessor interface standard: 64b data transfer, no Coprocessor 1 support, and single issue in-order data transfer to coprocessor.

The coprocessor interface is designed to ease integration with customer IP. The interface allows high-performance communication between the core and coprocessor. There are no late or critical signals on the interface.

Interrupt Controller Unit (ICU)

The M6200 includes an optional Interrupt Controller Unit (ICU) that supports the following features:

- Accepts up to 256 interrupt sources configurable at build-time with option of 8, 16, 32, 64, 128, or 256 sources.
- Supports MCU ASE, where outputs drives an 8-bit requested interrupt priority level.
- Distributes (hardwired) interrupt sources to the core.
- Backward compatibility with pre-defined MIPS Technologies interrupt modes configurable by software.
- Supports interrupt source sensitivity (level-positive, level-negative, edge-positive, edge-negative, dual-edge-sensitive) configurable at build-time. All sources are normalized to positive, level-sensitive signals.
- Interrupt Pending mask feature.
- Interrupt sources are mapped to *SI_Int[7:0]* or NMI. Mapping is software-controlled, and control registers are extended to reflect the widening of the *SI_Int* output bus.
- Single 32-bit watch-dog timer.
- Supports EIC Shadow register set use in EIC interrupt mode. Shadow set values are hard-wired (a build-time option).

Debug Support

The M6200 core provides for an optional Debug interface via the MIPS Debug Hub (MDH) and Advanced Peripheral Bus (APB) interface. MDH provides the capability for connection to a JTAG or APB compatible debug system for improved debug performance and support for multi-core systems. For more information, refer to the *MIPS Debug Hub Technical Reference Manual* (MD01070).

The APB accesses a number of registers used for determining core and debug status, changing debug modes, causing instructions to be executed, and providing other instrumentation functions.

The M6200 core also provides a special Debug mode of operation, in addition to standard User mode and Kernel modes of operation. Debug mode is entered after a debug exception is taken and continues until a debug exception return (DERET) instruction is executed. During this time, the processor executes the debug exception handler routine.

Optionally configurable at build time, a trace conversion block (PDT2ATB) can be included in the M6200 environment. The trace conversion block resides outside of the M6200 core and converts iFlowtrace signals to AMBA Trace Bus (ATB) interface signals.

CP0 Debug Registers

Four debug registers (*DEBUG*, *DEBUG2*, *DEPC*, and *DESAVE*) have been added to the MIPS Coprocessor 0 (CP0) register set. The *DEBUG* and *DEBUG2* registers show the cause of the debug exception and are used for setting up single-step operations. The *DEPC* (Debug Exception Program Counter) register holds the address on which the debug exception was taken, which is used to resume program execution after the debug operation completes. Finally, the *DESAVE* (Debug Exception Save) register enables the saving of general-purpose registers used during execution of the debug exception handler.

To exit debug mode, a Debug Exception Return (DERET) instruction is executed. When this instruction is executed, the system exits debug mode, allowing normal execution of application and system code to resume.

Hardware Breakpoints

There are several types of *simple* hardware breakpoints defined in the Debug specification. These halt the normal operation of the CPU and force the system into debug mode. There are two types of simple hardware breakpoints implemented in the M6200 core: Instruction breakpoints and Data breakpoints. Additionally, *complex* hardware breakpoints can be included, which allow the detection of more intricate sequences of events.

The M6200 core can be configured with the following breakpoint options:

- No data or instruction
- Two data and four instruction breakpoints
- Four data and eight instruction breakpoints, with complex breakpoints

Instruction breakpoints match on instruction execution operations, and the breakpoint is set on the virtual address. A mask can be applied to the virtual address to set breakpoints on a range of instructions.

Data breakpoints match on load/store transactions, and the breakpoint is set on a virtual address value, with the same single address or address range as the Instruction breakpoint. Data breakpoints can be set on a load, a store, or both. Data breakpoints can also be set to match on the operand value of the load/store operation, with byte-granularity masking. Finally, masks can be applied to both the virtual address and the load/store value.

In addition, the M6200 core has a configurable feature to support data and instruction address-range triggered breakpoints, where a breakpoint can occur when a virtual address is either within or outside a pair of 32-bit addresses. Unlike the traditional address-mask control, address-range triggering is not restricted to a power-of-two binary boundary.

Complex breakpoints utilize the simple instruction and data breakpoints and cause a breakpoint exception when particular combinations of events are seen. Complex breakpoint features include:

- Pass Counters - Each time a matching condition is seen, a counter is decremented. The break or trigger will only be enabled when the counter has counted down to 0.
- Tuples - A tuple is the pairing of an instruction and a data breakpoint. The tuple will match if both the virtual address of the load or store instruction matches the instruction breakpoint, and the data breakpoint of the resulting load or store address and optional data value matches.
- Priming - This allows a breakpoint to be enabled only after other break conditions have been met. Also called sequential or armed triggering.
- Qualified - This feature uses a data breakpoint to qualify when an instruction breakpoint can be taken. Once a load matches the data address and the data value, the instruction break will be enabled. If a load matches the address, but has mis-matching data, the instruction break will be disabled.

Performance Counters

Performance counters are used to accumulate occurrences of internal predefined events/cycles/conditions for program analysis, debug, or profiling. A few examples of event types are clock cycles, instructions executed, specific instruction types executed, loads, stores, exceptions, and cycles while the CPU is stalled. There are two, 32-bit counters. Each can count one of the 64 internal predefined events selected by a corresponding control register. A counter overflow can be programmed to generate an interrupt, where the interrupt handler software can maintain larger total counts.

PC/Address Sampling

This sampling function is used for program profiling and hot-spots analysis. Instruction PC and/or Load/Store addresses can be sampled periodically. The result is scanned out through the APB Debug port. The Debug Control Register (DCR) is used to specify the sample period and the sample trigger.

Fast Debug Channel (FDC)

The M6200 core includes optional FDC as a mechanism for high bandwidth data transfer between a debug host/probe and a target. FDC provides a FIFO buffering scheme to transfer data serially, with low CPU overhead and minimized waiting time. The data transfers occur in the background, and the target CPU can either choose to check the status of the transfer periodically, or it can choose to be interrupted at the end of the transfer.

iFlowtrace™

The M6200 core has an option for a simple trace mechanism called iFlowtrace. This mechanism only traces the instruction PC, not data addresses or values. This simplification allows the trace block to be smaller and the trace compression to be more efficient. iFlowtrace memory can be configured as off-chip, on-chip, or both.

iFlowtrace also offers special-event trace modes when normal tracing is disabled, namely:

- Function Call/Return and Exception Tracing mode to trace the PC value of function calls and returns and/or exceptions and returns.
- Breakpoint Match mode traces the breakpoint ID of a matching breakpoint and, for data breakpoints, the PC value of the instruction that caused it.

- Filtered Data Tracing mode traces the ID of a matching data breakpoint, the load or store data value, access type and memory access size, and the low-order address bits of the memory access, which is useful when the data breakpoint is set up to match a range of addresses.
- User Trace Messages. The user can instrument their code to add their own 32-bit value messages into the trace by writing to the two Cop0 UTM registers.
- Delta Cycle mode works in combination with the above trace modes to provide a timestamp between stored events. It reports the number of cycles that have elapsed since the last message was generated and put into the trace.

Secure Debug

Secure Debug improves security by disabling untrusted debug access. The *EJ_DisableProbeDebug* signal disables all debug functionality (including trace), with the exception of an override for a probe to generate the Debug Interrupt Exception, OciBrk. FDC remains enabled.

Testability

Testability for production testing of the core is supported through the use of internal scan and memory BIST.

Internal Scan

Full mux-based scan for maximum test coverage is supported, with a configurable number of scan chains. ATPG test coverage can exceed 99%, depending on standard cell libraries and configuration options.

Memory BIST

Memory BIST for the on-chip trace memory is optional.

User-specified Memory BIST

Memory BIST can be inserted with a CAD tool or other user-specified method. Wrapper modules and special side-band signal buses of configurable width are provided within the core to facilitate this approach.

Build-Time Configuration Options

The M6200 core allows a number of features to be customized based on the intended application. Table 5 summarizes the key configuration options that can be selected when the core is synthesized and implemented.

For a core that has already been built, software can determine the value of many of these options by checking an appropriate register field. Refer to the *MIPS32® M6200 Processor Core Family Programmer's Guide* for a more complete description of these fields. The value of some options that do not have a functional effect on the core are not visible to software.

Table 5 CPU Build-time Configuration Options

Feature	Options	Software Visibility
Integer register file sets	1, 2, 4, 8 or 16	$SRSCtl_{HSS}$
DSP Module	Present or not	$Config3_{DSPP}$ and $Config3_{DSP2P}$
Memory Protection Unit	Present or not. If present, 1 - 16 regions	N/A
Debug Controller via APB Port	Present or not	N/A
Fast Debug Channel (FDC)	Present or not	DCR_{FDCI}
Instruction/data hardware breakpoints	0/0, 4/2, or 8/4	$DCR_{InstBrk}$, IBS_{BCN} $DCR_{DataBrk}$, DBS_{BCN}
Hardware breakpoint trigger	By address match, or address match and address range	$IBCn_{hwart}$ and $DBCn_{hwart}$
Complex breakpoints	0/0 or 8/4	DCR_{CBT}

* These bits indicate the presence of an external block. Bits will not be set if interface is present, but block is not.

Table 5 CPU Build-time Configuration Options (Continued)

Feature	Options	Software Visibility
Performance Counters	Present or not	*Config1*$_{PC}$
iFlowtrace hardware	Present or not	*Config3*$_{ITL}$
iFlowtrace on-chip trace memory size	256B - 8MB	ITCBRDP
iFlowtrace off-chip PIB	Present or not	IFCTL$_{OfC}$
Coprocessor2 interface	Present or not	*Config1*$_{C2*}$
Interrupt Controller (ICU)	Present or not. If present, 8, 16, 32, 64, 128, 256 interrupts, interrupt polarity and edge/level sensitivity, and mapped shadow set value.	N/A
SRAM ECC	Present or not	*ErrCtl*$_{EE}$
Parity on data and address bus	Present or not	*ErrCtl*$_{PE}$
Interrupt synchronizers	Present or not	N/A
Interrupt Vector Offset	Compute from Vector Input or Immediate Offset	N/A
* These bits indicate the presence of an external block. Bits will not be set if interface is present, but block is not.		

Revision History

Revision	Date	Description
01.00	December 15, 2015	• Release of document for RC 1.0

Public. This publication contains proprietary information which is subject to change without notice and is supplied ‘as is’, without any warranty of any kind.